**★★★☆☆**

**題組：Problem Set Archive** **with Online Judge**

**題號：10801: Lift Hopping**

**解題者：丘珮珊**

**解題日期：2015年5月14日**

**題意：**

一摩天大樓高度不超過100 層樓(編號0~99) ，內有n(1~5)台電梯。每台電梯上下一層樓的速度T(1~100秒/樓)都不同，且各電梯只停靠特定樓層。若要換搭另一台電梯，則兩電梯都要停在同一樓，並須等待60秒。求從0樓搭電梯到k樓所需的最時間(秒)，不可能到達則輸出IMPOSSIBLE。

**題意範例：**

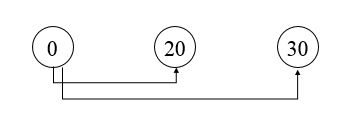
2 30 2台電梯，目的k為30樓  
10 5 電梯1速度為10秒/樓，電梯2速度為5秒/樓  
0 1 3 13 15 20 99 電梯1停靠的樓層  
4 13 15 19 20 30 電梯2停靠的樓層  
=>275 (130+60+85)  
 電梯1，0~13樓，13\*10=130秒  
 換電梯2，60秒  
 電梯2，13~30樓，17\*5=85秒

3 50 3台電梯，目的k為50樓  
10 50 100 速度分別為10秒/樓、50秒/樓、 100秒/樓  
0 10 30 40 電梯1停靠的樓層  
0 20 30 電梯2停靠的樓層  
0 20 50 電梯3停靠的樓層  
=>3920 (300+60+500+60+3000)  
 電梯1，0~30樓，30\*10=300秒，換電梯等60秒  
 電梯2，30~20樓，10\*50=500秒，換電梯等60秒  
 電梯3，20~50樓，30\*100=3000

**解法：**

先計算不換電梯時各層樓間移動所需的最短時間(計算權重)，最後再計算換電梯時0樓到各樓所需的最短時間(尋找最短路徑)。

1. 不換電梯： (計算權重、找相鄰的點)

二維陣列time[i][j]：i到j樓所需的最短時間，初始值為無限大。每當輸入電梯停靠的樓層時，更新time：  
有一電梯的速度為t秒/樓，以此電梯可停靠的各樓層i當作起始點，若到達其他可停靠樓層j的時間比time所存的短  
(|i-j|\*t<time[i][j])，則更新time的值。 **2.**換電梯：(尋找最短路徑)  
(1)Dijkstra演算法：

一維陣列mint[i]：0到i樓所需的最短時間，初始值為不換電梯的最短時間(time[0][i])。

Queue Q：紀錄已是最小值的點，初始值為0樓。

(i)找到mint[]的最小值mint[i]，且i不在Q中，將i push入Q。

(ii)以i為中繼點重新計算mint：j=可停靠的樓層，若0到i樓+換電梯60秒+i到j樓，小於目前0到j樓的時間(mint[i]+60+time[i][j]<mint[j])，則更新mint[j]。

(iii)步驟重複步驟(1)、(2)直到Q內含有所有可停靠的樓層(或Q含有k樓時結束)。  
(2)Bellman-Ford演算法：時間複雜度O(VE)。

一維陣列mint[i]：0到i樓所需的最短時間，初始值為不換電梯的最短時間(time[0][i])。

(i)以i為中繼點重新計算mint：i、j=可停靠的樓層，窮舉所有(i,j)，使mint[i]+60+time[i][j]< mint[j]，更新mint[j]。

(ii)重複步驟(1)V(# of vertex)-1次。  
(3)SPFA(Shortest Path Faster Algorithm)演算法：Bellman-Ford演算法優化，減少步驟(1)重複的次數。

一維陣列mint[i]：0到i樓所需的最短時間，初始值為不換電梯的最短時間(time[0][i])。  
queue Q：欲成為中繼點更新mint的樓層，初始值為0可到達的樓層(time[0][i]!=無限大)。

(i)pop Q得i。

(ii)以i為中繼點重新計算mint：j=可停靠的樓層，若mint[i]+60+time[i][j]<mint[j]，更新mint[j]。

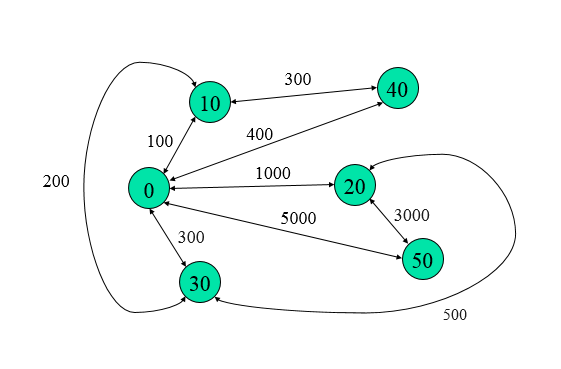
(iii)當mint[j]的值更改時，代表以j為中繼點可能出現更短的路徑，因此將j push入Q(若j在Q中則不用)。

(iv)重複步驟(1)~(3)直到Q內為空。

3.輸出：  
若mint[k]=無限大則輸出IMPOSSIBLE，否則mint[k]即為所求。

**解法範例：**3 50 3台電梯，目的k為50樓  
10 50 100 速度分別為10秒/樓、50秒/樓、 100秒/樓  
0 10 30 40 電梯1停靠的樓層  
0 20 30 電梯2停靠的樓層  
0 20 50 電梯3停靠的樓層  
1.不換電梯： (計算權重)| i-j |\*t < time[i][j]

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **time** | **0** | **10** | **20** | **30** | **40** | **50** | **…** |
| **0** | = | 100 | 1000 | 300 | 400 | 5000 |  |
| **10** | 100 | = |  | 200 | 300 |  |  |
| **20** | 1000 |  | = | 500 |  | 3000 |  |
| **30** | 300 | 200 | 500 | = | 100 |  |  |
| **40** | 400 | 300 |  | 100 | = |  |  |
| **50** | 5000 |  | 3000 |  |  | = |  |
| **…** |  |  |  |  |  |  | = |

****2.換電梯：(尋找最短路徑、找相鄰的點)SPFA演算法

mint[i]+60+time[i][j]< mint[j]

初始化：Push:10、20、30、40、50

Q mint

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 10 | 20 | 30 | 40 | 50 |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 0 | 10 | 20 | 30 | 10 | 50 |
| 0 | 100 | 1000 | 300 | 400 | 5000 |

Pop:10

Q mint

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 20 | 30 | 40 | 50 |  |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 0 | 10 | 20 | 30 | 10 | 50 |
| 0 | 100 | 1000 | 300 | 400 | 5000 |

Pop:20、Push:50(已存在)

Q mint

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 30 | 40 | 50 |  |  |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 0 | 10 | 20 | 30 | 10 | 50 |
| 0 | 100 | 1000 | 300 | 400 | 4060 |

Pop:30、Push:20

Q mint

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 40 | 50 | 20 |  |  |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 0 | 10 | 20 | 30 | 10 | 50 |
| 0 | 100 | 860 | 300 | 400 | 4060 |

Pop:40

…(略)

Pop:50

…(略)

Pop:20、Push:50

Q mint

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 50 |  |  |  |  |

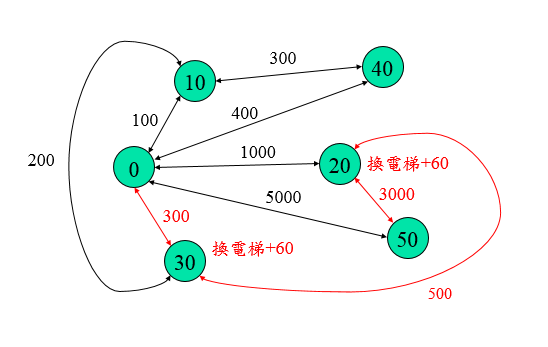
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 0 | 10 | 20 | 30 | 10 | 50 |
| 0 | 100 | 860 | 300 | 400 | 3920 |

Pop:50

Q mint

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  |  |  |  |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 0 | 10 | 20 | 30 | 10 | 50 |
| 0 | 100 | 860 | 300 | 400 | 3920 |

****

**討論：**

(1)一開始看到題目知道要找最短路徑，但卻不知該如何下手，後來想到最短路徑問題與圖有很大的關連，所以找出圖的特徵後，就能比較容易解題。例如本題：找出相鄰的點、各邊的權重。

**程式：**

#include <iostream>

#include <vector>

#include <sstream>

#include <string>

#include <queue>

#define Inf 99999

using namespace std;

void Drawtab(vector< vector<int> > \*weight); //debug

int main()

{

int n, k; //n elevator, floor k

int t[5]; //speed of every elevator

while(cin >> n >> k) //n elevators, floor 0 to floor k

{

int i; //be used in for loop

vector<int> mint(100); //min time of floor 0 to every floor

vector<bool> inqueue(100); //element i weather in queue(inqueue[i])

queue<int> que; //SPFA

vector< vector<int> > weight(100); //graphic weight

//initialization

mint.assign(100, Inf);

inqueue.assign(100, false);

for(i=0; i<100; ++i)

weight[i].assign(100, Inf);

//Part 1:make the graph

//input every elevator"s speed

for(i=0; i<n; ++i)

cin >> t[i];

cin.ignore();

//input which floor elevator can stop. then find the weight

for(i=0; i<n; ++i)

{

stringstream ss;

string str;

vector<int> floor(100);

int j = 0;

//input floors

getline(cin, str);

ss.str(str);

while(ss >> floor[j])

++j;

//find the weight

for(int a=0; a<j; ++a)

{

for(int b=a; b<j; ++b)

{

int f1 = floor[a];

int f2 = floor[b];

if((f2-f1)\*t[i] < weight[f1][f2])

{

weight[f1][f2] = (f2-f1)\*t[i];

weight[f2][f1] = weight[f1][f2];

}

} //END: for

}//END: for(int a=0; a<j; ++a)

}

//END: input which floor elevator can stop. then find the weight

//Drawtab(&weight);

//Part2:shortest path-SPFA

//initialization

mint[0] = 0;

for(i=1; i<100; ++i)

{

if(weight[0][i]!=Inf)

{

que.push(i);

inqueue[i] = true;

mint[i] = weight[0][i];

} //END: if

} //END: for

//Find shortest path

while(!que.empty())

{

int f = que.front();

que.pop();

inqueue[f] = false;

//find mint

for(i=0; i<100; ++i)

if(mint[f]+weight[f][i]+60 < mint[i])

{

mint[i] = mint[f]+weight[f][i]+60;

if(!inqueue[i])

{

que.push(i);

inqueue[i] = 1;

}

} //END: if(mint[f]+weight[f][i]+60 < mint[i])

}

//END: Find shortest path

//Part3:output

if(mint[k]!=Inf)

cout << mint[k] << endl;

else

cout << "IMPOSSIBLE" << endl;

} //END: while(cin >> n >> k)

return 0;

}

void Drawtab(vector< vector<int> > \*weight)

{

for(int i=0; i<=50; i+=10)

{

for(int j=0; j<=50; j+=10)

{

if((\*weight)[i][j] != Inf)

cout << (\*weight)[i][j] << " ";

else

cout << "Inf" << " ";

}

cout << endl;

}

}