HUST

ĐẠI HỌC BÁCH KHOA HÀ NỘI

HANOI UNIVERSITY OF SCIENCE AND TECHNOLOGY

ONE LOVE. ONE FUTURE.

Applied Algorithm Lab

Bus inter-city

ONE LOVE. ONE FUTURE.

Bus inter-city

- There are n city: 1, 2,…, n.
  - Between 2 cities, there may exist a **bidirectional road** connecting them.
  - Each city **i** has a **bus route i** with:
    - **C[i]**: The fare required each time a passenger boards the bus.
    - **D[i]**: The maximum number of cities the bus can travel through in a single journey using the connected roads.
- **Objective:** Find a path from city 1 to city n with lowest cost.
- **Input:** n, m (#roads), {C(1), D(1)},..., {C(n), D(n)}, the set of roads
- **Output:** The minimum cost of the path

Bus inter-city

- Example

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

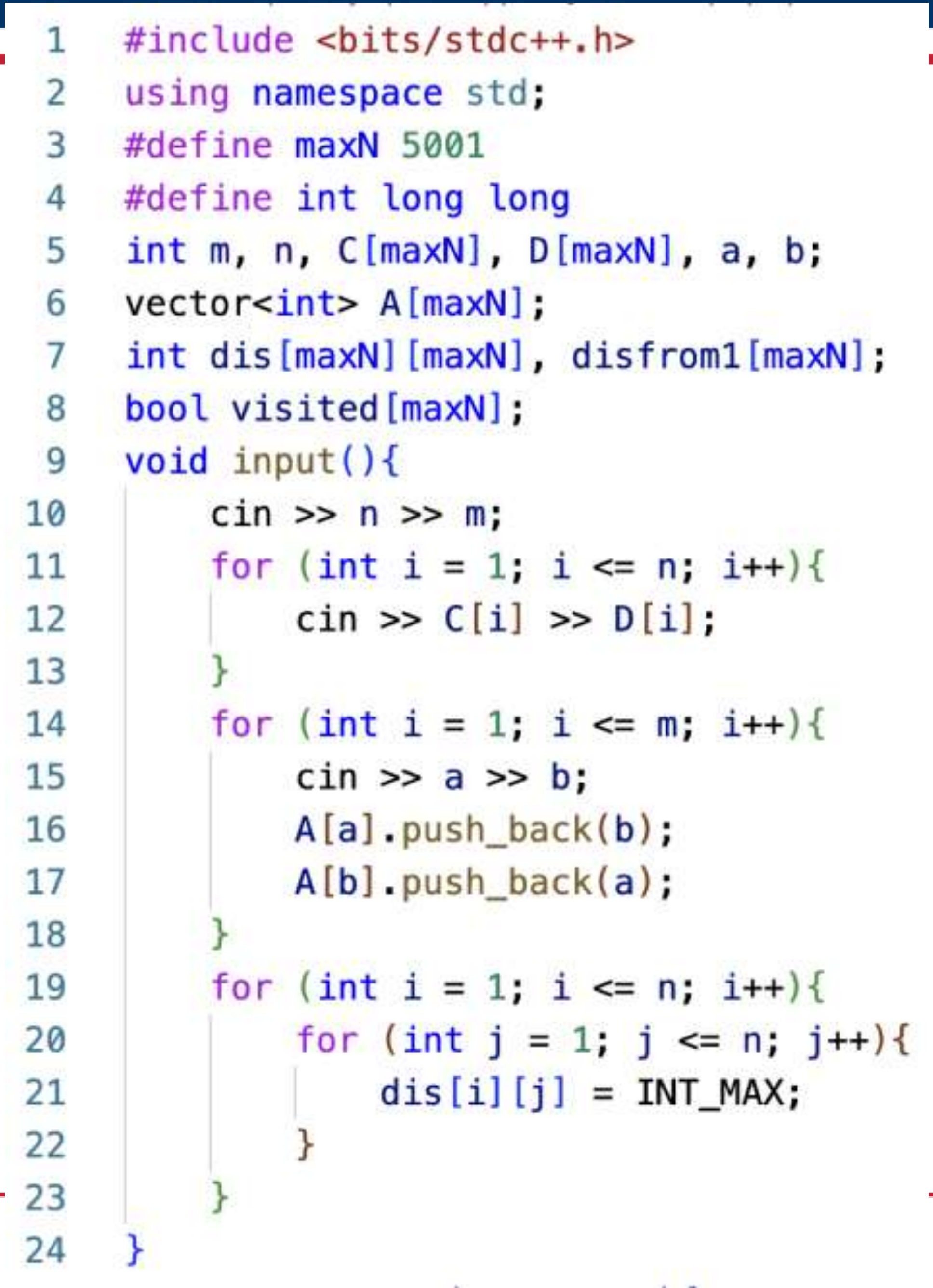
```
30
```

Explain: the path found from city 1 to city 6 with mimimum cost is
On bus from city 1 -> city 4 cost $10
On bus from city 4 -> city 6 cost $20
Total cost: $10 + $20 = $30

Bus inter-city

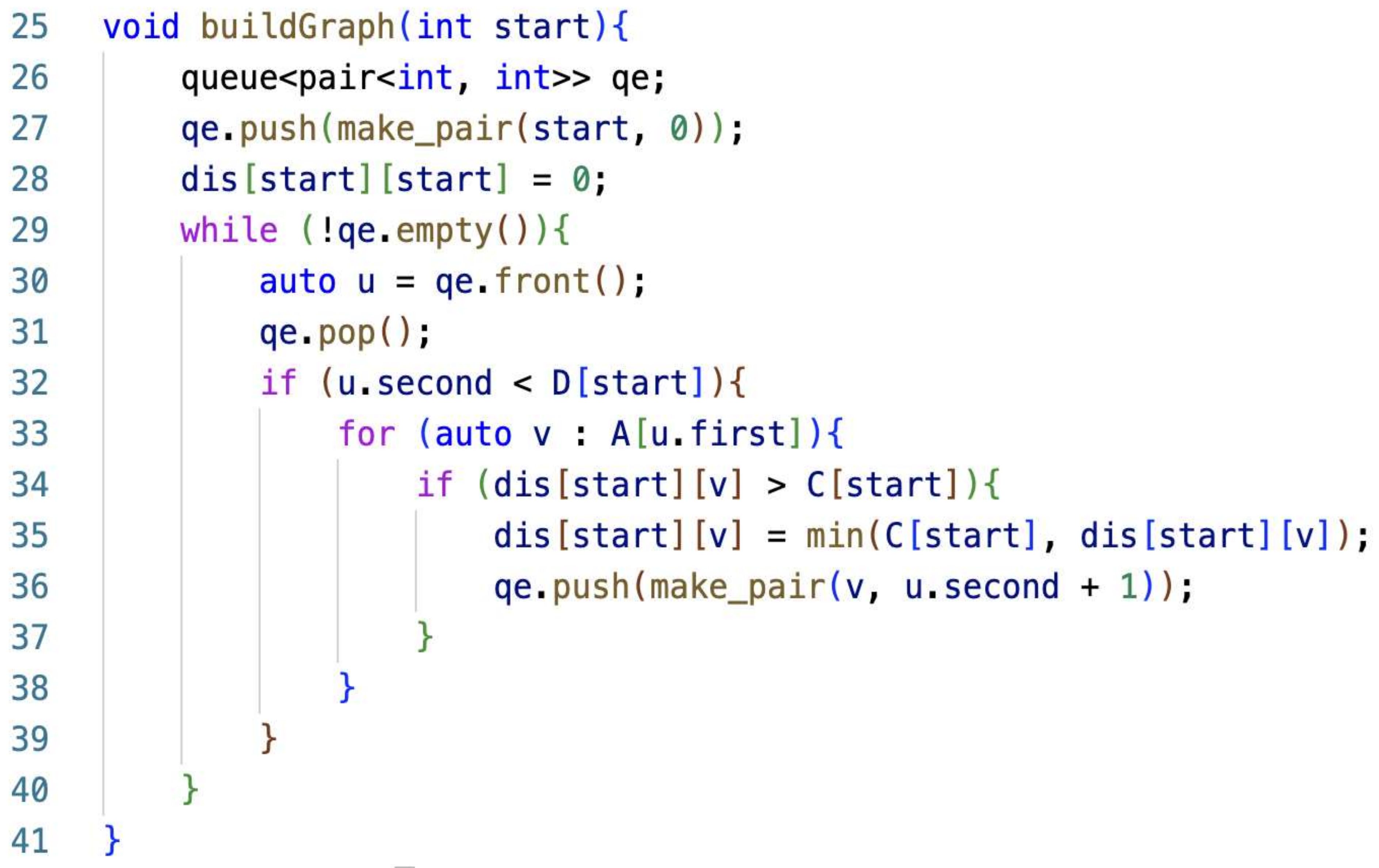
- Idea to solve: Use Dijkstra algorithm on a new cost graph
  - Build **cost graph**: A graph with same node set but edge (u,v) means we can go on bus from u to go to v, cost c(u,v) = C[u], dist[u][v] $\leq$ D[u]
    - Use BFS(u) with a depth limitation
  - Run Dijkstra from node 1 to find min cost path to city n

Bus inter-city - Implementation



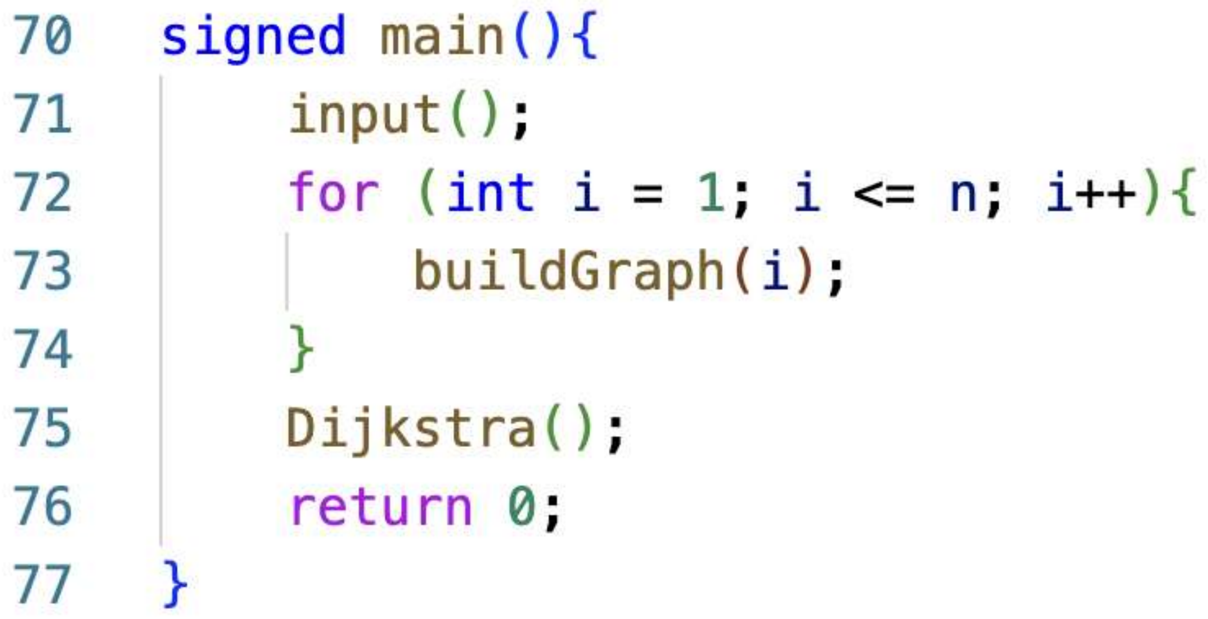
```cpp
#include <bits/stdc++.h>
using namespace std;
#define maxN 5001
#define int long long
int m, n, C[maxN], D[maxN], a, b;
vector<int> A[maxN];
int dis[maxN][maxN], disfrom1[maxN];
bool visited[maxN];
void input(){
    cin >> n >> m;
    for (int i = 1; i <= n; i++){
        cin >> C[i] >> D[i];
    }
    for (int i = 1; i <= m; i++){
        cin >> a >> b;
        A[a].push_back(b);
        A[b].push_back(a);
    }
    for (int i = 1; i <= n; i++){
        for (int j = 1; j <= n; j++){
            dis[i][j] = INT_MAX;
        }
    }
}
```

Bus inter-city - Implementation



```cpp
void buildGraph(int start){
    queue<pair<int, int>> qe;
    qe.push(make_pair(start, 0));
    dis[start][start] = 0;
    while (!qe.empty()){
        auto u = qe.front();
        qe.pop();
        if (u.second < D[start]){
            for (auto v : A[u.first]){
                if (dis[start][v] > C[start]){
                    dis[start][v] = min(C[start], dis[start][v]);
                    qe.push(make_pair(v, u.second + 1));
                }
            }
        }
    }
}
```

Bus inter-city - Implementation



```
signed main(){
    input();
    for (int i = 1; i <= n; i++){
        buildGraph(i);
    }
    Dijkstra();
    return 0;
}
```

THANK YOU !