Inter-City Bus

Có n thành phố 1, 2, ..., n. Giữa 2 thành phố i và j có thể có 1 con đường (2 chiều) kết nối giữa chúng.

Mỗi thành phố i có tuyến buýt i với C[i] là giá vé mỗi khi lên xe và D[i] là số thành phố tối đa mà buýt i có thể đi đến trên 1 hành trình đi qua các con đường kết nối.

Hãy tìm cách đi từ thành phố 1 đến thành phố n với số tiền phải trả là ít nhất

Input

- Dòng 1: chứa 2 số nguyên dương n và m trong đó n là số thành phố và m là số con đường kết nối các thành phố (1 <= n <= 5000, 1 <= m <= 10000)
- Dòng i+1 (i = 1,2,...,n): chứa 2 số nguyên dương C[i] và D[i] (1 <= C[i] <= 10000, 1 <= D[i] <= 100)
- Dòng n+1+i (i = 1, 2, ..., m): chứa 2 số nguyên dương i và j trong đó giữa thành phố i và j có con đường kết nối

Output

- Số tiền tối thiểu phải bỏ ra để đi buýt từ thành phố 1 đến thành phố n

Inter-City Bus

Algorithm

- Run BFS for computing the cost for traveling with one bus from a city i to another reachable city j → Cost graph G
- Apply the Dijkstra algorithm for finding the shortest path from city 1 to city n in G

Inter-City Bus

Input

6 6

10 2

30 1

50 1

20 3

30 1

20 1

1 2

1 3

1 5

2 4

2 5

4 6

Output

30

Giải thích:

-Lên buýt 1 từ thành phố 1 đến thành phố 4 mất 10 đồng

-Lên buýt 4 từ thành phố 4 đến thành phố 6 mất 20 đồng

Tổng cộng mất 10 + 20 = 30 đồng

Implementation – Inter-City Bus

```
#include <bits/stdc++.h>
using namespace std;
#define N 5001
#define INF 1000000000
int w[N][N]; // w[u][v] is the weight of the arc(u,v)
int n,m;
int C[N], D[N];
int d[N]; // d[i] is the distance from the source to i during the BFS
bool f[N]; // f[v] = true means that the shortest path from s to v was found
vector<int> A[N]; // A[v] is the list of adjacent cities of v
```

Implementation – Inter-City Bus

```
void BFS(int i){
    queue<int> Q;
    for(int j = 1; j <= n; j++) d[j] = -1;
    d[i] = 0; Q.push(i);
    while(!Q.empty()){
        int v = Q.front(); Q.pop();
        //cout << "BFS(" << i << ") POP v = " << v << endl;
        w[i][v] = C[i];// weight of the arc (i,v)  is the price of the bus at city i
        for(int j = 0; j < A[v].size(); j++){
            int u = A[v][j];
            if(d[u] >= 0) continue; // u has been visited
            d[u] = d[v] + 1;
            if(d[u] <= D[i]) Q.push(u);
            //cout << "BFS(" << i << ") PUSH u = " << u << endl;
        }
    }
}
```

Implementation – Inter-City Bus

```
void dijkstra(int s, int t){
    for(int v = 1; v <= n; v++){ d[v] = w[s][v]; f[v] = false;}
    d[s] = 0; f[s] = true;
    for(int i = 1; i <= n; i++){
        int u = -1; int minD = INF;
        // select a node u having minimum d[u]
        for(int v = 1; v <= n; v++) if(!f[v]){
            if(d[v] < minD){ u = v; minD = d[v]; }
        }
        f[u] = true;
        if(u == t) break; // the result was found
        for(int v = 1; v <= n; v++) if(!f[v]){
            if(d[v] > d[u] + w[u][v]){
                d[v] = d[u] + w[u][v];
            }
        }
    }
    cout << d[t];
}
```

Implementation – Inter-City Bus

```
void buildGraph(){
    for(int i = 1; i <= n; i++)
        for(int j = 1; j <= n; j++)
            w[i][j] = INF;
    for(int i = 1; i <= n; i++){
        BFS(i);
    }
}
```

Implementation – Inter-City Bus

```
void input(){
    cin >> n >> m;
    for(int i = 1; i <= n; i++){
        cin >> C[i] >> D[i];
    }
    for(int i = 1; i <= m; i++){
        int u,v;
        cin >> u >> v;
        A[u].push_back(v);
        A[v].push_back(u);
    }
}
void solve(){
    buildGraph();
    dijkstra(1,n);
}

int main(){
    input();
    solve();
    return 0;
}
```