

*University of Cincinnati*  
*Department of Electrical & Computer Engineering and Computer Science*

**20 ENFD 112 – Fundamentals of Programming**

**LABORATORY 7: SORT, ARRAYS, ALGORITHMS**

Spring 2008

## 1. Objective

The objective of this assignment is to exercise your ability to sort, hold data read from a file, use arrays to establish a loop invariant, implement an algorithm for solving a basic real-world optimization problem.

## 2. Problem 1 - Reading data from file

Given a file containing a collection of costs of laying a cable between two cities. Cities are identified by a number. All integers from 1 to the number of cities are used for city identities. The first line of the file contains two numbers stating the number of cables and the number of cities, in that order. Each remaining line specifies the cost of laying a particular cable. The format of such a line is

`<city-identity-1> <city-identity-2> <cost>`

where all three tokens are numbers. The meaning of each line is: the cost of a cable laid between `<city-identity-1>` and `<city-identity-2>` is `<cost>`. An example of a small input file is this:

```
10 5
1 5 100
2 4 141
3 5 121
2 3 138
4 5 102
1 2 139
1 3 143
1 4 139
2 5 101
3 4 159
```

The problem is to read the data of a file with this format into a matrix called `cables` with the number of rows equal to the number of cables and the number of columns equal to three. Each row of `cables` will contain two cities (of type number) and the cost of a cable between them (also of type number). See the `Help` section in case of trouble.

## 3. Problem 2 - Sorting the data

Sort `cables` so that all elements of any row remain together in a row but all rows are in increasing order of cost (most likely the third column). See the `Help` section in case of trouble.

## 4. Problem 3 - Creating a group array

Create a one-dimensional array `group` that has a number of elements equal to the number of distinct

cities with respect to the file read in above and is such that all its elements are distinct integers. See the **Help** section in case of trouble.

## 5. Problem 4 - Finding a Minimum Cost Spanning Network

Let  $C$  be a list of cities. Let  $D$  be a list of cables, each of which connects two cities in  $C$ . For all  $d \in D$ , define  $f(d)$  to be the cost of cable  $d$ . A sublist  $D' \subset D$  of cables which allows every city in  $C$  to connect with every other city in  $C$  is called a *spanning network* of  $(C, D, f)$ . The cost of a spanning network  $D'$  is the sum of the costs of the cables in  $D'$ : that is  $\sum_{d \in D'} f(d)$ . The *minimum cost spanning network* of  $(C, D, f)$  is a spanning network of  $(C, D, f)$  which has cost no greater than any other spanning network of  $(C, D, f)$ . The problem is, given  $C$ ,  $D$ , and  $f$ , find the minimum cost spanning network of  $(C, D, f)$ .

Your program should output the total minimum cost and the number of cables involved in the solution. Your program should maintain an array `solution` which holds all cables in the solution so that typing `solution` at the MATLAB prompt will show the cables themselves.

### 5.1 Analysis

An algorithm that works is this:

```
Create a solution list S, initially empty.
Read in all the data to the cables array.
Sort the cables array by increasing order of cost.
For each cable d in the cables array, in order, do the following:
    Let c1 and c2 be the cities d connects
    If c1 and c2 are NOT yet connected by cables in S add d to S.
end
```

The first line is trivial to program. The next two lines are taken care of by the solutions to the first and second problems above. The fourth line is a for loop where a particular row or column of `cables` (you choose, but choose carefully) is considered on each iteration. This is taken care of by indexing into the array. The sixth line, and the only one left to consider, is tricky. How can we determine easily whether two cities are connected by cables in list  $S$ ?

What we can try to do is associate a number with each city in such a way that when the numbers of two cities are consulted, we can tell right away whether they are connected. What would be a good association? Well, suppose the number associated with two cities is the same if the two cities are connected and different if they are not connected? Then the test is simple: compare the two numbers for equality. We try to build on this idea. How do we associate a number with a city? We can create an array with a number of elements equal to the number of cities and use the  $i^{th}$  element of the array to hold the association with the  $i^{th}$  city. Call this array `group`. The test is something like

```
if group(cable(i,1)) ~= group(cable(i,2))
```

where `cable(i,...)` are elements of a single cable from the sorted array. But how does the `group` array get its values? Well, recall that we are trying to determine connectedness among cables in  $S$ . Initially, there are no cables in  $S$ . That means no two cities are connected, initially. By our definition of `group` values, all elements of `group` should initially be different. You can make them different any way you like, it does not matter. You did this already in the solution to the third problem above. When do `group` values get changed? When a cable is placed into

the solution list  $S$ . What should happen to the elements of `group` then? Let `cable(i,1)` and `cable(i,2)` be the two cities connecting a cable that is about to be placed in list  $S$ . A number of cities may have the same `group` number as `cable(i,1)` and a number of cities may have the same `group` number as `cable(i,2)`. We need to change the `group` numbers of the latter cities to equal `group(cable(i,1))`. Then the meaning of `group` is back again to what we want it to be. A `for` loop can do this.

The only thing we haven't considered is how to add a cable to  $S$ . This is covered in the help section.

## 6. Help

To open a file use something similar to this:

```
fid = fopen(filename,'r');    % open the data file, read only
```

To read the next number in a file use something like this:

```
n = fscanf(fid,'%d',1);      % get the number of data points
```

To read a row at a time use something like this:

```
cable = fscanf(fid,'%d',3);  % read a cable (3 numbers) from file
```

To add a row to an array use something like this:

```
[cables] = [cables cable];  % append a cable to an array cables
```

To sort a matrix by rows use `sortrows` (you can do a help on this function at the MATLAB prompt). Be careful, the matrix to sort may have to be transposed! To define an empty solution array use:

```
solution = [];
```

To define a `group` array with different integers, the easiest thing to do is use `1:m` where  $m$  is the number of elements in the array. To add a cable to the solution array use:

```
[solution] = [solution c];
```

where `c` contains the data of the cable to add but the data is organized differently.

## 7. Submission

Submit an `m` file which solves the problem of Section 5 on or before May 25 using blackboard. See the course webpage at

<http://gauss.ececs.uc.edu/Courses/HTML/E112.html>

for instructions.