Report: Algorithm Design Term Project

Subject: CS3201 Algorithm Design

Submitted to

Asst. Prof. Dr. Thitipong Tanprasert

Submitted by

1. Pichayuth Kittisriswai 5713091

This report is a part of the course Algorithm Design in the semester 1/2016

Problem Overview

Problem Code: *1982*

Problem Name: *Electrification Plan*

Problem Description: *Some country has n cities. The government has decided to electrify all these cities. At first, power stations in k different cities were built. The other cities should be connected with the power stations via power lines. For any cities i, j it is possible to build a power line between them in c_{ij} roubles. The country is in crisis after a civil war, so the government decided to build only a few power lines. Of course from every city there must be a path along the lines to some city with a power station. Find the minimum possible cost to build all necessary power lines.*

Input

The first line contains integers n and k ($1 \le k \le n \le 100$). The second line contains k different integers that are the numbers of the cities with power stations. The next n lines contain an $n \times n$ table of integers $\{c_{ij}\}$ ($0 \le c_{ij} \le 10^5$). It is guaranteed that $c_{ij} = c_{ji}$, $c_{ij} > 0$ for $i \ne j$, $c_{ii} = 0$

Output

Output the minimum cost to electrify all the cities.

Sample

Input	Output
4 2 1 4 0 2 4 3 2 0 5 2 4 5 0 1 3 2 1 0	3

Problem Author: *Mikhail Rubinchik*

Problem Source: *Open Ural FU Championship 2013*

Difficulty: *130*

Restriction: *Time limit 0.5 second*

Memory limit 64MB

Code:

```
read1=raw_input().split()
city = int(read1[0])
pCity = int(read1[1])
powerStation = [0 for i in range(city)]

hasPower  = [False for i in range(city)]

a = [[0 for i in range(city)]for j in range(city)]
read2 = raw_input().split()

for i in range(pCity):
    powerStation[i] = (int(read2[i]))
    hasPower[powerStation[i]-1] = True

for i in range(city):
    read3 = raw_input().split()
    for j in range(city):
        a[i][j] = int(read3[j])

def elec(K,n):

    mnCost = mxCost = mnCostId = totalCost = 0

    for k in range(K,n):
        mnCost = 99999999

        for i in range(n):
            if(hasPower[i] == False):

                for j in range(k):
                    if(a[i][powerStation[j]-1] < mnCost and i != (powerStation[j]-1)):
                        mnCost = a[i][powerStation[j]-1]
                        mnCostId = i

        totalCost += mnCost
        hasPower[mnCostId] = True
        powerStation[k] = mnCostId+1


    print totalCost

elec(pCity,city)
```

Part I

```
read1=raw_input().split()
city = int(read1[0])
pCity = int(read1[1])
powerStation = [0 for i in range(city)]

hasPower  = [False for i in range(city)]

a = [[0 for i in range(city)]for j in range(city)]
read2 = raw_input().split()

for i in range(pCity):
    powerStation[i] = (int(read2[i]))
    hasPower[powerStation[i]-1] = True

for i in range(city):
    read3 = raw_input().split()
    for j in range(city):
        a[i][j] = int(read3[j])
```

For the first part, I receive the input and assign these values into the variables.

city *will store the number of all cities in the country.*

pCity *will store the number of all cities that already have the power station*

powerStation *is an array that store the cities that have power station*

hasPower *is an array that store the cities which already have the power station or the cities that are connected with the power line.*

Part 2

```
def elec(K,n):

    mnCost = mxCost = mnCostId = totalCost = 0

1   for k in range(K,n):
        mnCost = 99999999

    2   for i in range(n):
            if(hasPower[i] == False):

                3   for j in range(k):
                        if(a[i][powerStation[j]-1] < mnCost and i != (powerStation[j]-1)):
                            mnCost = a[i][powerStation[j]-1]
                            mnCostId = i

    4   totalCost += mnCost
        hasPower[mnCostId] = True
        powerStation[k] = mnCostId+1


    print totalCost

elec(pCity,city)
```

In this part will contain the main logic of the algorithm. For the first for loop will run equal to the number of non electricity cities, for example if there is 1 city that has electricity out of 3 cities the first loop will run 2 times. After getting into the first loop, I create another variable which is ***mnCost*** *to store the minimum cost to build the power line for city* ***k****. For the second for loop, this loop will run through all the cities and check whether that city* ***i*** *already have the power. If the city* ***i*** *doesn't have the power yet, the third loop will be called. For the third loop, I will compare the cost from the city* ***i*** *to the cities that already have the electricity and store it in the variable* ***mnCost*** *and also keep the city* ***i****. For the last part, I will keep the minimum cost from each city* ***k*** *into* ***totalCost*** *and also mark the city* ***i*** *as already have the electricity and add the city* ***i*** *into* ***powerStation*** *array. I have to add 1 to the* ***powerStation[k] = mnCostId + 1*** *,because the index start with 0 but the input cities start from 1.*

Runtime Analysis

*For the worst-case, there is only one city that already have the electricity out of **n** cities. So the first for loop will run **(n)** times and for the second for loop will have to run **(n)** times and the last inner for the worst case will also run (**n**) times. So when we multiply all the times from 3 loops, the running time is **O(N^3)***

Test set

#1

Input	Output
4 2 1 4 0 2 4 3 2 0 5 2 4 5 0 1 3 2 1 0	3

#2

Input	Output
5 2 1 3 0 1 2 3 4 1 0 2 3 4 2 2 0 3 4 3 3 3 0 4 4 4 4 4 0	8

#3

Input	Output
4 2 1 4 0 2 6 3 2 0 5 2 4 6 0 1 3 2 2 0	3

#4

Input	Output
4 2 1 4 0 4 8 9 4 0 2 7 8 2 0 1 9 7 1 0	3

#5

Input	Output
4 2 1 3 0 6 7 8 6 0 2 3 7 2 0 4 8 3 4 0	5

#6

Input	Output
4 1 4 0 1 1 100000 1 0 1 100000 1 1 0 100000 100000 100000 100000 0	100002

Result

#1 - #3

```
 RESTART: D:/OneDrive/University/Year 3/Algorithym Design/Weekly exercise/Code/After Mid/termproject2.py
4 2
1 4
0 2 4 3
2 0 5 2
4 5 0 1
3 2 1 0

3
>>>
 RESTART: D:/OneDrive/University/Year 3/Algorithym Design/Weekly exercise/Code/After Mid/termproject2.py
5 2
1 3
0 1 2 3 4
1 0 2 3 4
2 2 0 3 4
3 3 3 0 4
4 4 4 4 0

8
>>>
 RESTART: D:/OneDrive/University/Year 3/Algorithym Design/Weekly exercise/Code/After Mid/termproject2.py
4 2
1 4
0 2 6 3
2 0 5 2
4 6 0 1
3 2 2 0

3
```

#4 - #6

```
 RESTART: D:/OneDrive/University/Year 3/Algorithym Design/Weekly exercise/Code/After Mid/termproject2.py
4 2
1 4
0 4 8 9
4 0 2 7
8 2 0 1
9 7 1 0

3
>>>
 RESTART: D:/OneDrive/University/Year 3/Algorithym Design/Weekly exercise/Code/After Mid/termproject2.py
4 2
1 3
0 6 7 8
6 0 2 3
7 2 0 4
8 3 4 0

5
>>>
 RESTART: D:/OneDrive/University/Year 3/Algorithym Design/Weekly exercise/Code/After Mid/termproject2.py
4 1
4
0 1 1 100000
1 0 1 100000
1 1 0 100000
100000 100000 100000 0

100002
```

Timus Result

7152093	06:39:29 1 Dec 2016	pichayuth	1982. Electrification Plan	Python 2.7	Accepted		0.046	552 KB

Reference:

http://acm.timus.ru/problem.aspx?space=1&num=1982