```
#include <bits/stdc++.h>
using namespace std;
struct Data{
 int node, dist, cost;
 Data(int a, int b, int c){
 node = a;
 dist = b;
 cost = c;
 }
};
struct Comparator{
 bool operator() (Data a, Data b) {
 return !(a.cost < b.cost);
 }
};
class Solution {
public:
 vector<vector<int>> cost;
 int findCheapestPrice(int n, vector<vector<int>>& flights, int
src, int dst, int K) {
 cost = vector<vector<int> >(n + 1, vector<int>(K + 10,
INT_MAX));
 vector<vector<int> > graph[n];
 for (int i = 0; i < flights.size(); i++) {
 int u = flights[i][0];
 int v = flights[i][1];
 graph[u].push_back({ v, flights[i][2] });
 }
 priority_queue<Data, vector<Data>, Comparator> q;
 q.push(Data(src, 0, 0));
 cost[src][0] = 0;
 int ans = -1;
 while (!q.empty()) {
 Data temp = q.top();
 int curr = temp.node;
 q.pop();
 int dist = temp.dist;
 if (curr == dst)
 return temp.cost;
 dist++;
 if (dist > K + 1)
```

```
 continue;
 for (int i = 0; i < graph[curr].size(); i++) {
 int neighbour = graph[curr][i][0];
 if (cost[neighbour][dist] > cost[curr][dist - 1] +
graph[curr][i][1]) {
 cost[neighbour][dist] = cost[curr][dist - 1] +
graph[curr][i][1];
 q.push(Data(neighbour, dist,
cost[neighbour][dist]));
 }
 }
 }
 return -1;
 }
};
int main(){
 Solution ob;
 vector<vector<int>> v = {{0,1,100},{1,2,100},{0,2,500}};
 cout << (ob.findCheapestPrice(3, v, 0, 2, 1));
}
#include <bits/stdc++.h>
using namespace std;
struct Data{
 int node, dist, cost;
 Data(int a, int b, int c){
 node = a;
 dist = b;
 cost = c;
 }
};
struct Comparator{
 bool operator() (Data a, Data b) {
 return !(a.cost < b.cost);
 }
};
class Solution {
public:
 vector<vector<int>> cost;
 int findCheapestPrice(int n, vector<vector<int>>& flights, int
src, int dst, int K) {
 cost = vector<vector<int> >(n + 1, vector<int>(K + 10,
```

```
INT_MAX));
 vector<vector<int> > graph[n];
 for (int i = 0; i < flights.size(); i++) {
 int u = flights[i][0];
 int v = flights[i][1];
 graph[u].push_back({ v, flights[i][2] });
 }
 priority_queue<Data, vector<Data>, Comparator> q;
 q.push(Data(src, 0, 0));
 cost[src][0] = 0;
 int ans = -1;
 while (!q.empty()) {
 Data temp = q.top();
 int curr = temp.node;
 q.pop();
 int dist = temp.dist;
 if (curr == dst)
 return temp.cost;
 dist++;
 if (dist > K + 1)
 continue;
 for (int i = 0; i < graph[curr].size(); i++) {
 int neighbour = graph[curr][i][0];
 if (cost[neighbour][dist] > cost[curr][dist - 1] +
graph[curr][i][1]) {
 cost[neighbour][dist] = cost[curr][dist - 1] +
graph[curr][i][1];
 q.push(Data(neighbour, dist,
cost[neighbour][dist]));
 }
 }
 }
 return -1;
 }
};
int main(){
 Solution ob;
 vector<vector<int>> v = {{0,1,100},{1,2,100},{0,2,500}};
 cout << (ob.findCheapestPrice(3, v, 0, 2, 1));
}
#include <bits/stdc++.h>
```

```
using namespace std;
struct Data{
 int node, dist, cost;
 Data(int a, int b, int c){
 node = a;
 dist = b;
 cost = c;
 }
};
struct Comparator{
 bool operator() (Data a, Data b) {
 return !(a.cost < b.cost);
 }
};
class Solution {
public:
 vector<vector<int>> cost;
 int findCheapestPrice(int n, vector<vector<int>>& flights, int
src, int dst, int K) {
 cost = vector<vector<int> >(n + 1, vector<int>(K + 10,
INT_MAX));
 vector<vector<int> > graph[n];
 for (int i = 0; i < flights.size(); i++) {
 int u = flights[i][0];
 int v = flights[i][1];
 graph[u].push_back({ v, flights[i][2] });
 }
 priority_queue<Data, vector<Data>, Comparator> q;
 q.push(Data(src, 0, 0));
 cost[src][0] = 0;
 int ans = -1;
 while (!q.empty()) {
 Data temp = q.top();
 int curr = temp.node;
 q.pop();
 int dist = temp.dist;
 if (curr == dst)
 return temp.cost;
 dist++;
 if (dist > K + 1)
 continue;
```

```
 for (int i = 0; i < graph[curr].size(); i++) {
 int neighbour = graph[curr][i][0];
 if (cost[neighbour][dist] > cost[curr][dist - 1] +
graph[curr][i][1]) {
 cost[neighbour][dist] = cost[curr][dist - 1] +
graph[curr][i][1];
 q.push(Data(neighbour, dist,
cost[neighbour][dist]));
 }
 }
 }
 return -1;
 }
};
int main(){
 Solution ob;
 vector<vector<int>> v = {{0,1,100},{1,2,100},{0,2,500}};
 cout << (ob.findCheapestPrice(3, v, 0, 2, 1));
}
```