kruskal.c++

```cpp
#include <iostream>

using namespace std;

const int INFI = 99;
int adj_matrix[10][10];

void update(int root[], int u, int v, int n) {
    int temp = root[v];
    for (int i = 1; i <= n; i++) {
        if (root[i] == temp)
            root[i] = root[u];
    }
}

void kruskal(int n) {
    int u, v, mst[n-1][2], root[n+1], cost = 0;

    for (int i = 1; i <= n; i++)
        root[i] = i;

    int i = 0;
    while (i != n - 1) {
        int min_edge = INFI;
        for (int i = 1; i <= n; i++) {
            for (int j = i + 1; j <= n; j++) {
                if (adj_matrix[i][j] < min_edge) {
                    min_edge = adj_matrix[i][j];
                    u = i;
                    v = j;
                }
            }
        }

        adj_matrix[u][v] = adj_matrix[v][u] = INFI;
        if (root[u] != root[v]) {
            mst[i][0] = u;
            mst[i][1] = v;
            cost += min_edge;
            update(root, u, v, n);
            i++;
        }
    }

    cout << "\nThe edges of minimum spanning tree:\n";
    for (int i = 0; i < n - 1; i++)
        cout << "(" << mst[i][0] << ", " << mst[i][1] << ")\n";
    cout << "The cost of minimum spanning tree: " << cost << endl;
}

int main() {
```

```
    int n;
    cout << "\nEnter the number of vertices: ";
    cin >> n;
    cout << "\nEnter the adjacency matrix: ";
    for (int i = 1; i <= n; i++)
        for (int j = 1; j <= n; j++)
            cin >> adj_matrix[i][j];
    kruskal(n);
    return 0;
}
```