I.T.U.

Faculty of Computer and Informatics

Computer Engineering

Lesson name: Analysis of Algorithms

Lesson Code: BLG 372E

Name Surname: Abdullah AYDEĞER

Number: 040090533

Instructor's Name: Zehra ÇATALTEPE

Assistant's Name: Meryem UZUN

Due Date: 20.05.2012

Table of Contents

Introduction

Maximum flow is one of the important problems. Maximum flow problem is trying to get maximum flow possibility from given source to sink. According to problem, source and sink can be transportation agencies, telecommunication network etc. In this Project I've implemented problem of maximum flow of transportation agencies.

My Algorithms

Finding Max Flow

My algorithm for finding maximum flow solution:

- For all possible paths in the given capacities, determining the agencies on the found path.
- According to found path, calculating minimum capacity of agencies on the path can flow.
- This calculated capacity subtracted all agencies on the found path, and added all inversion path(for residual graph).

```
while(BFS(1, lastNodeNum)) {     //while there is a possible 1-lastNodeNum(s-t) path
    minumumCapacity = flows[nodeBeforeMe[lastNodeNum]][lastNodeNum];
    temp = lastNodeNum;
    while(nodeBeforeMe[temp] != temp) {      //while for determining minumum capacity of the found path
       isMaxCap = false;
       cout << temp << "-";
       if(flows[nodeBeforeMe[temp]][temp] < minumumCapacity)
          minumumCapacity = flows[nodeBeforeMe[temp]][temp];
       temp = nodeBeforeMe[temp];
    }
    cout << temp;

    temp = lastNodeNum;
    while(nodeBeforeMe[temp] != temp) {      //while for subtraction to used capacity to the all agencies on the found path
       flows[nodeBeforeMe[temp]][temp] -= minumumCapacity;   //We've already gone here, so that delete our cost
       flows[temp][nodeBeforeMe[temp]] += minumumCapacity;   //For residual graph
       temp = nodeBeforeMe[temp];
    }
    cout << " " << minumumCapacity << "\n\n\t";
    totalFlow += minumumCapacity;
}
```

Figure 1. Finding Max Flow

Searching the Path between given agencies

My algorithm for searching the path is BFS(breath first search):

- Firstly, queue is initialized the given source node and node before source node is itself.
- For not found the sink node and queue is not empty, taking first element to the queue and trying to use any node(agency) for transporting. If any node can flow, then this is added to the queue and for determining path the node number is added to the before's array(nodeBeforeMe).
- If the trying node is sink node, then return true. Else try to get path to sink node.

```
assignAll(nodeBeforeMe, -1, lastNodeNum);    //firstly, make all before's -1. That will be used to determine path agencies
LAYER.push(s);
nodeBeforeMe[s] = s;

while( !LAYER.empty() && !found) {   //while not found and queue is not empty
   currentElement = LAYER.front();   //getting first element of queue
   LAYER.pop();  //pop from the queue
   for(int i=1; i<=lastNodeNum; i++) {   //for all possible agencies
      if(flows[currentElement][i] != 0 && nodeBeforeMe[i] == -1) {   //if there is a flow different from 0
         LAYER.push(i);                                              //and the agency is not expired before
         nodeBeforeMe[i] = currentElement;
          if(i == t) {
            found = true;
            break;
          }
      }
   }
}
return found;
```

Figure 2. BFS(returning boolean) in C++

Reading the input file for Mission 1

- For all calling Mission 1, before finding max flow between source to sink nodes, I've filled the necessary arrays with reading the input file. That can be seen in the below figure.

```
void Agency::readFlowsBeginning(string inputName) {
    /*****************************************************************************
    *   Function Name     : readFlows                                             *
    *   Aim to be written : reading the given input file to the class parameter 'flows' *
    *                       (will be used before each Mission 1 calling         *
    *   Parameters        : string parameter of input file name                   *
    *   Return value      : ---                                                   *
    *****************************************************************************/
    for(int i=0; i <= lastNodeNum; i++){
        for(int j=0; j <= lastNodeNum; j++){
            flows[i][j] = 0;                    //firstly, there is no flow
        }
    }

    while(!read.eof()){
        read >> v1 >> v2 >> val;
        flows[v1][v2] = val;        //fill the all flows according to given input file
    }
```

Reading the input file for Mission 2

- For all calling Mission 2, before determining is this maximum flow or not, I've filled the necessary arrays with reading the input file or capacities(that is read before). That can be seen in the below figure.

```
void Agency::readFlows(string inputName) {
    /*****************************************************************************
    *   Function Name     : readFlows                                             *
    *   Aim to be written : reading the given input file to the class parameter 'flows' *
    *                       (will be used before each Mission 2 calling         *
    *   Parameters        : string parameter of input file name                   *
    *   Return value      : ---                                                   *
    *****************************************************************************/
    for(int i=0; i <= lastNodeNum; i++){
        for(int j=0; j <= lastNodeNum; j++){
            flows[i][j] = capacities[i][j];     //filling flows to real capacities
        }
    }

    while(!read.eof()){
        read >> v1 >> v2 >> val;
        flows[v1][v2] = capacities[v1][v2] - val;   //If given input used the v1-v2 capacity then minus it from cap..
        flows[v2][v1] = val;    //for residual graph
    }
```

Data Structures

For implementing given problem(maximum transition), I've used Agency class with necessary parameters and methods as seen in the below figure.

```
class Agency{
    private:
        int lastNodeNum;    //sink node determining according to the input file last line
        int totalFlow;      //Total flow that can be gone from source node to sink node
        int *nodeBeforeMe;  //for used determining the path from source to sink
        int **flows;        //two dimensional array, that says flows[1][2] from 1. agency to 2.agency flow
        int **capacities;   //two dimensional array, that holds capacities of all possible transitions
        void assignAll(int *, int, int);
        void read(string );
        void readToGraph(string , int );
        bool BFS(int, int);
    public:
        Agency(string );
        ~Agency();
        int getTotalFlow();
        void readFlows(string );
        void findMaxFlow(int );
        void readFlowsBeginning(string );

};
```

Compilation and Running

I have compiled, without any error, my code in Win7 and Linux operating systems. For Linux environment I have tried to compile my code not only in the virtual machine for Linux(xubuntu environment) but also Ubuntu 11.10 version. My compilation code for Linux:

g++ main.cpp –o main

Running command:

./main file_name1 file_name2