

Assignment 4

Problem Statement : Write a program to solve the travelling salesman problem and to print the path and the cost using LC Branch and Bound.

```
#include <bits/stdc++.h>
using namespace std;

struct Node {
    vector<int> path;
    vector<bool> visited;
    int cost, bound, vertex, level;
    bool operator>(const Node &o) const { return bound > o.bound; }
};

int n;
vector<vector<int>>> dist;
int best = INT_MAX;
vector<int> bestPath;

int boundCalc(Node &node) {
    int b = node.cost;
    for (int i = 0; i < n; i++) {
        if (!node.visited[i]) {
            int mn = INT_MAX;
            for (int j = 0; j < n; j++) {
                if (i != j && (!node.visited[j] || j == 0)) {
                    mn = min(mn, dist[i][j]);
                }
            }
            b += mn;
        }
    }
    return b;
}

void solveTSP() {
    priority_queue<Node, vector<Node>, greater<Node>> pq;
    Node root;
    root.path.push_back(0);
    root.visited.assign(n, false);
    root.visited[0] = true;
    root.cost = 0;
```

```

root.vertex = 0;
root.level = 1;
root.bound = boundCalc(root);
pq.push(root);

while (!pq.empty()) {
    Node cur = pq.top(); pq.pop();
    if (cur.bound >= best) continue;

    if (cur.level == n) {
        int finalCost = cur.cost + dist[cur.vertex][0];
        if (finalCost < best) {
            best = finalCost;
            bestPath = cur.path;
            bestPath.push_back(0);
        }
        continue;
    }

    for (int j = 0; j < n; j++) {
        if (!cur.visited[j]) {
            Node child = cur;
            child.path.push_back(j);
            child.cost += dist[cur.vertex][j];
            child.vertex = j;
            child.level++;
            child.visited[j] = true;
            child.bound = boundCalc(child);
            if (child.bound < best) pq.push(child);
        }
    }
}

int main() {
    cin >> n;
    dist.assign(n, vector<int>(n));
    for (int i = 0; i < n; i++)
        for (int j = 0; j < n; j++)
            cin >> dist[i][j];
    solveTSP();
    for (int x : bestPath) cout << x << " ";
    cout << "\n" << best << "\n";
}

```

Test Case 1:

PROBLEMS SUBMIT STATUS STANDINGS CUSTOM TEST

Source:

```
1 #include <bits/stdc++.h>
2 using namespace std;
3
4 struct Node {
5     vector<int> path;
6     vector<bool> visited;
7     int cost, bound, vertex, level;
8     bool operator>(const Node &o) const { return bound > o.bound; }
9 };
10
11 int n;
12 vector<vector<int>> dist;
13 int best = INT_MAX;
14 vector<int> bestPath;
15
16 int boundCalc(Node &node) {
17     int b = node.cost;
18     for (int i = 0; i < n; i++) {
19         if (!node.visited[i]) {
20             int mn = INT_MAX;
21             for (int j = 0; j < n; j++) {
22                 if (i != j && (!node.visited[j] || j == 0)) {
23                     mn = min(mn, dist[i][j]);
24                 }
25             }
26             b += mn;
27         }
28     }
29     return b;
30 }
```

☐ Switch off editor

Tab size: 4

Run

Language: GNU G++17 7.3.0

Input:
4
0 10 15 20
10 0 35 25
15 35 0 30
20 25 30 0

Choose File No file chosen
No more than 256 KB

Output:
0 1 3 2 0
80
=====
Used: 46 ms, 4 KB

First 255 bytes only

Test Case 2:

PROBLEMS SUBMIT STATUS STANDINGS CUSTOM TEST

Source:

```
1 #include <bits/stdc++.h>
2 using namespace std;
3
4 struct Node {
5     vector<int> path;
6     vector<bool> visited;
7     int cost, bound, vertex, level;
8     bool operator>(const Node &o) const { return bound > o.bound; }
9 };
10
11 int n;
12 vector<vector<int>> dist;
13 int best = INT_MAX;
14 vector<int> bestPath;
15
16 int boundCalc(Node &node) {
17     int b = node.cost;
18     for (int i = 0; i < n; i++) {
19         if (!node.visited[i]) {
20             int mn = INT_MAX;
21             for (int j = 0; j < n; j++) {
22                 if (i != j && (!node.visited[j] || j == 0)) {
23                     mn = min(mn, dist[i][j]);
24                 }
25             }
26             b += mn;
27         }
28     }
29     return b;
30 }
```

☐ Switch off editor

Tab size: 4

Run

Language: GNU G++17 7.3.0

Input:
5
0 14 4 10 20
14 0 7 8 7
4 5 0 7 16
11 7 9 0 2
18 7 17 4 0

Choose File No file chosen
No more than 256 KB

Output:
0 3 4 1 2 0
30
=====
Used: 46 ms, 0 KB

First 255 bytes only