Algorithms

**Divide and Conquer**
- top down, subproblems may be repeated, recursive spec

**Dynamic Programming**
- bottom up, recursive spec, pseudo-polynomial time for some NP-complete problems

**Greedy Method**
- guaranteed polynomial time, must be a matroid problem

**Depth First Search**
- efficient graph traversal for certain problems such as planarity testing, bi-connectedness, topological sort

**Backtracking**
- general problem solving algorithm, could go exponential

**Branch and Bound**
- general solver in case of optimization
Algorithms

Divide and Conquer

**Description:** Let $I$ be an instance of a problem – the template for Divide-and-conquer is as follows:

**DC**($I$):

If $\text{size}(I) < \text{small}$ then return $\text{solution}(I)$

Otherwise

Divide $I$ into subinstances $I_1, I_2, ...$

Return $\text{combine}(\text{DC}(I_1), \text{DC}(I_2), ...)$
Algorithms

Divide and Conquer

Example: sort a list of $n$ integers

Instance: a vector $A$ of integers

Produce: a vector $A'$ s.t. $e \in A$ iff $e \in A'$; length($A$) == length($A'$);
if $e_1, e_2 \in A$ and $e_1 < e_2$ then $e_1$ has a lower index in $A'$ than $e_2$

Quicksort($A$):
If $|A| \leq 1$ then return $A$
Otherwise
Choose a number $p$ randomly from $A$
$A_1 = \{ b : b \in A, b < p \}$
$A_2 = \{ b : b \in A, b == p \}$
$A_3 = \{ b : b \in A, b > p \}$
Return Quicksort($A_1$) $\| A_2 \|$ Quicksort($A_3$)
Algorithms

Divide and Conquer

Example: the Satisfiability problem

Definitions and structures:

- a **variable**: denoted \( \nu \), takes value 1 or value 0, or is unassigned
- a **positive literal**: a variable, denoted \( \nu \)
- a **negative literal**: denoted \( \neg \nu \), takes value opposite \( \nu \) or unassigned
- a **literal**: a positive or negative literal
- a **clause**: a set of literals, takes value 1 only if one of its literals takes value 1, takes value 0 if all of its literals take value 0, is unassigned otherwise, empty clause has value 0
  
  e.g. \{ \nu_1, \neg \nu_4, \neg \nu_6, \nu_{10} \}

- a **Conjunctive Normal Form expression (CNF)**: a set of clauses, takes value 1 only if all clauses take value 1, takes value 0 if some clause takes value 0, is unassigned otherwise

  \{ \{ \nu_1, \neg \nu_4, \neg \nu_6, \nu_{10} \}, \{ \neg \nu_1, \nu_2, \neg \nu_3 \}, \{ \nu_2, \nu_3 \}, \{ \neg \nu_3, \neg \nu_5, \neg \nu_6 \} \}
Algorithms

Divide and Conquer

Example: the Satisfiability problem

Instance: a Boolean expression $I$ in Conjunctive Normal Form

Output: $true$ if there exists an assignment of values to variables of $I$ that causes $I$ to evaluate to 1 or $false$ otherwise

$SAT(I)$:

If $I == \emptyset$ then return $true$

Otherwise, if there is a $c \in I$ s.t. $c == \emptyset$ return $false$

Otherwise

Choose a variable $v$ contained in a clause of $I$

$I_0 = \{ c - \{v\} : c \in I, \neg v \not\in c \}$

$I_1 = \{ c - \{\neg v\} : c \in I, v \not\in c \}$

Return $SAT(I_0) \lor SAT(I_1)$
Algorithms

Divide and Conquer

Example: the Satisfiability problem

\[
\{\{v_1, \neg v_4, \neg v_6\}, \{\neg v_1, v_2, \neg v_3\}, \{v_2, v_3\}, \{\neg v_3, \neg v_5, \neg v_6\}\}
\]

\[v_3 = 1\]

\[
\{\{v_1, \neg v_4, \neg v_6\}, \{\neg v_1, v_2\}, \{\neg v_5, \neg v_6\}\}
\]

\[I_1 = \{c - \{-v\} : c \in I, v \not\in c \}\]

\[v_3 = 0\]

\[
\{\{v_1, \neg v_4, \neg v_6\}, \{v_2\}\}
\]

\[I_0 = \{c - \{v\} : c \in I, \neg v \not\in c \}\]
Dynamic Programming

**Description:** A tabular structure is maintained. The $i^{th}$ row has subproblems of size $i$. The $j^{th}$ column at row $i$ holds the solution to the $j^{th}$ subproblem of size $i$. Let $I$ denote a problem instance. Here is the template:

**DP($I$):**

For each subproblem $j$ of size == 1

\[
\text{TABLE}[1,j] = \text{solution to } j^{th} \text{ subproblem of size == 1}
\]

For $i=2$ to $n$

For each subproblem $j$ of size == $i$

\[
\text{TABLE}[i,j] = \text{solution to } j^{th} \text{ subproblem of size == } i
\]

Search TABLE[$n,\ast$] for the solution to $I$
Dynamic Programming

Example: find the \( n \)\(^{\text{th}} \) Fibonacci number

Instance: a number \( n \)

Produce: a solution to \( F(n) = F(n-1) + F(n-2); F(1)=1; F(2)=1; \)

\( F(n) \):

\[
\begin{align*}
\text{TABLE}[1,0] & = 1 \\
\text{TABLE}[2,0] & = 1 \\
\text{For } i=3 \text{ to } n \\
\text{TABLE}[i,0] & = \text{TABLE}[i-1,0] + \text{TABLE}[i-2,0] \\
\text{Return } \text{TABLE}[n,0]
\end{align*}
\]
**Algorithms**

**Dynamic Programming**

**Example:** find the \( n^{\text{th}} \) Fibonacci number

**Instance:** a number \( n \)

**Produce:** a solution to \( F(n) = F(n-1) + F(n-2); \) \( F(1)=1; \) \( F(2)=1; \)

\( F(n): \)

\[
\begin{align*}
\text{TABLE}[1,0] &= 1 \\
\text{TABLE}[2,0] &= 1 \\
\text{For } i=3 \text{ to } n \\
\quad \text{TABLE}[i,0] &= \text{TABLE}[i-1,0] + \text{TABLE}[i-2,0] \\
\text{Return } \text{TABLE}[n,0]
\end{align*}
\]
Algorithms

Dynamic Programming

Example: find the $n^{th}$ Fibonacci number

Instance: a number $n$

Produce: a solution to $F(n) = F(n-1) + F(n-2)$; $F(1) = 1$; $F(2) = 1$;

$F(n)$:

- TABLE[1,0] = 1
- TABLE[2,0] = 1

For $i=3$ to $n$

- TABLE[i,0] = TABLE[i-1,0] + TABLE[i-2,0]

Return TABLE[n,0]
### Dynamic Programming

**Example:** find the $n$th Fibonacci number

**Instance:** a number $n$

**Produce:** a solution to $F(n) = F(n-1) + F(n-2)$; $F(1) = 1$; $F(2) = 1$

**$F(n)$:**

<table>
<thead>
<tr>
<th>$i$</th>
<th>$F(i)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
</tr>
<tr>
<td>4</td>
<td>3</td>
</tr>
<tr>
<td>5</td>
<td>5</td>
</tr>
</tbody>
</table>

TABLE:

- $TABLE[1,0] = 1$
- $TABLE[2,0] = 1$
- For $i = 3$ to $n$
  - $TABLE[i,0] = TABLE[i-1,0] + TABLE[i-2,0]$

Return $TABLE[n,0]$
**Dynamic Programming**

**Example:** find the $n^{th}$ Fibonacci number

**Instance:** a number $n$

**Produce:** a solution to $F(n) = F(n-1) + F(n-2)$; $F(1)=1$; $F(2)=1$;

$F(n)$:

- $TABLE[1,0] = 1$
- $TABLE[2,0] = 1$
- For $i=3$ to $n$
  - $TABLE[i,0] = TABLE[i-1,0] + TABLE[i-2,0]$
- Return $TABLE[n,0]$

<p>| | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>8</td>
<td>5</td>
<td>3</td>
<td>2</td>
<td>1</td>
</tr>
</tbody>
</table>
Dynamic Programming

Example: find the $n^{th}$ Fibonacci number

Instance: a number $n$

Produce: a solution to $F(n) = F(n-1) + F(n-2)$; $F(1)=1$; $F(2)=1$;

$F(n)$:

- $\text{TABLE}[1,0] = 1$
- $\text{TABLE}[2,0] = 1$
- For $i=3$ to $n$
  - $\text{TABLE}[i,0] = \text{TABLE}[i-1,0] + \text{TABLE}[i-2,0]$
- Return $\text{TABLE}[n,0]$

<table>
<thead>
<tr>
<th>$n$</th>
</tr>
</thead>
<tbody>
<tr>
<td>13</td>
</tr>
<tr>
<td>8</td>
</tr>
<tr>
<td>5</td>
</tr>
<tr>
<td>3</td>
</tr>
<tr>
<td>2</td>
</tr>
<tr>
<td>1</td>
</tr>
<tr>
<td>1</td>
</tr>
</tbody>
</table>
**Algorithms**

**Dynamic Programming**

**Example:** find the \( n \)th Fibonacci number

**Instance:** a number \( n \)

**Produce:** a solution to \( \text{F}(n) = \text{F}(n-1) + \text{F}(n-2) \); \( \text{F}(1)=1; \text{F}(2)=1; \)

\( \text{F}(n) \):

- \( \text{TABLE}[1,0] = 1 \)
- \( \text{TABLE}[2,0] = 1 \)

For \( i=3 \) to \( n \)

- \( \text{TABLE}[i,0] = \text{TABLE}[i-1,0] + \text{TABLE}[i-2,0] \)

Return \( \text{TABLE}[n,0] \)
Algorithms

Dynamic Programming

Example: find a subset of a set of integers that sums to $m$

Instance: a set $A$ of integers and a function $size: A \rightarrow \mathbb{N}^+$

Output: true if there is a subset $A' \subseteq A$ s.t. $\sum_{a \in A'} size(a) = m$, false otherwise

Let $A = \{a_1, a_2, a_3, \ldots, a_n\}$

$TABLE[i,j]$: true if there is a subset of $\{a_1, a_2, \ldots, a_i\}$ that sums to $j$, false otherwise

$TABLE[1,j] = true$ if $j=0$ or if $j=size(a_1)$

false otherwise

$TABLE[i,j] = TABLE[i-1,j] \lor TABLE[i-1,j-size(a_i)]$, $i > 1$
Algorithms

Dynamic Programming

Example: find a subset of a set of integers that sums to \( m \)

Instance: a set \( A \) of integers and a function \( size: A \rightarrow \mathbb{N}^+ \)

Output: \( true \) if there is a subset \( A' \subseteq A \) s.t. \( \sum_{a \in A'} size(a) = m \), \( false \) otherwise

\textbf{Partition}(A,m):

For \( j=1 \) to \( m \) \hspace{1em} TABLE[1, j] = false

TABLE[1, 0] = true

TABLE[1, size(\( a_1 \))] = true

For \( i=2 \) to \( |A| \)

For \( j=0 \) to \( m \)

TABLE[\( i,j \)] = TABLE[\( i-1,j \) \( \lor \) TABLE[\( i-1,j-size(a_i)\)]]

Output TABLE[\( |A|, m \)]
# Dynamic Programming

**Example:** $A = \{3, 6, 4, 8, 7\}$, $m = 17$

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
<th>12</th>
<th>13</th>
<th>14</th>
<th>15</th>
<th>16</th>
<th>17</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>3,6,4,8,7</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>3,6,4,8</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>3,6,4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>3,6</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Dynamic Programming

Example: $A = \{3, 6, 4, 8, 7\}$, $m = 17$

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
<th>12</th>
<th>13</th>
<th>14</th>
<th>15</th>
<th>16</th>
<th>17</th>
</tr>
</thead>
<tbody>
<tr>
<td>{3,6,4,8,7}</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>{3,6,4,8}</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>{3,6,4}</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>{3,6}</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>{3}</td>
<td>T</td>
<td>T</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
**Algorithms**

**Dynamic Programming**

**Example:** $A = \{3, 6, 4, 8, 7\}$, $m = 17$

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
<th>12</th>
<th>13</th>
<th>14</th>
<th>15</th>
<th>16</th>
<th>17</th>
</tr>
</thead>
<tbody>
<tr>
<td>{3,6,4,8,7}</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>{3,6,4,8}</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>{3,6,4}</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>{3,6}</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>{3}</td>
<td>T</td>
<td>T</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Dynamic Programming

Example: \( A = \{3, 6, 4, 8, 7\}, \ m = 17 \)
**Dynamic Programming**

**Example:** $A = \{3, 6, 4, 8, 7\}, \ m = 17$

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
<th>12</th>
<th>13</th>
<th>14</th>
<th>15</th>
<th>16</th>
<th>17</th>
</tr>
</thead>
<tbody>
<tr>
<td>${3,6,4,8,7}$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>${3,6,4,8}$</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
</tr>
<tr>
<td>${3,6,4}$</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>${3,6}$</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>${3}$</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Dynamic Programming

Example: $A = \{3, 6, 4, 8, 7\}, m = 17$

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
<th>12</th>
<th>13</th>
<th>14</th>
<th>15</th>
<th>16</th>
<th>17</th>
</tr>
</thead>
<tbody>
<tr>
<td>${3,6,4,8,7}$</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
</tr>
<tr>
<td>${3,6,4,8}$</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
</tr>
<tr>
<td>${3,6,4}$</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
</tr>
<tr>
<td>${3,6}$</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
</tr>
<tr>
<td>${3}$</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
</tr>
</tbody>
</table>
Algorithms

Dynamic Programming

Example: the Knapsack problem

Instance: a set $A$ of objects, a function $\text{size}: A \rightarrow \mathbb{N}^+$, a function $\text{value}: A \rightarrow \mathbb{N}^+$, and an integer capacity $B$

Output: a subset $A^* \subseteq A$ whose total value is maximum over all subsets of $A' \subseteq A$ such that $\sum_{a \in A'} \text{size}(a) \leq B$

Let $A = \{a_1, a_2, a_3, \ldots, a_n\}$

TABLE[$i,j$]: a subset of $\{a_1, a_2, \ldots, a_i\}$ whose total size is $j$ and whose total value is at least as great as the total value of any subset of $\{a_1, a_2, \ldots, a_i\}$ whose total size is $j$.

TABLE[1,$j$] = 0 if $j = 0$

$= \text{value}(a_1)$ if $j = \text{size}(a_1)$

$= -1$ otherwise

TABLE[$i,j$] = max(TABLE[$i-1,j$], $\text{value}(a_i) + \text{TABLE}[i-1,j-\text{size}(a_i)]$)

where $i > 1$ and TABLE[1,$j$] $\geq 0$ and TABLE[1,j-$\text{size}(a_i)$] $\geq 0$
Algorithms

Dynamic Programming

Example: the Knapsack problem

Knapsack(A,B):

For \( j = 1 \) to \( B \)
\[
\text{TABLE}[1, j] = -1
\]
\[
\text{TABLE}[1, 0] = 0
\]
\[
\text{TABLE}[1, \text{size}(a_1)] = \text{value}(a_1)
\]

For \( i = 2 \) to \( |A| \)

For \( j = 0 \) to \( B \)

\[
\text{TABLE}[i, j] = \max(\text{TABLE}[i-1, j], \text{value}(a_i) + \text{TABLE}[i-1, j - \text{size}(a_i)])
\]

\( V = \text{TABLE}[|A|, 0] \)

For \( i = 1 \) to \( B \)

If \( \text{TABLE}[|A|, i] > V \) then \( V = \text{TABLE}[|A|, i] \)

Output \( V \)
**Algorithms**

**Dynamic Programming**

**Example:** \( A = \{ [3;16], [6;19], [4;9], [8;12], [7;14] \}, \ B = 17 \)

\[ a_i = [X;Y] \]

- value\( (a_i) \)
- size\( (a_i) \)

\[
\begin{array}{cccccccccccccccc}
0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 & 11 & 12 & 13 & 14 & 15 & 16 & 17 \\
\{3,6,4,8,7\} & 0 & & & & & & & & & & & & & & & & \\
\{3,6,4,8\} & 0 & & & & & & & & & & & & & & & & \\
\{3,6,4\} & 0 & & & & & & & & & & & & & & & & \\
\{3,6\} & 0 & & & & & & & & & & & & & & & & \\
\{3\} & 0 & -1 & -1 & 16 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 \\
\end{array}
\]
Dynamic Programming

Example: \( A = \{[3;16], [6;19], [4;9], [8;12], [7;14]\}, B = 17 \)

\[ a_i = [X;Y] \]

\[ \text{value}(a_i) \]

\[ \text{size}(a_i) \]

\[
\begin{array}{cccccccccccccccc}
0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 & 11 & 12 & 13 & 14 & 15 & 16 & 17 \\
\{3,6,4,8,7\} & 0 & & & & & & & & & & & & & & & & \\
\{3,6,4,8\} & 0 & & & & & & & & & & & & & & & & \\
\{3,6,4\} & 0 & & & & & & & & & & & & & & & & \\
\{3,6\} & 0 & & & & & 16 & & & & & & & & & & \\
\{3\} & 0 & -1 & -1 & 16 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 \\
\end{array}
\]

Subset = \{[3;16]\}
Dynamic Programming

Example: \( A = \{[3;16], [6;19], [4;9], [8;12], [7;14]\}, B = 17 \)

\[ a_i = [X;Y] \]

\[ \text{value}(a_i) \]

\[ \text{size}(a_i) \]

\[
\begin{array}{cccccccccccccccc}
0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 & 11 & 12 & 13 & 14 & 15 & 16 & 17 \\
\{3,6,4,8,7\} & 0 & & & & & & & & & & & & & & & \\
\{3,6,4,8\} & 0 & & & & & & & & & & & & & & & \\
\{3,6,4\} & 0 & & & & & & & & & & & & & & & \\
\{3,6\} & 0 & 16 & & & & & & & & & & & & & & & \\
\{3\} & 0 & -1 & -1 & 16 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1
\end{array}
\]

Subset = \{[6;19]\}
Algorithms

Dynamic Programming

Example: \( A = \{ [3; 16], [6; 19], [4; 9], [8; 12], [7; 14] \}, \ B = 17 \)

\( a_i = [X; Y] \)

- \( \text{value}(a_i) \)
- \( \text{size}(a_i) \)

\[
\begin{array}{cccccccccccccccc}
0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 & 11 & 12 & 13 & 14 & 15 & 16 & 17 \\
\{3,6,4,8,7\} & 0 & & & & & & & & & & & & & & & \\
\{3,6,4,8\} & 0 & & & & & & & & & & & & & & & \\
\{3,6,4\} & 0 & & & & & & & & & & & & & & & \\
\{3,6\} & 0 & 16 & 19 & & & & & & & & & & & & & & \\
\{3\} & 0 & -1 & -1 & 16 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & \\
\end{array}
\]

Subset = \{ [3:16], [6:19] \}
## Algorithms

### Dynamic Programming

**Example:** \( A = \{ [3;16], [6;19], [4;9], [8;12], [7;14] \} \), \( B = 17 \)

\[ a_i = [X;Y] \]

<table>
<thead>
<tr>
<th>{3,6,4,8,7}</th>
<th>0</th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>{3,6,4,8}</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>{3,6,4}</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>{3,6}</td>
<td>0</td>
<td>-1</td>
<td>-1</td>
<td>16</td>
<td>-1</td>
<td>-1</td>
<td>19</td>
<td>-1</td>
<td>-1</td>
<td>35</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
</tr>
<tr>
<td>{3}</td>
<td>0</td>
<td>-1</td>
<td>-1</td>
<td>16</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
</tr>
</tbody>
</table>

\[ value(a_i) \]

\[ size(a_i) \]
### Algorithms

#### Dynamic Programming

**Example:** \( A = \{ [3; 16], [6; 19], [4; 9], [8; 12], [7; 14] \}, \ B = 17 \)

\[
\begin{align*}
  a_i &= [X; Y] \\
  \text{value}(a_i) \\
  \text{size}(a_i)
\end{align*}
\]

\[
\begin{array}{cccccccccccccccc}
  & 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 & 11 & 12 & 13 & 14 & 15 & 16 & 17 \\
\{3,6,4,8,7\} & 0 & & & & & & & & & & & & & & & & \\
\{3,6,4,8\} & 0 & & & & & & & & & & & & & & & & \\
\{3,6,4\} & 0 & & & & & & & & & & & & & & & & 16 \\
\{3,6\} & 0 & -1 & -1 & 16 & -1 & -1 & 19 & -1 & -1 & 35 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & \\
\{3\} & 0 & -1 & -1 & 16 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1
\end{array}
\]

**Subset** = \{ [[6; 19]] \}
## Dynamic Programming

**Example:** \( A = \{ [3;16], [6;19], [4;9], [8;12], [7;14] \}, \ B = 17 \)

\[ a_i = [X;Y] \]

\[ \text{value}(a_i) \]

\[ \text{size}(a_i) \]

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
<th>12</th>
<th>13</th>
<th>14</th>
<th>15</th>
<th>16</th>
<th>17</th>
</tr>
</thead>
<tbody>
<tr>
<td>{3,6,4,8,7}</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>{3,6,4,8}</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>{3,6,4}</td>
<td>0</td>
<td></td>
<td>16</td>
<td>9</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>{3,6}</td>
<td>0</td>
<td>-1</td>
<td>-1</td>
<td>16</td>
<td>-1</td>
<td>-1</td>
<td>19</td>
<td>-1</td>
<td>-1</td>
<td>35</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
</tr>
<tr>
<td>{3}</td>
<td>0</td>
<td>-1</td>
<td>-1</td>
<td>16</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
</tr>
</tbody>
</table>

Subset = \{ [4;9] \}
### Dynamic Programming

**Example:**  
\[ A = \{ [3;16], [6;19], [4;9], [8;12], [7;14] \}, \quad B = 17 \]

\[ a_i = [X;Y] \]

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
<th>12</th>
<th>13</th>
<th>14</th>
<th>15</th>
<th>16</th>
<th>17</th>
</tr>
</thead>
<tbody>
<tr>
<td>{3,6,4,8,7}</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>{3,6,4,8}</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>{3,6,4}</td>
<td>0</td>
<td>16</td>
<td>9</td>
<td>19</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>{3,6}</td>
<td>0</td>
<td>-1</td>
<td>-1</td>
<td>16</td>
<td>-1</td>
<td>-1</td>
<td>19</td>
<td>-1</td>
<td>-1</td>
<td>35</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
</tr>
<tr>
<td>{3}</td>
<td>0</td>
<td>-1</td>
<td>-1</td>
<td>16</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
</tr>
</tbody>
</table>

**Subset = \{ [6;19] \}**
### Algorithms

#### Dynamic Programming

**Example:** $A = \{[3;16], [6;19], [4;9], [8;12], [7;14]\}, B = 17$

$a_i = [X;Y]$

- **value($a_i$)**
- **size($a_i$)**

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
<th>12</th>
<th>13</th>
<th>14</th>
<th>15</th>
<th>16</th>
<th>17</th>
</tr>
</thead>
<tbody>
<tr>
<td>{3,6,4,8,7}</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>{3,6,4,8}</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>{3,6,4}</td>
<td>0</td>
<td>16</td>
<td>9</td>
<td>19</td>
<td>25</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>{3,6}</td>
<td>0</td>
<td>-1</td>
<td>-1</td>
<td>16</td>
<td>-1</td>
<td>-1</td>
<td>19</td>
<td>-1</td>
<td>-1</td>
<td>35</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
</tr>
<tr>
<td>{3}</td>
<td>0</td>
<td>-1</td>
<td>-1</td>
<td>16</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
</tr>
</tbody>
</table>

Subset = \{[3;16],[4;9]\}
## Algorithms

### Dynamic Programming

**Example:** $A = \{[3;16], [6;19], [4;9], [8;12], [7;14]\}$, $B = 17$

$a_i = [X;Y]$

- **value($a_i$)**
- **size($a_i$)**

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
<th>12</th>
<th>13</th>
<th>14</th>
<th>15</th>
<th>16</th>
<th>17</th>
</tr>
</thead>
<tbody>
<tr>
<td>{3,6,4,8,7}</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>{3,6,4,8}</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>{3,6,4}</td>
<td>0</td>
<td>16</td>
<td>9</td>
<td>19</td>
<td>25</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>{3,6}</td>
<td>0</td>
<td>-1</td>
<td>-1</td>
<td>16</td>
<td>-1</td>
<td>-1</td>
<td>19</td>
<td>-1</td>
<td>-1</td>
<td>35</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
</tr>
<tr>
<td>{3}</td>
<td>0</td>
<td>-1</td>
<td>-1</td>
<td>16</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
</tr>
</tbody>
</table>

Subset = {[3;16],[6;19]}
**Algorithms**

**Dynamic Programming**

**Example:** \( A = \{[3;16], [6;19], [4;9], [8;12], [7;14]\}, \ B = 17 \)

\[ a_i = [X;Y] \]

- \( \text{value}(a_i) \)
- \( \text{size}(a_i) \)

\[
\begin{array}{cccccccccccccccc}
0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 & 11 & 12 & 13 & 14 & 15 & 16 & 17 \\
\{3, 6, 4, 8, 7\} & 0 & & & & & & & & & & & & & & \\
\{3, 6, 4, 8\} & 0 & & & & & & & & & & & & & & \\
\{3, 6, 4\} & 0 & 16 & 9 & 19 & 25 & & & 35 & & & & & & & & \\
\{3, 6\} & 0 & -1 & -1 & 16 & -1 & -1 & 19 & -1 & -1 & & & & & & & & \\
\{3\} & 0 & -1 & -1 & 16 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 \\
\end{array}
\]

Subset = \([6;19],[4;9]\)
### Dynamic Programming

**Example:** $A = \{[3;16], [6;19], [4;9], [8;12], [7;14]\}$, $B = 17$

Let $a_i = [X;Y]$ then:

- $value(a_i)$
- $size(a_i)$

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
<th>12</th>
<th>13</th>
<th>14</th>
<th>15</th>
<th>16</th>
<th>17</th>
</tr>
</thead>
<tbody>
<tr>
<td>{3,6,4,8,7}</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>{3,6,4,8}</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>{3,6,4}</td>
<td>0</td>
<td></td>
<td>16</td>
<td>9</td>
<td>19</td>
<td>25</td>
<td>35</td>
<td>28</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>{3,6}</td>
<td>0</td>
<td>-1</td>
<td>-1</td>
<td>16</td>
<td>-1</td>
<td>-1</td>
<td>19</td>
<td>-1</td>
<td>-1</td>
<td>35</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
</tr>
<tr>
<td>{3}</td>
<td>0</td>
<td>-1</td>
<td>-1</td>
<td>16</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
</tr>
</tbody>
</table>

subset = $\{[3;16],[6;19],[4;9]\}$
### Dynamic Programming

**Example:** \( A = \{ [3;16], [6;19], [4;9], [8;12], [7;14] \}, \ B = 17 \)

\[ a_i = [X;Y] \]

\[ \text{value}(a_i) \]

\[ \text{size}(a_i) \]

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
<th>12</th>
<th>13</th>
<th>14</th>
<th>15</th>
<th>16</th>
<th>17</th>
</tr>
</thead>
<tbody>
<tr>
<td>{3,6,4,8,7}</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>{3,6,4,8}</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>{3,6,4}</td>
<td>0</td>
<td>-1</td>
<td>-1</td>
<td>16</td>
<td>9</td>
<td>-1</td>
<td>19</td>
<td>25</td>
<td>-1</td>
<td>35</td>
<td>28</td>
<td>-1</td>
<td>-1</td>
<td>44</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
</tr>
<tr>
<td>{3,6}</td>
<td>0</td>
<td>-1</td>
<td>-1</td>
<td>16</td>
<td>-1</td>
<td>-1</td>
<td>19</td>
<td>-1</td>
<td>-1</td>
<td>35</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
</tr>
<tr>
<td>{3}</td>
<td>0</td>
<td>-1</td>
<td>-1</td>
<td>16</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
</tr>
</tbody>
</table>
**Algorithms**

**Dynamic Programming**

**Example:** \( A = \{ [3;16], [6;19], [4;9], [8;12], [7;14] \} \), \( B = 17 \)

\[ a_i = [X;Y] \]

- \( \text{value}(a_i) \)
- \( \text{size}(a_i) \)

\[
\begin{array}{cccccccccccccccccc}
0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 & 11 & 12 & 13 & 14 & 15 & 16 & 17 \\
\{3,6,4,8,7\} & 0 &  &  &  &  &  &  &  &  &  &  &  &  &  &  &  &  &  \\
\{3,6,4,8\} & 0 &  & 16 &  &  &  &  &  &  &  &  &  &  &  &  &  &  &  \\
\{3,6,4\} & 0 & -1 & -1 & 16 & 9 & -1 & 19 & 25 & -1 & 35 & 28 & -1 & -1 & 44 & -1 & -1 & -1 & -1 \\
\{3,6\} & 0 & -1 & -1 & 16 & -1 & -1 & 19 & -1 & -1 & 35 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 \\
\{3\} & 0 & -1 & -1 & 16 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 \\
\end{array}
\]

Subset = \{\{3;16\}\}
## Dynamic Programming

**Example:** \( A = \{ [3;16], [6;19], [4;9], [8;12], [7;14] \}, B = 17 \)

\[ a_i = [X;Y] \]

\[ \text{value}(a_i) \]

\[ \text{size}(a_i) \]

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
<th>12</th>
<th>13</th>
<th>14</th>
<th>15</th>
<th>16</th>
<th>17</th>
</tr>
</thead>
<tbody>
<tr>
<td>{3,6,4,8,7}</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>{3,6,4,8}</td>
<td>0</td>
<td></td>
<td>16</td>
<td>9</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>{3,6,4}</td>
<td>0</td>
<td>-1</td>
<td>-1</td>
<td>16</td>
<td>9</td>
<td>-1</td>
<td>19</td>
<td>25</td>
<td>-1</td>
<td>35</td>
<td>28</td>
<td>-1</td>
<td>-1</td>
<td>44</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
</tr>
<tr>
<td>{3,6}</td>
<td>0</td>
<td>-1</td>
<td>-1</td>
<td>16</td>
<td>-1</td>
<td>-1</td>
<td>19</td>
<td>-1</td>
<td>-1</td>
<td>35</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
</tr>
<tr>
<td>{3}</td>
<td>0</td>
<td>-1</td>
<td>-1</td>
<td>16</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
</tr>
</tbody>
</table>

Subset = \{ [[4;9]] \}
Dynamic Programming

Example: \( A = \{[3;16], [6;19], [4;9], [8;12], [7;14]\}, B = 17 \)

\[ a_i = [X;Y] \]

\[ \text{value}(a_i) \]

\[ \text{size}(a_i) \]

|       | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 |
|-------|---|---|---|---|---|---|---|---|---|---|----|----|----|----|----|----|----|----|----|
| \{3,6,4,8,7\} | 0 |   |   |   |   |   |   |   |   |   |    |    |    |    |    |    |    |    |    |
| \{3,6,4,8\}   | 0 | 16| 9 | 19|   |   |   |   |   |   |    |    |    |    |    |    |    |    |    |
| \{3,6,4\}     | 0 |-1 |-1 |16| 9 |-1 |19 |25 |-1 |35 |28 |-1 |-1 |44 |-1 |-1 |-1 |-1 |-1 |    |
| \{3,6\}       | 0 |-1 |-1 |16|-1 |-1 |19 |-1 |-1 |35 |-1 |-1 |-1 |-1 |-1 |-1 |-1 |-1 |-1 |    |
| \{3\}         | 0 |-1 |-1 |16|-1 |-1 |-1 |-1 |-1 |-1 |-1 |-1 |-1 |-1 |-1 |-1 |-1 |-1 |-1 |    |

Subset = \{[6;19]\}
### Algorithms

#### Dynamic Programming

**Example:**  \( A = \{[3;16], [6;19], [4;9], [8;12], [7;14]\} \), \( B = 17 \)

\[ a_i = [X;Y] \]

- **value** \((a_i)\)
- **size** \((a_i)\)

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
<th>12</th>
<th>13</th>
<th>14</th>
<th>15</th>
<th>16</th>
<th>17</th>
</tr>
</thead>
<tbody>
<tr>
<td>{3,6,4,8,7}</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>{3,6,4,8}</td>
<td>0</td>
<td>16</td>
<td>9</td>
<td>19</td>
<td>25</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>{3,6,4}</td>
<td>0</td>
<td>-1</td>
<td>-1</td>
<td>16</td>
<td>9</td>
<td>-1</td>
<td>19</td>
<td>25</td>
<td>-1</td>
<td>35</td>
<td>28</td>
<td>-1</td>
<td>-1</td>
<td>44</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
</tr>
<tr>
<td>{3,6}</td>
<td>0</td>
<td>-1</td>
<td>-1</td>
<td>16</td>
<td>-1</td>
<td>-1</td>
<td>19</td>
<td>-1</td>
<td>-1</td>
<td>35</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
</tr>
<tr>
<td>{3}</td>
<td>0</td>
<td>-1</td>
<td>-1</td>
<td>16</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
</tr>
</tbody>
</table>

**Subset = \{[3;16],[4;9]\}**
### Dynamic Programming

**Example:** \( A = \{ [3;16], [6;19], [4;9], [8;12], [7;14] \} \), \( B = 17 \)

\[
a_i = [X;Y]
\]

- \( \text{value}(a_i) \)
- \( \text{size}(a_i) \)

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
<th>12</th>
<th>13</th>
<th>14</th>
<th>15</th>
<th>16</th>
<th>17</th>
</tr>
</thead>
<tbody>
<tr>
<td>{3,6,4,8,7}</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>{3,6,4,8}</td>
<td>0</td>
<td>16</td>
<td>9</td>
<td>19</td>
<td>25</td>
<td>12</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>{3,6,4}</td>
<td>0</td>
<td>-1</td>
<td>-1</td>
<td>16</td>
<td>9</td>
<td>-1</td>
<td>19</td>
<td>25</td>
<td>-1</td>
<td>35</td>
<td>28</td>
<td>-1</td>
<td>-1</td>
<td>44</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
</tr>
<tr>
<td>{3,6}</td>
<td>0</td>
<td>-1</td>
<td>-1</td>
<td>16</td>
<td>-1</td>
<td>-1</td>
<td>19</td>
<td>-1</td>
<td>-1</td>
<td>35</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
</tr>
<tr>
<td>{3}</td>
<td>0</td>
<td>-1</td>
<td>-1</td>
<td>16</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
</tr>
</tbody>
</table>

Subset = \{[[8;12]]\}
## Dynamic Programming

**Example:** \( A = \{[3;16], [6;19], [4;9], [8;12], [7;14] \} \), \( B = 17 \)

\( a_i = [X;Y] \)

- **value\((a_i)\)**
- **size\((a_i)\)**

<table>
<thead>
<tr>
<th>{3,6,4,8,7}</th>
<th>0</th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th>16</th>
<th>9</th>
<th>19</th>
<th>25</th>
<th>12</th>
<th>35</th>
</tr>
</thead>
<tbody>
<tr>
<td>{3,6,4,8}</td>
<td>0</td>
<td>16</td>
<td>9</td>
<td>19</td>
<td>25</td>
<td>-1</td>
<td>35</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>{3,6,4}</td>
<td>0</td>
<td>16</td>
<td>9</td>
<td>19</td>
<td>25</td>
<td>-1</td>
<td>35</td>
<td>28</td>
<td>-1</td>
<td>-1</td>
<td>44</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
</tr>
<tr>
<td>{3,6}</td>
<td>0</td>
<td>-1</td>
<td>-1</td>
<td>16</td>
<td>9</td>
<td>-1</td>
<td>19</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>35</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
</tr>
<tr>
<td>{3}</td>
<td>0</td>
<td>-1</td>
<td>-1</td>
<td>16</td>
<td>-1</td>
<td>-1</td>
<td>19</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>35</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
</tr>
</tbody>
</table>

Subset = \{[3;16], [6;19]\}
## Algorithms

### Dynamic Programming

**Example:** \( A = \{[3;16], [6;19], [4;9], [8;12], [7;14]\}, \ B = 17 \)

\[
\begin{align*}
    a_i & = [X;Y] \\
    \text{value}(a_i) & = \text{size}(a_i)
\end{align*}
\]

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
<th>12</th>
<th>13</th>
<th>14</th>
<th>15</th>
<th>16</th>
<th>17</th>
</tr>
</thead>
<tbody>
<tr>
<td>{3,6,4,8,7}</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>{3,6,4,8}</td>
<td>0</td>
<td>16</td>
<td>9</td>
<td>19</td>
<td>25</td>
<td>12</td>
<td>35</td>
<td>28</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>{3,6,4}</td>
<td>0</td>
<td>-1</td>
<td>-1</td>
<td>16</td>
<td>9</td>
<td>-1</td>
<td>19</td>
<td>25</td>
<td>-1</td>
<td>35</td>
<td>28</td>
<td>-1</td>
<td>-1</td>
<td>44</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
</tr>
<tr>
<td>{3,6}</td>
<td>0</td>
<td>-1</td>
<td>-1</td>
<td>16</td>
<td>-1</td>
<td>-1</td>
<td>19</td>
<td>-1</td>
<td>-1</td>
<td>35</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
</tr>
<tr>
<td>{3}</td>
<td>0</td>
<td>-1</td>
<td>-1</td>
<td>16</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
</tr>
</tbody>
</table>

**Subset = \{[6;19],[4;9]\}**
Dynamic Programming

**Example:** $A = \{[3;16], [6;19], [4;9], [8;12], [7;14]\}, B = 17$

\[
a_i = [X;Y]
\]

- **value($a_i$)**
- **size($a_i$)**

```
<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
<th>12</th>
<th>13</th>
<th>14</th>
<th>15</th>
<th>16</th>
<th>17</th>
</tr>
</thead>
<tbody>
<tr>
<td>{3,6,4,8,7}</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>{3,6,4,8}</td>
<td>0</td>
<td>16</td>
<td>9</td>
<td>19</td>
<td>25</td>
<td>12</td>
<td>35</td>
<td>28</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>{3,6,4}</td>
<td>0</td>
<td>-1</td>
<td>-1</td>
<td>16</td>
<td>9</td>
<td>-1</td>
<td>19</td>
<td>25</td>
<td>-1</td>
<td>35</td>
<td>28</td>
<td>-1</td>
<td>-1</td>
<td>44</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
</tr>
<tr>
<td>{3,6}</td>
<td>0</td>
<td>-1</td>
<td>-1</td>
<td>16</td>
<td>-1</td>
<td>-1</td>
<td>19</td>
<td>-1</td>
<td>-1</td>
<td>35</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
</tr>
<tr>
<td>{3}</td>
<td>0</td>
<td>-1</td>
<td>-1</td>
<td>16</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
</tr>
</tbody>
</table>
```

**Subset** = {[3;16], [8;12]}
Dynamic Programming

Example: $A = \{[3;16], [6;19], [4;9], [8;12], [7;14]\}$, $B = 17$

$a_i = [X;Y]$

\[
\begin{array}{cccccccccccccccccc}
0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 & 11 & 12 & 13 & 14 & 15 & 16 & 17 \\
\{3,6,4,8,7\} & 0 & & & & & & & & & & & & & & & \\
\{3,6,4,8\} & 0 & 16 & 9 & 19 & 25 & 12 & 35 & 28 & 28 & 21 & & & & & & & & \\
\{3,6,4\} & 0 & -1 & -1 & 16 & 9 & -1 & 19 & 25 & -1 & 35 & 28 & -1 & -1 & 44 & -1 & -1 & -1 & \\
\{3,6\} & 0 & -1 & -1 & 16 & -1 & -1 & 19 & -1 & -1 & 35 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & \\
\{3\} & 0 & -1 & -1 & 16 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & \\
\end{array}
\]

Subset = $\{[4;9],[8;12]\}$
Dynamic Programming

Example: \( A = \{[3;16], [6;19], [4;9], [8;12], [7;14]\}, B = 17 \)

\[ a_i = [X;Y] \]

\[ \text{value}(a_i) \]

\[ \text{size}(a_i) \]

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
<th>12</th>
<th>13</th>
<th>14</th>
<th>15</th>
<th>16</th>
<th>17</th>
</tr>
</thead>
<tbody>
<tr>
<td>{3,6,4,8,7}</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>{3,6,4,8}</td>
<td>0</td>
<td>16</td>
<td>9</td>
<td>19</td>
<td>25</td>
<td>12</td>
<td>35</td>
<td>28</td>
<td>28</td>
<td>21</td>
<td>44</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>{3,6,4}</td>
<td>0</td>
<td>-1</td>
<td>-1</td>
<td>16</td>
<td>9</td>
<td>-1</td>
<td>19</td>
<td>25</td>
<td>-1</td>
<td>35</td>
<td>28</td>
<td>-1</td>
<td>-1</td>
<td>44</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
</tr>
<tr>
<td>{3,6}</td>
<td>0</td>
<td>-1</td>
<td>-1</td>
<td>16</td>
<td>-1</td>
<td>-1</td>
<td>19</td>
<td>-1</td>
<td>-1</td>
<td>35</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
</tr>
<tr>
<td>{3}</td>
<td>0</td>
<td>-1</td>
<td>-1</td>
<td>16</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
</tr>
</tbody>
</table>

Subset = \{[3;16],[6:19],[4;9]\}
Dynamic Programming

Example: \( A = \{[3;16], [6;19], [4;9], [8;12], [7;14]\} \), \( B = 17 \)

\[ a_i = [X;Y] \]

\[ \text{value}(a_i) \]
\[ \text{size}(a_i) \]

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
<th>12</th>
<th>13</th>
<th>14</th>
<th>15</th>
<th>16</th>
<th>17</th>
</tr>
</thead>
<tbody>
<tr>
<td>{3,6,4,8,7}</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>{3,6,4,8}</td>
<td>0</td>
<td>16</td>
<td>9</td>
<td>19</td>
<td>25</td>
<td>12</td>
<td>35</td>
<td>28</td>
<td>28</td>
<td>21</td>
<td>44</td>
<td>27</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>{3,6,4}</td>
<td>0</td>
<td>-1</td>
<td>-1</td>
<td>16</td>
<td>9</td>
<td>-1</td>
<td>19</td>
<td>25</td>
<td>-1</td>
<td>35</td>
<td>28</td>
<td>-1</td>
<td>-1</td>
<td>44</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
</tr>
<tr>
<td>{3,6}</td>
<td>0</td>
<td>-1</td>
<td>-1</td>
<td>16</td>
<td>-1</td>
<td>-1</td>
<td>19</td>
<td>-1</td>
<td>-1</td>
<td>35</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
</tr>
<tr>
<td>{3}</td>
<td>0</td>
<td>-1</td>
<td>-1</td>
<td>16</td>
<td>-1</td>
<td>-1</td>
<td>19</td>
<td>-1</td>
<td>-1</td>
<td>35</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
</tr>
</tbody>
</table>

Subset = \{[6;19],[8;12]\}
## Dynamic Programming

**Example:** \( A = \{[3;16], [6;19], [4;9], [8;12], [7;14]\}, B = 17 \)

\( a_i = [X;Y] \)

- **Value** \( \text{value}(a_i) \)
- **Size** \( \text{size}(a_i) \)

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
<th>12</th>
<th>13</th>
<th>14</th>
<th>15</th>
<th>16</th>
<th>17</th>
</tr>
</thead>
<tbody>
<tr>
<td>{3,6,4,8,7}</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>{3,6,4,8}</td>
<td>0</td>
<td>16</td>
<td>9</td>
<td>19</td>
<td>25</td>
<td>12</td>
<td>35</td>
<td>28</td>
<td>28</td>
<td>21</td>
<td>44</td>
<td>27</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>{3,6,4}</td>
<td>0</td>
<td>-1</td>
<td>-1</td>
<td>16</td>
<td>9</td>
<td>-1</td>
<td>19</td>
<td>25</td>
<td>-1</td>
<td>35</td>
<td>28</td>
<td>-1</td>
<td>-1</td>
<td>44</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
</tr>
<tr>
<td>{3,6}</td>
<td>0</td>
<td>-1</td>
<td>-1</td>
<td>16</td>
<td>-1</td>
<td>-1</td>
<td>19</td>
<td>-1</td>
<td>-1</td>
<td>35</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
</tr>
<tr>
<td>{3}</td>
<td>0</td>
<td>-1</td>
<td>-1</td>
<td>16</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
</tr>
</tbody>
</table>

**Subset** = \{[3;16],[4;9],[8;12]\}
## Algorithms

### Dynamic Programming

**Example:** $A = \{[3;16], [6;19], [4;9], [8;12], [7;14]\}, B = 17$

$a_i = [X;Y]

\[ \text{value}(a_i) \]

\[ \text{size}(a_i) \]

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
<th>12</th>
<th>13</th>
<th>14</th>
<th>15</th>
<th>16</th>
<th>17</th>
</tr>
</thead>
<tbody>
<tr>
<td>{3,6,4,8,7}</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>{3,6,4,8}</td>
<td>0</td>
<td>16</td>
<td>9</td>
<td>19</td>
<td>25</td>
<td>12</td>
<td>35</td>
<td>28</td>
<td>21</td>
<td>44</td>
<td>27</td>
<td>37</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>47</td>
</tr>
<tr>
<td>{3,6,4}</td>
<td>0</td>
<td>-1</td>
<td>-1</td>
<td>16</td>
<td>9</td>
<td>-1</td>
<td>19</td>
<td>25</td>
<td>-1</td>
<td>35</td>
<td>28</td>
<td>-1</td>
<td>-1</td>
<td>44</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
</tr>
<tr>
<td>{3,6}</td>
<td>0</td>
<td>-1</td>
<td>-1</td>
<td>16</td>
<td>-1</td>
<td>-1</td>
<td>19</td>
<td>-1</td>
<td>-1</td>
<td>35</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
</tr>
<tr>
<td>{3}</td>
<td>0</td>
<td>-1</td>
<td>-1</td>
<td>16</td>
<td>-1</td>
<td>-1</td>
<td>19</td>
<td>-1</td>
<td>-1</td>
<td>35</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
</tr>
</tbody>
</table>

Subset = $\{[3;16], [6;19], [8;12]\}$
Dynamic Programming

Example: \( A = \{[3;16], [6;19], [4;9], [8;12], [7;14]\} \), \( B = 17 \)

\[ a_i = [X;Y] \]

<table>
<thead>
<tr>
<th>{3,6,4,8,7}</th>
<th>0</th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>{3,6,4,8}</td>
<td>0</td>
<td>-1</td>
<td>-1</td>
<td>16</td>
<td>9</td>
<td>-1</td>
<td>19</td>
<td>25</td>
<td>12</td>
<td>35</td>
<td>28</td>
<td>28</td>
<td>21</td>
<td>44</td>
<td>27</td>
<td>37</td>
<td>-1</td>
</tr>
<tr>
<td>{3,6,4}</td>
<td>0</td>
<td>-1</td>
<td>-1</td>
<td>16</td>
<td>9</td>
<td>-1</td>
<td>19</td>
<td>25</td>
<td>-1</td>
<td>35</td>
<td>28</td>
<td>-1</td>
<td>44</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
</tr>
<tr>
<td>{3,6}</td>
<td>0</td>
<td>-1</td>
<td>-1</td>
<td>16</td>
<td>-1</td>
<td>-1</td>
<td>19</td>
<td>-1</td>
<td>-1</td>
<td>35</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
</tr>
</tbody>
</table>
| \{3\}         | 0 | -1| -1| 16|-1 |-1 |-1 |-1 |-1 |-1 |-1 |-1 |-1 |-1 |-1 |-1 |-1 |}
Algorithms

Dynamic Programming

Example: \( A = \{[3;16], [6;19], [4;9], [8;12], [7;14]\}, \ B = 17 \)

\[ a_i = [X;Y] \]

\[ \text{value}(a_i) \]

\[ \text{size}(a_i) \]

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
<th>12</th>
<th>13</th>
<th>14</th>
<th>15</th>
<th>16</th>
<th>17</th>
</tr>
</thead>
<tbody>
<tr>
<td>{3,6,4,8,7}</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>{3,6,4,8}</td>
<td>0</td>
<td>-1</td>
<td>-1</td>
<td>16</td>
<td>9</td>
<td>-1</td>
<td>19</td>
<td>25</td>
<td>12</td>
<td>35</td>
<td>28</td>
<td>28</td>
<td>21</td>
<td>44</td>
<td>27</td>
<td>37</td>
<td>-1</td>
<td>47</td>
</tr>
<tr>
<td>{3,6,4}</td>
<td>0</td>
<td>-1</td>
<td>-1</td>
<td>16</td>
<td>9</td>
<td>-1</td>
<td>19</td>
<td>25</td>
<td>-1</td>
<td>35</td>
<td>28</td>
<td>-1</td>
<td>-1</td>
<td>44</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
</tr>
<tr>
<td>{3,6}</td>
<td>0</td>
<td>-1</td>
<td>-1</td>
<td>16</td>
<td>-1</td>
<td>-1</td>
<td>19</td>
<td>-1</td>
<td>-1</td>
<td>35</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
</tr>
<tr>
<td>{3}</td>
<td>0</td>
<td>-1</td>
<td>-1</td>
<td>16</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
</tr>
</tbody>
</table>

Subset = \{[3;16]\}
**Algorithms**

**Dynamic Programming**

**Example:** $A = \{[3;16], [6;19], [4;9], [8;12], [7;14]\}$, $B = 17$

\[ a_i = [X;Y] \]

\[ \text{value}(a_i) \]

\[ \text{size}(a_i) \]

\[
\begin{array}{cccccccccccccccc}
0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 & 11 & 12 & 13 & 14 & 15 & 16 & 17 \\
\{3,6,4,8,7\} & 0 & 16 & 9 & & & & & & & & & & & & & & \\
\{3,6,4,8\} & 0 & -1 & -1 & 16 & 9 & -1 & 19 & 25 & 12 & 35 & 28 & 28 & 21 & 44 & 27 & 37 & -1 & 47 \\
\{3,6,4\} & 0 & -1 & -1 & 16 & 9 & -1 & 19 & 25 & -1 & 35 & 28 & -1 & -1 & 44 & -1 & -1 & -1 & -1 \\
\{3,6\} & 0 & -1 & -1 & 16 & -1 & -1 & 19 & -1 & -1 & 35 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 \\
\{3\} & 0 & -1 & -1 & 16 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 \\
\end{array}
\]

Subset = \{[[4;9]]\}
**Dynamic Programming**

**Example:**  \( A = \{ [3;16], [6;19], [4;9], [8;12], [7:14] \}, B = 17 \)

\[ a_i = [X;Y] \]

\[ \text{value}(a_i) \]

\[ \text{size}(a_i) \]

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
<th>12</th>
<th>13</th>
<th>14</th>
<th>15</th>
<th>16</th>
<th>17</th>
</tr>
</thead>
<tbody>
<tr>
<td>{3,6,4,8,7}</td>
<td>0</td>
<td>16</td>
<td>9</td>
<td>19</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>{3,6,4,8}</td>
<td>0</td>
<td>-1</td>
<td>-1</td>
<td>16</td>
<td>9</td>
<td>-1</td>
<td>19</td>
<td>25</td>
<td>12</td>
<td>35</td>
<td>28</td>
<td>28</td>
<td>21</td>
<td>44</td>
<td>27</td>
<td>37</td>
<td>-1</td>
<td>47</td>
</tr>
<tr>
<td>{3,6,4}</td>
<td>0</td>
<td>-1</td>
<td>-1</td>
<td>16</td>
<td>9</td>
<td>-1</td>
<td>19</td>
<td>25</td>
<td>-1</td>
<td>35</td>
<td>28</td>
<td>-1</td>
<td>-1</td>
<td>44</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
</tr>
<tr>
<td>{3,6}</td>
<td>0</td>
<td>-1</td>
<td>-1</td>
<td>16</td>
<td>-1</td>
<td>-1</td>
<td>19</td>
<td>-1</td>
<td>-1</td>
<td>35</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
</tr>
<tr>
<td>{3}</td>
<td>0</td>
<td>-1</td>
<td>-1</td>
<td>16</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
</tr>
</tbody>
</table>

Subset = \{ [6:19] \}
## Dynamic Programming

**Example:**  \( A = \{[3; 16], [6; 19], [4; 9], [8; 12], [7; 14]\}, \ B = 17 \)

\[
a_i = \begin{bmatrix} X; Y \end{bmatrix}
\]

- **Value** \( \text{value}(a_i) \)
- **Size** \( \text{size}(a_i) \)

<table>
<thead>
<tr>
<th></th>
<th>{3,6,4,8,7}</th>
<th></th>
<th>{3,6,4,8}</th>
<th></th>
<th>{3,6,4}</th>
<th></th>
<th>{3,6}</th>
<th></th>
<th>{3}</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td></td>
<td>0</td>
<td></td>
<td>0</td>
<td></td>
<td>0</td>
<td></td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>16</td>
<td></td>
<td>16</td>
<td></td>
<td>16</td>
<td></td>
<td>16</td>
<td></td>
<td>16</td>
</tr>
<tr>
<td>2</td>
<td>9</td>
<td></td>
<td>9</td>
<td></td>
<td>9</td>
<td></td>
<td>9</td>
<td></td>
<td>9</td>
</tr>
<tr>
<td>3</td>
<td>19</td>
<td></td>
<td>19</td>
<td></td>
<td>19</td>
<td></td>
<td>19</td>
<td></td>
<td>19</td>
</tr>
<tr>
<td>4</td>
<td>25</td>
<td></td>
<td>25</td>
<td></td>
<td>25</td>
<td></td>
<td>25</td>
<td></td>
<td>25</td>
</tr>
<tr>
<td>5</td>
<td>12</td>
<td></td>
<td>12</td>
<td></td>
<td>12</td>
<td></td>
<td>12</td>
<td></td>
<td>12</td>
</tr>
<tr>
<td>6</td>
<td>35</td>
<td></td>
<td>35</td>
<td></td>
<td>35</td>
<td></td>
<td>35</td>
<td></td>
<td>35</td>
</tr>
<tr>
<td>7</td>
<td>28</td>
<td></td>
<td>28</td>
<td></td>
<td>28</td>
<td></td>
<td>28</td>
<td></td>
<td>28</td>
</tr>
<tr>
<td>8</td>
<td>44</td>
<td></td>
<td>44</td>
<td></td>
<td>44</td>
<td></td>
<td>44</td>
<td></td>
<td>44</td>
</tr>
<tr>
<td>9</td>
<td>27</td>
<td></td>
<td>27</td>
<td></td>
<td>27</td>
<td></td>
<td>27</td>
<td></td>
<td>27</td>
</tr>
<tr>
<td>10</td>
<td>37</td>
<td></td>
<td>37</td>
<td></td>
<td>37</td>
<td></td>
<td>37</td>
<td></td>
<td>37</td>
</tr>
<tr>
<td>11</td>
<td>-1</td>
<td></td>
<td>-1</td>
<td></td>
<td>-1</td>
<td></td>
<td>-1</td>
<td></td>
<td>-1</td>
</tr>
<tr>
<td>12</td>
<td>47</td>
<td></td>
<td>47</td>
<td></td>
<td>47</td>
<td></td>
<td>47</td>
<td></td>
<td>47</td>
</tr>
</tbody>
</table>

**Subset = \{[3;16],[4;9]\}**
Algorithms

Dynamic Programming

Example: \( A = \{[3;16], [6;19], [4;9], [8;12], [7;14]\}, B = 17 \)

\[ a_i = [X;Y] \]

\[ \text{value}(a_i) \]

\[ \text{size}(a_i) \]

| Subset | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 |
|--------|---|---|---|---|---|---|---|---|---|---|----|----|----|----|----|----|----|----|----|
| \{3,6,4,8,7\} | 0 | 16 | 9 | 19 | 25 | 12 |   |   |   |   |    |    |    |    |    |    |    |    |    |
| \{3,6,4,8\}   | 0 | -1 | -1 | 16 | 9 | -1 | 19 | 25 | 12 | 35 | 28 | 28 | 21 | 44 | 27 | 37 | -1 | 47 |
| \{3,6,4\}     | 0 | -1 | -1 | 16 | 9 | -1 | 19 | 25 | -1 | 35 | 28 | -1 | -1 | 44 | -1 | -1 | -1 | -1 |
| \{3,6\}       | 0 | -1 | -1 | 16 | -1 | -1 | 19 | -1 | -1 | 35 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 |
| \{3\}         | 0 | -1 | -1 | 16 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 |

Subset = \{[8;12]\}
## Algorithms

### Dynamic Programming

**Example:** $A = \{[3;16], [6;19], [4;9], [8;12], [7;14]\} , B = 17$

$a_i = [X;Y]$

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
<th>12</th>
<th>13</th>
<th>14</th>
<th>15</th>
<th>16</th>
<th>17</th>
</tr>
</thead>
<tbody>
<tr>
<td>{3,6,4,8,7}</td>
<td>0</td>
<td>16</td>
<td>9</td>
<td>19</td>
<td>25</td>
<td>12</td>
<td>35</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>{3,6,4,8}</td>
<td>0</td>
<td>-1</td>
<td>-1</td>
<td>16</td>
<td>9</td>
<td>-1</td>
<td>19</td>
<td>25</td>
<td>12</td>
<td>35</td>
<td>28</td>
<td>28</td>
<td>21</td>
<td>44</td>
<td>27</td>
<td>37</td>
<td>-1</td>
<td>47</td>
</tr>
<tr>
<td>{3,6,4}</td>
<td>0</td>
<td>-1</td>
<td>-1</td>
<td>16</td>
<td>9</td>
<td>-1</td>
<td>19</td>
<td>25</td>
<td>-1</td>
<td>35</td>
<td>28</td>
<td>-1</td>
<td>-1</td>
<td>44</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
</tr>
<tr>
<td>{3,6}</td>
<td>0</td>
<td>-1</td>
<td>-1</td>
<td>16</td>
<td>-1</td>
<td>-1</td>
<td>19</td>
<td>-1</td>
<td>-1</td>
<td>35</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
</tr>
<tr>
<td>{3}</td>
<td>0</td>
<td>-1</td>
<td>-1</td>
<td>16</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
</tr>
</tbody>
</table>

Subset = {\{3;16],[6;19]\}
**Algorithms**

**Dynamic Programming**

**Example:** $A = \{[3;16], [6;19], [4;9], [8;12], [7;14]\}$, $B = 17$

\[ a_i = [X;Y] \]

- **value**($a_i$)
- **size**($a_i$)

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
<th>12</th>
<th>13</th>
<th>14</th>
<th>15</th>
<th>16</th>
<th>17</th>
</tr>
</thead>
<tbody>
<tr>
<td>{3,6,4,8,7}</td>
<td>0</td>
<td>16</td>
<td>9</td>
<td>19</td>
<td>25</td>
<td>12</td>
<td>35</td>
<td>30</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>{3,6,4,8}</td>
<td>0</td>
<td>-1</td>
<td>-1</td>
<td>16</td>
<td>9</td>
<td>-1</td>
<td>19</td>
<td>25</td>
<td>12</td>
<td>35</td>
<td>28</td>
<td>28</td>
<td>21</td>
<td>44</td>
<td>27</td>
<td>37</td>
<td>-1</td>
<td>47</td>
</tr>
<tr>
<td>{3,6,4}</td>
<td>0</td>
<td>-1</td>
<td>-1</td>
<td>16</td>
<td>9</td>
<td>-1</td>
<td>19</td>
<td>25</td>
<td>-1</td>
<td>35</td>
<td>28</td>
<td>-1</td>
<td>-1</td>
<td>44</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
</tr>
<tr>
<td>{3,6}</td>
<td>0</td>
<td>-1</td>
<td>-1</td>
<td>16</td>
<td>-1</td>
<td>-1</td>
<td>19</td>
<td>-1</td>
<td>-1</td>
<td>35</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
</tr>
<tr>
<td>{3}</td>
<td>0</td>
<td>-1</td>
<td>-1</td>
<td>16</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
</tr>
</tbody>
</table>

Subset = \{[3;16],[7;14]\}
Algorithms

Dynamic Programming

Example: $A = \{[3;16], [6;19], [4;9], [8;12], [7;14]\}, \ B = 17$

$$a_i = [X;Y]$$

<table>
<thead>
<tr>
<th>Value($a_i$)</th>
<th>Size($a_i$)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>-1</td>
</tr>
<tr>
<td>16</td>
<td>9</td>
</tr>
<tr>
<td>19</td>
<td>25</td>
</tr>
<tr>
<td>12</td>
<td>35</td>
</tr>
<tr>
<td>30</td>
<td>28</td>
</tr>
</tbody>
</table>

Subset = $\{[3;16],[8;12]\}$
### Algorithms

#### Dynamic Programming

**Example:** \( A = \{[3;16], [6;19], [4;9], [8;12], [7;14]\} \), \( B = 17 \)

\[ a_i = [X;Y] \]

\[ \text{value}(a_i) \]

\[ \text{size}(a_i) \]

|     | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 |
|-----|---|---|---|---|---|---|---|---|---|---|----|----|----|----|----|----|----|----|----|
| \{3,6,4,8,7\} | 0 | 16 | 9 | 19 | 25 | 12 | 35 | 30 | 28 | 21 |     |     |     |     |     |     |     |     |
| \{3,6,4,8\}    | 0 | -1 | -1 | 16 | 9 | -1 | 19 | 25 | 12 | 35 | 28 | 21 | 44 | 27 | 37 | -1 | 47 |
| \{3,6,4\}      | 0 | -1 | -1 | 16 | 9 | -1 | 19 | 25 | -1 | 35 | 28 | -1 | -1 | 44 | -1 | -1 | -1 | -1 |
| \{3,6\}        | 0 | -1 | -1 | 16 | -1 | -1 | 19 | -1 | -1 | 35 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 |
| \{3\}          | 0 | -1 | -1 | 16 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 |

**Subset = \{[4;9],[8;12]\}**
## Algorithms

### Dynamic Programming

**Example:** \( A = \{ [3;16], [6;19], [4;9], [8;12], [7;14] \}, \ B = 17 \)

\[ a_i = [X;Y] \]

\[ \text{value}(a_i) \]

\[ \text{size}(a_i) \]

<table>
<thead>
<tr>
<th>{3,6,4,8,7}</th>
<th>0</th>
<th>16</th>
<th>9</th>
<th>19</th>
<th>25</th>
<th>12</th>
<th>35</th>
<th>30</th>
<th>28</th>
<th>21</th>
<th>44</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>{3,6,4,8}</td>
<td>0</td>
<td>-1</td>
<td>-1</td>
<td>16</td>
<td>9</td>
<td>-1</td>
<td>19</td>
<td>25</td>
<td>12</td>
<td>35</td>
<td>28</td>
<td>28</td>
<td>21</td>
</tr>
<tr>
<td>{3,6,4}</td>
<td>0</td>
<td>-1</td>
<td>-1</td>
<td>16</td>
<td>9</td>
<td>-1</td>
<td>19</td>
<td>25</td>
<td>-1</td>
<td>35</td>
<td>28</td>
<td>-1</td>
<td>-1</td>
</tr>
<tr>
<td>{3,6}</td>
<td>0</td>
<td>-1</td>
<td>-1</td>
<td>16</td>
<td>-1</td>
<td>-1</td>
<td>19</td>
<td>-1</td>
<td>-1</td>
<td>35</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
</tr>
<tr>
<td>{3}</td>
<td>0</td>
<td>-1</td>
<td>-1</td>
<td>16</td>
<td>-1</td>
<td>-1</td>
<td>19</td>
<td>-1</td>
<td>-1</td>
<td>35</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
</tr>
</tbody>
</table>

Subset = {[3;16],[6;19],[4;9]}
Example: $A = \{[3;16], [6;19], [4;9], [8;12], [7;14]\}, \ B = 17$

\[
\begin{align*}
a_i &= [X;Y] \\
value(a_i) \\
size(a_i)
\end{align*}
\]

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
<th>12</th>
<th>13</th>
<th>14</th>
<th>15</th>
<th>16</th>
<th>17</th>
</tr>
</thead>
<tbody>
<tr>
<td>{3,6,4,8,7}</td>
<td>0</td>
<td>16</td>
<td>9</td>
<td>19</td>
<td>25</td>
<td>12</td>
<td>35</td>
<td>30</td>
<td>28</td>
<td>21</td>
<td>44</td>
<td>39</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>{3,6,4,8}</td>
<td>0</td>
<td>-1</td>
<td>-1</td>
<td>16</td>
<td>9</td>
<td>-1</td>
<td>19</td>
<td>25</td>
<td>12</td>
<td>35</td>
<td>28</td>
<td>28</td>
<td>21</td>
<td>44</td>
<td>27</td>
<td>37</td>
<td>-1</td>
<td>47</td>
</tr>
<tr>
<td>{3,6,4}</td>
<td>0</td>
<td>-1</td>
<td>-1</td>
<td>16</td>
<td>9</td>
<td>-1</td>
<td>19</td>
<td>25</td>
<td>-1</td>
<td>35</td>
<td>28</td>
<td>-1</td>
<td>-1</td>
<td>44</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
</tr>
<tr>
<td>{3,6}</td>
<td>0</td>
<td>-1</td>
<td>-1</td>
<td>16</td>
<td>-1</td>
<td>-1</td>
<td>19</td>
<td>-1</td>
<td>-1</td>
<td>35</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
</tr>
<tr>
<td>{3}</td>
<td>0</td>
<td>-1</td>
<td>-1</td>
<td>16</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
</tr>
</tbody>
</table>

Subset = \{[3;16],[4;9],[7;14]\}
Dynamic Programming

Example: \( A = \{ [3;16], [6;19], [4;9], [8;12], [7;14] \} \), \( B = 17 \)

\[ a_i = [X;Y] \]

\[ \text{value}(a_i) \]

\[ \text{size}(a_i) \]

\[
\begin{array}{cccccccccccccccc}
0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 & 11 & 12 & 13 & 14 & 15 & 16 & 17 \\
\{3,6,4,8,7\} & 0 & 16 & 9 & 19 & 25 & 12 & 35 & 30 & 28 & 21 & 44 & 39 & 37 & & & & \\
\{3,6,4\} & 0 & -1 & -1 & 16 & 9 & -1 & 19 & 25 & -1 & 35 & 28 & -1 & -1 & 44 & -1 & -1 & -1 & -1 \\
\{3,6\} & 0 & -1 & -1 & 16 & -1 & -1 & 19 & -1 & -1 & 35 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 \\
\{3\} & 0 & -1 & -1 & 16 & -1 & -1 & -1 & 19 & 19 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 \\
\end{array}
\]

Subset = \{[3;16],[4;9],[8;12]\}
## Algorithms

### Dynamic Programming

**Example:** \( A = \{[3;16], [6;19], [4;9], [8;12], [7;14]\}, B = 17 \)

\[ a_i = [X;Y] \]

\[ \text{value}(a_i) \]

\[ \text{size}(a_i) \]

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
<th>12</th>
<th>13</th>
<th>14</th>
<th>15</th>
<th>16</th>
<th>17</th>
</tr>
</thead>
<tbody>
<tr>
<td>{3,4,6,8,7}</td>
<td>0</td>
<td>16</td>
<td>9</td>
<td>19</td>
<td>25</td>
<td>12</td>
<td>35</td>
<td>30</td>
<td>28</td>
<td>21</td>
<td>44</td>
<td>39</td>
<td>37</td>
<td>49</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>{3,6,4,8}</td>
<td>0</td>
<td>-1</td>
<td>-1</td>
<td>16</td>
<td>9</td>
<td>-1</td>
<td>19</td>
<td>25</td>
<td>12</td>
<td>35</td>
<td>-1</td>
<td>28</td>
<td>28</td>
<td>21</td>
<td>44</td>
<td>27</td>
<td>37</td>
<td>-1</td>
</tr>
<tr>
<td>{3,6,4}</td>
<td>0</td>
<td>-1</td>
<td>-1</td>
<td>16</td>
<td>9</td>
<td>-1</td>
<td>19</td>
<td>25</td>
<td>-1</td>
<td>35</td>
<td>28</td>
<td>-1</td>
<td>-1</td>
<td>44</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
</tr>
<tr>
<td>{3,6}</td>
<td>0</td>
<td>-1</td>
<td>-1</td>
<td>16</td>
<td>-1</td>
<td>-1</td>
<td>19</td>
<td>-1</td>
<td>-1</td>
<td>35</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
</tr>
<tr>
<td>{3}</td>
<td>0</td>
<td>-1</td>
<td>-1</td>
<td>16</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
</tr>
</tbody>
</table>

 Subset = \{[3;16],[6;19],[7;14]\}
# Algorithms

## Dynamic Programming

**Example:** $A = \{[3;16], [6;19], [4;9], [8;12], [7;14]\}$, $B = 17$

\[
a_i = [X;Y]
\]

- **value**($a_i$)
- **size**($a_i$)

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
<th>12</th>
<th>13</th>
<th>14</th>
<th>15</th>
<th>16</th>
<th>17</th>
</tr>
</thead>
<tbody>
<tr>
<td>{3,6,4,8,7}</td>
<td>0</td>
<td>16</td>
<td>9</td>
<td>19</td>
<td>25</td>
<td>12</td>
<td>35</td>
<td>30</td>
<td>28</td>
<td>21</td>
<td>44</td>
<td>39</td>
<td>37</td>
<td>49</td>
<td>47</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>{3,6,4,8}</td>
<td>0</td>
<td>-1</td>
<td>-1</td>
<td>16</td>
<td>9</td>
<td>-1</td>
<td>19</td>
<td>25</td>
<td>12</td>
<td>35</td>
<td>28</td>
<td>21</td>
<td>44</td>
<td>27</td>
<td>37</td>
<td>-1</td>
<td>47</td>
<td></td>
</tr>
<tr>
<td>{3,6,4}</td>
<td>0</td>
<td>-1</td>
<td>-1</td>
<td>16</td>
<td>9</td>
<td>-1</td>
<td>19</td>
<td>25</td>
<td>-1</td>
<td>35</td>
<td>28</td>
<td>-1</td>
<td>-1</td>
<td>44</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
</tr>
<tr>
<td>{3,6}</td>
<td>0</td>
<td>-1</td>
<td>-1</td>
<td>16</td>
<td>-1</td>
<td>-1</td>
<td>19</td>
<td>-1</td>
<td>-1</td>
<td>35</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
</tr>
<tr>
<td>{3}</td>
<td>0</td>
<td>-1</td>
<td>-1</td>
<td>16</td>
<td>-1</td>
<td>-1</td>
<td>19</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
</tr>
</tbody>
</table>

Subset = \{[3;16],[6;19],[8;12]\}
## Algorithms

### Dynamic Programming

**Example:** \( A = \{ [3;16], [6;19], [4;9], [8;12], [7;14] \} \), \( B = 17 \)

\[ a_i = [X;Y] \]

- value \( (a_i) \)
- size \( (a_i) \)

<table>
<thead>
<tr>
<th>( )</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
<th>12</th>
<th>13</th>
<th>14</th>
<th>15</th>
<th>16</th>
<th>17</th>
</tr>
</thead>
<tbody>
<tr>
<td>{3,6,4,8,7}</td>
<td>0</td>
<td>16</td>
<td>9</td>
<td>19</td>
<td>25</td>
<td>12</td>
<td>35</td>
<td>30</td>
<td>28</td>
<td>21</td>
<td>44</td>
<td>39</td>
<td>37</td>
<td>49</td>
<td>47</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>{3,6,4,8}</td>
<td>0</td>
<td>-1</td>
<td>-1</td>
<td>16</td>
<td>9</td>
<td>-1</td>
<td>19</td>
<td>25</td>
<td>12</td>
<td>35</td>
<td>28</td>
<td>28</td>
<td>21</td>
<td>44</td>
<td>27</td>
<td>37</td>
<td>-1</td>
<td>47</td>
</tr>
<tr>
<td>{3,6,4}</td>
<td>0</td>
<td>-1</td>
<td>-1</td>
<td>16</td>
<td>9</td>
<td>-1</td>
<td>19</td>
<td>25</td>
<td>-1</td>
<td>35</td>
<td>28</td>
<td>-1</td>
<td>-1</td>
<td>44</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td></td>
</tr>
<tr>
<td>{3,6}</td>
<td>0</td>
<td>-1</td>
<td>-1</td>
<td>16</td>
<td>-1</td>
<td>-1</td>
<td>19</td>
<td>-1</td>
<td>-1</td>
<td>35</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td></td>
</tr>
<tr>
<td>{3}</td>
<td>0</td>
<td>-1</td>
<td>-1</td>
<td>16</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td></td>
</tr>
</tbody>
</table>

Subset = \{ [3;16],[6;19],[7;14] \}
Algorithms

Greedy Method

Description: Given a set $E$ of objects,
a function $c : E \times E \times E \ldots \rightarrow \{\text{true}, \text{false}\}$
a function $v : E \rightarrow \mathbb{N}^+$
such that

\[
\forall E_1 \subseteq E \ c(E_1) == \text{true} \rightarrow c(E_2 \subset E_1) == \text{true}
\]
\[
\forall E_1 \subseteq E, E_2 \subseteq E \ \text{s.t.} \ c(E_1) == \text{true}, c(E_2) == \text{true}, |E_2| = |E_1| + 1
\]
\[
\exists e \in E_2 \ \text{where} \ c(E_1 \cup \{e\}) == \text{true}
\]

Find a maximum cardinality subset $E^* \subseteq E$ such that

$c(E^*) == \text{true}$ and

$\forall E' \subseteq E \ c(E') == \text{true} \rightarrow \sum_{e \in E^*} v(e) \geq \sum_{e \in E'} v(e)$
Algorithms

Greedy Method

Problem: Given a set $E = \{a_1, a_2, ...a_n\}$ of $n$ objects, positive integer weights $v(a_1), v(a_2), ..., v(a_n)$ and a constraint function $c : E \rightarrow \{true, false\}$

Find a maximum cardinality subset $E^* \subseteq E$ such that

$c(E^*) == true$ and

$\forall E' \subseteq E, \ c(E') == true \rightarrow \sum_{a \in E'} v(a) \le \sum_{a \in E^*} v(a)$. 

Possible Solution:

$T \leftarrow \emptyset$

Repeat the following until $E = \emptyset$

Let $a$ be an element of $E$ such that $v(a)$ is minimum

If $c(T \cup \{a\}) = true$ then $T \leftarrow T \cup \{a\}$

$E \leftarrow E \setminus \{a\}$

Output $T$
Algorithms

Greedy Method

Example: Minimum Spanning Tree

Instance: A graph $G = (V, E)$ and a function $v : E \to \mathbb{N}^+$

Find: A subset $E' \subseteq E$ such that every vertex in $V$ is an endpoint of at least one edge in $E'$, $(V, E')$ is a connected tree and, for any other $E'' \subseteq E$ such that every vertex in $V$ is an endpoint of at least one edge in $E'$ and $(V, E'')$ is a connected tree, $\sum_{e \in E'} v(e) \leq \sum_{e \in E''} v(e)$

Objects: edges in $E$

Weights: $v$

Constraint Function: $c(E' \subseteq E) == true$ iff $E'$ is a forest

A forest is a collection of edges with no cycles
Algorithms

Greedy Method

Example: Minimum Spanning Tree

\[ T \leftarrow \emptyset \]

Repeat the following until \( E = \emptyset \)

Let \( e \) be an element of \( E \) such that \( v(e) \) is minimum

If \( T \cup \{e\} \) is a forest then \( T \leftarrow T \cup \{e\} \)

\( E \leftarrow E \setminus \{e\} \)

Output \( T \)

How to check if \( T \cup \{e\} \) is a forest:

- Attach group numbers \( g(w) \) to each \( w \in V \) with the meaning:
  \[ g(w_1) \equiv g(w_2) \text{ iff there is a path from } w_1 \text{ to } w_2 \text{ in } T \]

Initialize: \( g(w) \) is different for all \( w \)

Check: let \( e = \langle w_1, w_2 \rangle \); \( T \cup \{e\} \) is a forest iff \( g(w_1) \equiv g(w_2) \)

Update: for all \( w \) such that \( g(w) \equiv g(w_2) \), set \( g(w) = g(w_1) \)
Algorithms

Greedy Method

Example: Minimum Spanning Tree

\[ T \leftarrow \emptyset \]

Repeat the following until \( E = \emptyset \)

Let \( e \) be an element of \( E \) such that \( v(e) \) is minimum

If \( T \cup \{e\} \) is a forest then \( T \leftarrow T \cup \{e\} \)

\( E \leftarrow E \setminus \{e\} \)

Output \( T \)

How to check if \( T \cup \{e\} \) is a forest:

- Use an inverted tree – one \( w \) represents all vertices connected in \( T \), there is a (short) path from any \( w \) to its representative

Initialize: all \( w \) are representatives of themselves: \( rep(w) = w \)

Check: let \( e = \langle w_1, w_2 \rangle \); \( T \cup \{e\} \) is a forest iff \( rep(w_1) \neq rep(w_2) \)

Update: join tree rooted at \( w_2 \) to tree rooted at \( w_1 \) - new tree has \( w_1 \) as root and contains all vertices from both trees
Algorithms

Greedy Method

Example: Minimum Spanning Tree

Let $w$ be one of the endpoints of the lowest weight edge $e_L$ in $E$

$S \leftarrow \{w\}$

$T \leftarrow \emptyset$

Repeat the following until all vertices are in $S$

Let $e$ be the lowest weight edge of $E$ connecting a vertex not in $S$ with a vertex in $S$ and let $w$ be the endpoint of $e$ that is not in $S$

$S \leftarrow S \cup \{w\}$

$T \leftarrow T \cup \{e\}$

Output $T$

Invariant:

After every iteration of the loop, $T$ contains a minimum weight tree that spans all vertices in $S$
Algorithms

Greedy Method

How to find the minimum weight edge connecting a vertex not in $S$ with a vertex in $S$:

Construct a table $R$: rows are vertex indices, $R[i,1]$ is the vertex in $S$ that is 'closest' to vertex $w_i$ and $R[i,2]$ is the weight of the edge that connects $w_i$ with $R[i,1]$

Initialize: for all rows $i$, $R[i,1]$ is not important, $R[i,2] = \infty$

Check: find $i$ for which $R[i,2]$ is minimum – that identifies the vertex to add to $S$

Update: when a vertex $w_i$ is added to $S$, for every edge $e$ that connects $w_i$ to an endpoint $w_j$ that is not in $S$, if $s(e) < R[i,2]$ then set $R[i,2] = s(e)$ and set $R[i,1] = j$
Algorithms

Greedy Method

\[
T = \begin{array}{cccc}
{} & 1 & 2 & 3 \\
\{} & & & \\
\end{array}
\]

\[
R = \begin{array}{ccc}
1 & - & - \\
2 & 1 & 8 \\
3 & 1 & 16 \\
4 & 1 & 19 \\
5 & 1 & 6 \\
\end{array}
\]
Add \{1,5\} to T

Green edges are MST over vertices 1,5
Add \{1,5\} to T

Is 2 closer to S? Check \(w(\{2,5\}) = 18\) vs. 8
Algorithms

Greedy Method

Add \{1,5\} to T
Is 2 closer to S? Check $w(\{2,5\}) = 18$ vs. 8
Is 3 closer to S? Check $w(\{3,5\}) = 12$ vs. 16
Add \{1,5\} to T
Is 2 closer to S? Check \(w(\{2,5\}) = 18\) vs. 8
Is 3 closer to S? Check \(w(\{3,5\}) = 12\) vs. 16
Greedy Method

Add \{1,5\} to \(T\)

Is 2 closer to \(S\)? Check \(w(\{2,5\}) = 18\) vs. 8
Is 3 closer to \(S\)? Check \(w(\{3,5\}) = 12\) vs. 16
Is 4 closer to \(S\)? Check \(w(\{4,5\}) = 9\) vs. 19
Greedy Method

Add \(\{1,5\}\) to \(T\)

Is 2 closer to S? Check \(w(\{2,5\}) = 18\) vs. 8

Is 3 closer to S? Check \(w(\{3,5\}) = 12\) vs. 16

Is 4 closer to S? Check \(w(\{4,5\}) = 9\) vs. 19
Add \{1,2\} to \(T\)

Green edges are MST over vertices 1,2,5
Greedy Method

Algorithms

Add \{1,2\} to T
Is 3 closer to S? Check \(w(\{3,2\}) = 7\) vs. 12
Add \{1,2\} to \textbf{T}.

Is 3 closer to S? Check \(w(\{3,2\}) = 7\) vs. 12.
Greedy Method

Add \{1,2\} to \mathbf{T}

Is 3 closer to S? Check \(w(\{3,2\}) = 7\) vs. 12

Is 3 closer to S? Check \(w(\{4,2\}) = 10\) vs. 9
Add \{2,3\} to T

Green edges are MST over vertices 1,2,3,5
Add \{4,5\} to T

Green edges are MST over vertices 1,2,3,4,5
Algorithms

Greedy Method

Example: Integer Deadline Scheduling

Instance: A set $E = \{a_1, a_2, \ldots, a_n\}$ of $n$ jobs each requiring unit time to complete, a deadline function $d : E \rightarrow N^+$, and a profit function $p : E \rightarrow N^+$

Find: A schedule $\{a_{\pi_1}, a_{\pi_2}, \ldots, a_{\pi_n}\}$ for a single processor such that

$$\sum_{a \in \{a_i : \pi_i^{-1} \leq d(a_i)\}} p(a) \text{ is maximum over all schedules}$$

Objects: elements of $E$

Weights: $p$

Constraint Function: $c(E' \subseteq E) == true$ iff there is a schedule where all $a \in E'$ can be completed before their deadlines. Such a schedule is said to be at full profit
**Algorithms**

**Greedy Method**

**Example:** Integer Deadline Scheduling

\[ T \leftarrow \emptyset \]

Repeat the following until \( E = \emptyset \)

Let \( a \) be an element of \( E \) such that \( p(a) \) is maximum

If \( T \cup \{a\} \) is at full profit then \( T \leftarrow T \cup \{a\} \)

\[ E \leftarrow E \setminus \{a\} \]

Output \( T \)

How to check if \( T \cup \{a\} \) is at full profit:

> Let \( T = [a_{\pi_1}, a_{\pi_2}, \ldots, a_{\pi_k}] \) such that for \( 1 \leq i \leq k, \ i \leq d(a_{\pi_i}) \).

Assume all \( a_{\pi_i} \) are in increasing order in \( T \). Temporarily place \( a \) in \( T \) maintaining order.

**Initialize:** \( T \leftarrow \emptyset \)

**Check:** is at full profit iff for \( 1 \leq i \leq k+1, \ i \leq d(a_{\pi_i}) \) in tmp \( T \)

**Update:** make the temporary \( T \) permanent
Algorithms

Greedy Method

Example: Integer Deadline Scheduling

\[ T \leftarrow \emptyset \]

Repeat the following until \( E = \emptyset \)

- Let \( a \) be an element of \( E \) such that \( p(a) \) is maximum
- If \( T \cup \{a\} \) is at full profit then \( T \leftarrow T \cup \{a\} \)
- \( E \leftarrow E \setminus \{a\} \)

Output \( T \)

How to check if \( T \cup \{a\} \) is at full profit:

- Let \( T = [a_{\pi_1}, a_{\pi_2}, \ldots, a_{\pi_k}] \) such that for \( 1 \leq i \leq k, \ i \leq d(a_{\pi_i}) \).

\( T \) has integer positions, some of which are NULL. Attempt to place \( a \) in the highest NULL position of \( T \) before or at position \( d(a) \).

Initialize: \( T \leftarrow \emptyset \)

Check: is at full profit iff \( a \) can be placed

Update: make the temporary \( T \) permanent
Greedy Method

Example: Integer Deadline Scheduling

\[ T \leftarrow \emptyset \]  

Repeat the following until \( E = \emptyset \)

Let \( a \) be an element of \( E \) such that \( p(a) \) is maximum

If \( T \cup \{a\} \) is at full profit then \( T \leftarrow T \cup \{a\} \)

\( E \leftarrow E \setminus \{a\} \)

Output \( T \)

How to check if \( T \cup \{a\} \) is at full profit:

- Let \( T = [a_{\pi_1}, a_{\pi_2}, \ldots, a_{\pi_k}] \) such that for \( 1 \leq i \leq k \), \( i \leq d(a_{\pi_i}) \).

Same as previous except a supporting structure is used to learn occupied positions to speed up the search for a NULL.

Initialize: \( T \leftarrow \emptyset \)

Check: is at full profit iff \( a \) can be placed

Update: make the temporary \( T \) permanent
Algorithms

Backtrack

**Description:** Let $A = \{a_1, a_2, \ldots, a_n\}$ be a set of $n$ variables. Let $V = \{v_1, v_2, \ldots, v_r\}$ be a set of $r$ values. Each variable may be given a value from $V$ or it may be unassigned in which case it has value $\perp$. An *assignment* is an $n$–tuple of values from $V$ which has been given to the $n$ variables. Let $V' = V \cup \{\perp\}$. A *partial assignment* is an $n$–tuple of values from $V'$ which has been given to the variables. If $D'$ and $D''$ are (partial) assignments such that every component of $D'$ which is not equal to $\perp$ matches its counterpart in $D''$ then we say $D' \triangleleft D''$ or $D'$ *precedes* $D''$.

**Problem:** Given sets $A$ and $V$ above and a constraint function $c : V' \times V' \times \ldots \times V' \rightarrow \{true, false\}$ which maps (partial) assignments to true or false and has the (precedence) property that for any partial assignments $D'$ and $D''$, if $D' \triangleleft D''$ then if $c(D') == false$ then $c(D'') == false$. Find an assignment $D$ such that $c(D) == true$. 
Algorithms

Backtrack

Algorithm Template:

\[ D = \langle \bot, \bot, \ldots, \bot \rangle \]

\[ i \leftarrow 0 \]

Backtrack \((i)\):

If \(c(D) == \text{false} \) then return

Otherwise, if \(i == n\), Output \(D\)

Otherwise,

for \(j = 1\) to \(r\) do the following:

\[ d_{i+1} \leftarrow v_j \]

Backtrack \((i+1)\)

end

\[ d_{i+1} = \bot \]
Algorithms

Backtrack

Example: n-Queens

Given: An $n \times n$ chessboard and $n$ queens

Find: a placement of queens on the board so that no two are attacking each other

$c_Q(D)$:

For $k \leftarrow i-1$ downto 1 do the following:

If $d_i - (k-i) \leq n$ and $d_k = d_i - (k-i)$ then Output $false$

Otherwise, if $d_i + (k-i) \geq 0$ and $d_k = d_i + (k-i)$ then Output $false$

Otherwise, if $d_k == d_i$ then Output $false$

Output $true$
Example: Satisfiability

**Given:** A CNF formula $\varphi$

**Find:** Whether there exists an assignment of values to the variables of $\varphi$ that causes $\varphi$ to evaluate to $1$.

$c_{Sat}(D)$:

For each clause $C \in \varphi$ do the following:

- If all literals in $C$ have value $0$ under $D$ then Output $false$.
- Output $true$.
Algorithms

Backtrack

Example: Sudoku

Given: A $9 \times 9$ Sudoku board – 0 in a position means free, a number fixes that position, each number $\in \{1...9\}$

Find: The numbers in positions such that all numbers in any $3 \times 3$ subsquare are distinct, numbers in all rows, all columns, and two diagonals are distinct.

$\text{distinct}(D)$:
For $i \leftarrow 1$ to 8 do the following:
   For $j \leftarrow i+1$ to 9 do the following: If $d_i == d_j$ Output false
Output true

$\text{c}_{\text{Sudoku}}(D)$:
For each $3 \times 3$ subsquare $i$, if $\neg \text{distinct}(D_i)$ Output false
For each row $j$, if $\neg \text{distinct}(D_j)$ Output false
For each column $k$, if $\neg \text{distinct}(D_k)$ Output false
For each diagonal $l$, if $\neg \text{distinct}(D_l)$ Output false
Output true
Algorithms

Branch and Bound

Description: Let \( A = \{a_1, a_2, \ldots, a_n\} \) be a set of \( n \) variables. Let \( V = \{v_1, v_2, \ldots, v_r\} \) be a set of \( r \) values. Each variable may be given a value from \( V \) or it may be unassigned in which case it has value \( \bot \). An assignment is an \( n \)-tuple of values from \( V \) which has been given to the \( n \) variables. Let \( V' = V \cup \{\bot\} \). A partial assignment is an \( n \)-tuple of values from \( V' \) which has been given to the variables. If \( D' \) and \( D'' \) are (partial) assignments such that every component of \( D' \) which is not equal to \( \bot \) matches its counterpart in \( D'' \) then we say \( D' \prec D'' \) or \( D' \) precedes \( D'' \).

Problem: Given sets \( A \) and \( V \) above, a constraint function \( c: V' \times V' \times \ldots \times V' \rightarrow \{true, false\} \), and a function \( w: V' \times V' \times \ldots \times V' \rightarrow \mathbb{N}^+ \) which maps partial assignments to positive integers. Find an assignment \( D^* \) such that \( c(D^*) = true \), and \( w(D^*) \geq w(D) \) for all other complete assignments \( D \).
Algorithms

Branch and Bound

Conventions:

PAL – list of partial assignments to be explored in search

\( h: V' \times V' \times ... \times V' \rightarrow N^+ \) - function that supplies a relative estimate of the value of an optimal complete solution which preceeds the particular partial assignment that is the argument of \( h \). \( h \) will be an underestimate always. That is, \( h(D') \leq w(D) \) if \( D' \triangleleft D \) and \( D \) is complete, and \( h(D) = w(D) \) if \( D \) is complete. Moreover, for partial assignment \( D' \), \( h(D') \) will be no greater than \( w(D) \) for any complete assignment \( D \) that preceeds \( D' \)

\( U \) – a variable that will contain an upper bound on the optimal optimal solution. \( U \) is initialized to \( \infty \) and is reduced periodically during execution of the BB algorithm
Algorithms

Branch and Bound
Algorithm Template:

\( \text{BB}(w,c) : \)

\[ U \leftarrow \infty; \quad D \leftarrow \langle \bot, \bot, \ldots, \bot \rangle \]

Put \( D \) into \( \text{PAL} \)

Repeat the following until \( \text{PAL} \) is empty:

Choose and remove \( D \) (such that \( h(D) \) is minimum) from \( \text{PAL} \)
Choose \( x \), an unassigned variable in \( D \)
For each value \( v \in V \) do the following:

Extend \( D \) to \( D' \) with value \( v \) assigned to \( x \)
If \( h(D') \) < \( U \) then do the following:

If \( D' \) is not complete, put \( D' \) into \( \text{PAL} \)
If \( D' \) is complete and \( w(D') \) < \( U \) then do this:

\[ U \leftarrow w(D'); \quad M \leftarrow D'; \]

Remove from \( \text{PAL} \) any \( D \) s.t. \( U \leq h(D) \)

Output \( M \)
Branch and Bound

Example: Traveling Salesman Problem

Visit every city exactly once
Return to starting city
Minimum cost tour
Algorithms

Branch and Bound

Example: Traveling Salesman Problem – choose estimator $h$

Observe: Cost of any tour is half of the sum of costs of two tour edges connected to each vertex.
Algorithms

Branch and Bound

Example: Traveling Salesman Problem – choose estimator $h$

Observe: Cost of any tour is half of the sum of costs of two tour edges connected to each vertex.

Sum of costs of two tour edges connected to a vertex is at least the sum of the two lowest cost edges connected to the vertex.
Algorithms

Branch and Bound

Example: Traveling Salesman Problem – choose estimator $h$

Observe: Cost of any tour is half of the sum of costs of two tour edges connected to each vertex.

Sum of costs of two tour edges connected to a vertex is at least the sum of the two lowest cost edges connected to the vertex.

Cost of any tour is at least half of the sum of costs of the two lowest cost edges connected to each vertex.
Algorithms

Branch and Bound

Example: Traveling Salesman Problem – choose estimator $h$

**Observe**: Cost of any tour is half of the sum of costs of two tour edges connected to each vertex.

Sum of costs of two tour edges connected to a vertex is at least the sum of the two lowest cost edges connected to the vertex.

Cost of any tour is at least half of the sum of costs of the two lowest cost edges connected to each vertex.

$h$: include cost of any edge in partial tour

Plus, for all vertices having one edge in the Partial tour, add the cost of cheapest edge, Plus for all other, add two lowest cost edges
**Algorithms**

**Branch and Bound**

Example: Traveling Salesman Problem

S is the solution set of edges – we seek to build S incrementally by extending a current partial tour, starting from $\emptyset$.

Give every edge a unique integer identity. $D = \langle d_1, d_2, ..., d_m \rangle$ is a 0-1 vector: $d_i = 0(1)$ means edge with identity i is not (is) in S. Initially $D \leftarrow \langle \bot, \bot, ..., \bot \rangle$

Yellow nodes represent assignments in the PAL
Algorithms

Branch and Bound

Example: Traveling Salesman Problem

{1,4} not in S

{1,4} in S

Expand the only yellow node
Branch and Bound

Example: Traveling Salesman Problem

Light blue line shows what extension to the current partial tour we are exploring
Branch and Bound

Example: Traveling Salesman Problem

$h = \frac{(14+15+18+19+15)}{2} = 40.5$
Algorithms

Branch and Bound

Example: Traveling Salesman Problem

Yellow nodes represent assignments in the PAL
Algorithms

Branch and Bound

Example: Traveling Salesman Problem

\{1,4\} not in S

\{1,4\} in S

\( h = 40.5 \)
Algorithms

Branch and Bound

Example: Traveling Salesman Problem

\{1,4\} not in S

\[ h = 40.5 \]

\{1,4\} in S

\[ h = \frac{(27+15+18+28+15)}{2} = 51.5 \]
Branch and Bound

Example: Traveling Salesman Problem

- \{1,4\} not in S
  - $h = 40.5$

- \{1,4\} in S
  - $h = 51.5$
Algorithms

Branch and Bound

Example: Traveling Salesman Problem

Expand the partial tour of lowest $h$ value from the PAL (yellow nodes)
Algorithms

Branch and Bound

Example: Traveling Salesman Problem
Branch and Bound

Example: Traveling Salesman Problem

Any vertex with just two edges left must use those edges in a solution
Branch and Bound

Example: Traveling Salesman Problem

\[ h = \frac{(14 + 17 + 18 + 19 + 15)}{2} = 41.5 \]

\[ h = 51.5 \]

\[ h = (14 + 17 + 18 + 19 + 15)/2 = 41.5 \]
Branch and Bound

Example: Traveling Salesman Problem

\[ h = \frac{(14 + 15 + 18 + 20 + 15)}{2} = 41 \]

\[ h = 51.5 \]

\[ h = 41.5 \]
Algorithms

Branch and Bound

Example: Traveling Salesman Problem

{1,4} not in S
\( h = 51.5 \)

{3,4} not in S
\( h = 41.5 \)

{3,4} in S
\( h = 41 \)

{1,4} in S
\( h = 51.5 \)
Branch and Bound

Example: Traveling Salesman Problem

Expand the yellow node with lowest $h$ value
Branch and Bound

Example: Traveling Salesman Problem
Any vertex with just two edges left must use those edges in a solution.
Algorithms

Branch and Bound

Example: Traveling Salesman Problem

Any vertex with two edges in the partial solution cannot have any other edges in a solution
Branch and Bound

Example: Traveling Salesman Problem

Any edge that completes a cycle not containing all vertices cannot be in a solution
Algorithms

Branch and Bound

Example: Traveling Salesman Problem

Red nodes represent feasible solutions
Branch and Bound

Example: Traveling Salesman Problem

A feasible solution has been found – record its total cost

\[ U = 61 \]
Algorithms

Branch and Bound

Example: Traveling Salesman Problem

\[
\text{U} = 61
\]
Branch and Bound

Example: Traveling Salesman Problem

\( U = 61 \)
Algorithms

Branch and Bound

Example: Traveling Salesman Problem

\[ U = 61 \]
Algorithms

Branch and Bound

Example: Traveling Salesman Problem

\[ h = \frac{(14+15+18+20+15)}{2} = 41 \]

\[ U = 61 \]
Algorithms

Branch and Bound

Example: Traveling Salesman Problem

- \( h = 51.5 \) for \( \{1,4\} \) not in \( S \)
- \( h = 41.5 \) for \( \{3,4\} \) not in \( S \)
- \( h = 41 \) for \( \{1,2\} \) not in \( S \)
- \( h = 41 \) for \( \{3,5\} \) not in \( S \)
- \( h = 61 \) for \( \{1,2\} \) in \( S \)
- \( h = 61 \) for \( \{3,5\} \) in \( S \)

\( U = 61 \)
Algorithms

Branch and Bound

Example: Traveling Salesman Problem

$$h = 51.5$$

$$h = 41.5$$

$$h = 41$$

$$U = 61$$
Algorithms

Branch and Bound

Example: Traveling Salesman Problem

\[ U = 61 \]
Branch and Bound

Example: Traveling Salesman Problem

\begin{align*}
\{1,4\} & \text{ not in } S \\
\{3,4\} & \text{ not in } S \\
\{1,2\} & \text{ in } S \\
\{3,5\} & \text{ in } S
\end{align*}

\begin{align*}
h &= 51.5 \\
h &= 41.5 \\
h &= 61 \\
h &= 41 \\
h &= \frac{14 + 18 + 21 + 21 + 18}{2} = 46
\end{align*}

\[ U = 61 \]
Algorithms

Branch and Bound

Example: Traveling Salesman Problem

\[
\begin{align*}
\{1,4\} & \text{ not in } S \\
\{3,4\} & \text{ not in } S \\
\{1,2\} & \text{ not in } S \\
\{3,5\} & \text{ not in } S \\
\{1,4\} & \text{ in } S \\
\{3,4\} & \text{ in } S \\
\{1,2\} & \text{ in } S \\
\{3,5\} & \text{ in } S
\end{align*}
\]

\[h = 51.5\]

\[h = 41.5\]

\[h = 41\]

\[h = 61\]

\[U = 46\]

Blue nodes will not be expanded
Algorithms

Branch and Bound

Example: Traveling Salesman Problem

\[ U = 46 \]
Branch and Bound

Example: Traveling Salesman Problem

Algorithms
Branch and Bound

Example: Traveling Salesman Problem

\[ h = \frac{(24+26+27+20+27)}{2} = 62 \]

\[ U = 46 \]
Algorithms

Branch and Bound

Example: Traveling Salesman Problem

\( U = 46 \)
Branch and Bound

Example: Traveling Salesman Problem

\[ \begin{align*}
\text{\{1,4\} not in } S
\text{\{3,4\} not in } S
\text{\{1,2\} not in } S
\text{\{3,5\} not in } S
\text{\{2,5\} not in } S
\text{\{1,4\} in } S
\text{\{3,4\} in } S
\text{\{1,2\} in } S
\text{\{3,5\} in } S
\text{\{2,5\} in } S
\end{align*} \]

\[ U = 46 \]
Algorithm

Branch and Bound

Example: Traveling Salesman Problem

$U = 46$
Branch and Bound

Example: Traveling Salesman Problem

Algorithms
Algorithms

Branch and Bound

Example: Traveling Salesman Problem

\[ h = \frac{(14+15+18+20+15)}{2} = 41 \]
Branch and Bound

Example: Traveling Salesman Problem

Algorithms

\[ \{1,4\} \text{ not in } S \quad h = 51.5 \]
\[ \{3,4\} \text{ not in } S \quad h = 41.5 \]
\[ \{1,2\} \text{ in } S \quad h = 61 \]
\[ \{3,5\} \text{ not in } S \quad h = 46 \]
\[ \{1,2\} \text{ not in } S \quad h = 62 \]
\[ \{3,5\} \text{ in } S \quad h = 51.5 \]
\[ \{2,5\} \text{ not in } S \quad h = 41 \]
\[ \{2,5\} \text{ in } S \quad h = 62 \]

Stop – \( PAL \) is empty,
Return \( \leftarrow \) path

\[ U = 41 \]
Algorithms

Depth First Search

Algorithm Template:

\textbf{DFS}(V, E) :

For all \( v \) in \( V \) mark \( v \) “new”

While there is a vertex \( v \) in \( V \) marked “new”, \textbf{Search}(v)

\textbf{Search}(v) :

Mark \( v \) “old” and visit \( v \)

For each vertex \( w \) in \( V \) that is adjacent to \( v \) do the following:

If \( w \) is marked “new” then do the following:

Visit edge \( \langle v, w \rangle \)

\textbf{Search}(w)
Linear Programming

Description:

Given a $n \times m$ real-valued matrix $A$, and $n$-dimensional vectors $b$ and $c$, find an $x$ such that

$$c^T x$$

is maximum

Subject to:

$$A_1 x \leq b_1$$

$$A_2 x = b_2$$

where $A$ has been partitioned into $A_1$ and $A_2$ and $b$ has been partitioned into $b_1$ and $b_2$
Algorithms

Linear Programming

Example:

Maximum flow in a network:

**Given:** a graph $G = (V, E)$ with distinguished vertices 
$s, t \in V$ and a “flow capacity” $b_e$ for each edge $e \in E$ 
(such a structure is called a flow network).

**Find:** the maximum total flow through the given flow 
network such that no edge carries a flow greater than its 
capacity.

Note: total flow at any junction of pipes is always 0, 
same as Kirkoff's law
Example:

Maximum flow in a network:

Numbers are capacities on the edges, letters are flow variables.
Algorithms

Linear Programming

Example:

Maximum flow in a network:

Numbers are capacities on the edges, letters are flow variables.
Example:

Maximum flow in a network:

Maximize $S$

Subject to: $S - a - e = 0$
$e - b - f = 0$
$f - c - d = 0$
$a + b - g = 0$
$c + g - h = 0$
$d + h - S = 0$

$a \leq 4$, $b \leq 3$, $c \leq 4$, $d \leq 7$, $e \leq 5$, $f \leq 6$, $g \leq 3$, $h \leq 2$
$-a \leq 4$, $-b \leq 3$, $-c \leq 4$, $-d \leq 7$, $-e \leq 5$, $-f \leq 6$, $-g \leq 3$, $-h \leq 2$
Maximize:

\[
(0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 1)^T \times \begin{pmatrix}
  a \\
  b \\
  c \\
  d \\
  e \\
  f \\
  g \\
  h \\
  S
\end{pmatrix}
\]
Subject to:
\[
\begin{pmatrix}
-1 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 1 \\
0 & -1 & 0 & 0 & 1 & -1 & 0 & 0 & 0 \\
0 & 0 & -1 & -1 & 0 & 1 & 0 & 0 & 0 \\
1 & 1 & 0 & 0 & 0 & 0 & -1 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 0 & 1 & -1 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & -1 \\
\end{pmatrix}
\begin{pmatrix}
a \\ b \\ c \\ d \\ e \\ f \\ g \\ h \\ S \\
\end{pmatrix}
= 
\begin{pmatrix}
0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
\end{pmatrix}
\]

\[
\begin{pmatrix}
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
-1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 \\
\end{pmatrix}
\begin{pmatrix}
a \\ b \\ c \\ d \\ e \\ f \\ g \\ h \\ S \\
\end{pmatrix}
\leq 
\begin{pmatrix}
4 \\
4 \\
3 \\
3 \\
4 \\
4 \\
7 \\
7 \\
5 \\
5 \\
6 \\
6 \\
3 \\
3 \\
2 \\
2 \\
\end{pmatrix}
\]