

# 最小生成树

2153726 罗宇翔

(经济与管理学院, 信息管理与信息系统)

# 目录

<b>1</b>	<b>实验目的</b>	<b>1</b>
<b>2</b>	<b>实验内容</b>	<b>1</b>
<b>3</b>	<b>实验环境</b>	<b>1</b>
3.1	Visual Studio Code . . . . .	1
3.2	g++ . . . . .	1
<b>4</b>	<b>实验原理和方法</b>	<b>1</b>
4.1	Kruskal 算法 . . . . .	1
4.2	并查集 . . . . .	2
4.2.1	查询 . . . . .	2
4.2.2	合并 . . . . .	3
<b>5</b>	<b>实验代码</b>	<b>3</b>
<b>6</b>	<b>实验数据及结果分析</b>	<b>8</b>

---

# 1 实验目的

使用  $C++$  实现寻找赋权图的最小生成树。

# 2 实验内容

使用 *Kruskal* 算法和并查集数据结构实现最小生成树。

# 3 实验环境

## 3.1 Visual Studio Code

Version: 1.89.1

Browser:

Mozilla/5.0 (Macintosh; Intel Mac OS X 10\_15\_7)

AppleWebKit/537.36 (KHTML, like Gecko) Code/1.89.1

Chrome/120.0.6099.291 Electron/28.2.8 Safari/537.36

## 3.2 g++

Apple clang version 14.0.0 (clang-1400.0.29.202)

Target: x86\_64-apple-darwin21.6.0

Thread model: posix

# 4 实验原理和方法

## 4.1 Kruskal 算法

此算法的核心思想是：

- (1) 假设该图  $G$  是不连通的，对该图的边以权值非降序重新排列
- (2) 对于排序表中的每条边，如果现在把它放入最小生成树  $T$  不会形成回路的话，则把它加入到  $T$  中；否则丢弃

## 4.2 并查集

并查集是一种储存集合的数据结构，它以一颗  $n$  叉树作为一个集合，树中的节点表示对应集合中的元素。

并查集支持两种操作：

查询 (*Find*): 查询某个元素所属集合 (查询对应的树的根节点)，这可以用于判断两个元素是否属于同一集合

合并 (*Unite*): 合并两个元素所属集合 (合并对应的树)

在本问题中，使用并查集储存已联通的点，避免生成回路。

### 4.2.1 查询

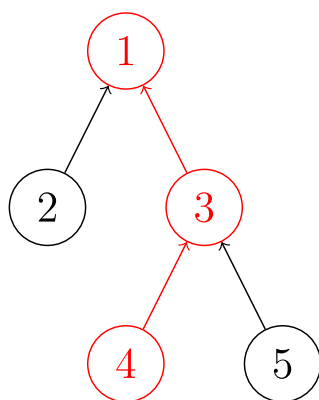


Figure 1: 查询过程

对于一个元素，例如 4，查询其所在的集合，只需要查询其根结点，通过递归地向父结点移动即可。

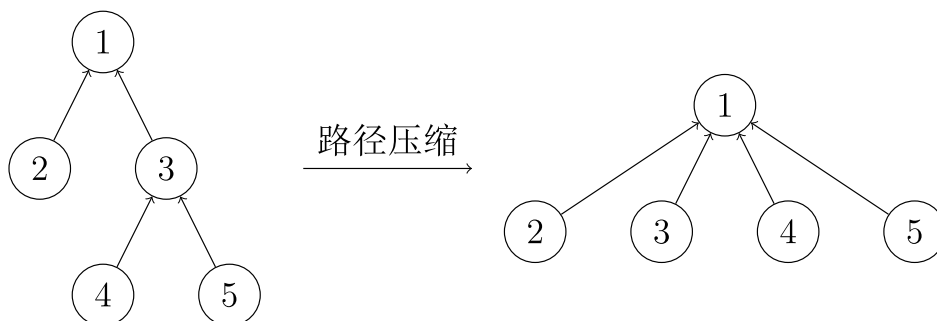


Figure 2: 压缩过程

查询过程中，为了下次查询方便，可将树的深度调整为 1。

### 4.2.2 合并

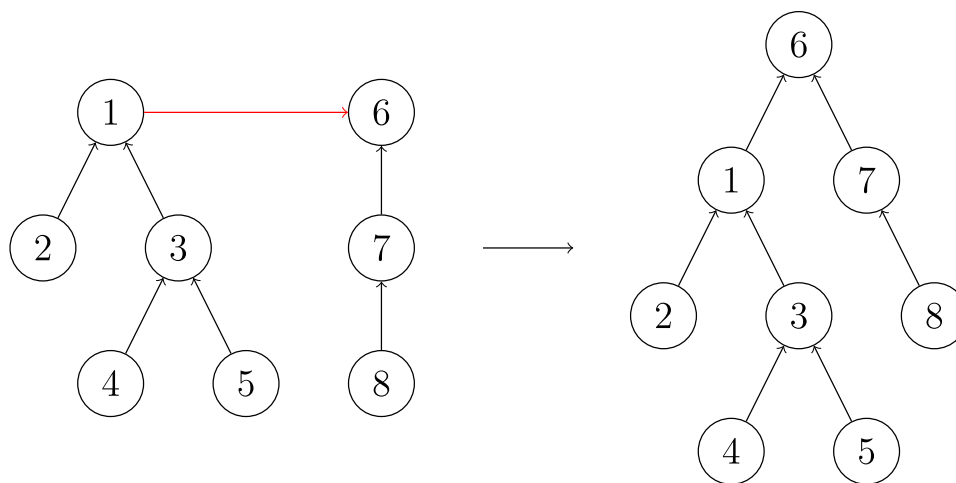


Figure 3: 合并过程

对于两个集合，若要合并，只需要让其中一个集合的根结点作为另一个集合根结点的子结点即可。为了降低下次查找的次数，可令结点数较少的树向较多的树合并。

## 5 实验代码

```
#include <algorithm>
#include <iostream>
#include <sstream>
#include <vector>

using namespace std;

struct Edge { // define Edge structure
    int src, dest, weight;
};

bool compare_edges(const Edge &a, const Edge &b) { // define
    comparison function for edge's weight
    return a.weight < b.weight;
```

---

```
}
```

```
class UnionFind { // Disjoint-set
private:
    vector< int > parent, rank;

public:
    UnionFind(int n) {
        parent.resize(n);
        rank.resize(n);
        for (int i = 0; i < n; ++i) {
            parent[i] = i;
            rank[i] = 0;
        }
    }

    int find(int x) {
        if (parent[x] != x) {
            parent[x] = find(parent[x]);
        }
        return parent[x];
    }

    void unite(int x, int y) {
        int rootX = find(x);
        int rootY = find(y);
        if (rootX == rootY) {
            return;
        }
        if (rank[rootX] < rank[rootY]) {
            parent[rootX] = rootY;
        } else if (rank[rootX] > rank[rootY]) {
```

---

```

        parent[rootY] = rootX;
    } else {
        parent[rootY] = rootX;
        rank[rootX]++;
    }
}
};

vector< Edge > kruskal(vector< Edge > &edges, int V) {
    sort(edges.begin(), edges.end(), compare_edges); // ascend
    sort
    vector< Edge > result;
    UnionFind uf(V);
    for (int i = 0; i < edges.size(); i++) {
        int src = edges[i].src;
        int dest = edges[i].dest;
        if (uf.find(src) != uf.find(dest)) {
            result.push_back(edges[i]);
            uf.unite(src, dest);
        }
    }
    return result;
}

int main() {
    while (true) {
        string input;
        int V, E;
        cout << "Enter number of vertices and edges(enter 0 0
to quit):\n";
        getline(cin, input);
        stringstream ss(input);

```

---

```

    if (!(ss >> V >> E)) {
        cout << "Invalid input\n";
        continue;
    }
    char extra;
    if (ss >> extra) {
        cout << "Exist extra character\n";
        continue;
    }
    if (V == 0 && E == 0) {
        cout << "Thanks for using\n";
        break;
    }
    if (V < 0 || E < 0) {
        cout << "Invalid input\n";
        continue;
    }
    vector< Edge > edges(E);
    int i = 0;
    while (i < E) {
        printf("Enter source, destination, and weight for
each edge(%d/%d):\n", i + 1, E);
        string input;
        getline(cin, input);
        stringstream ss(input);
        if (!(ss >> edges[i].src >> edges[i].dest >> edges[
i].weight)) {
            cout << "Invalid input\n";
            continue;
        }
        char extra;
        if (ss >> extra) {

```



---

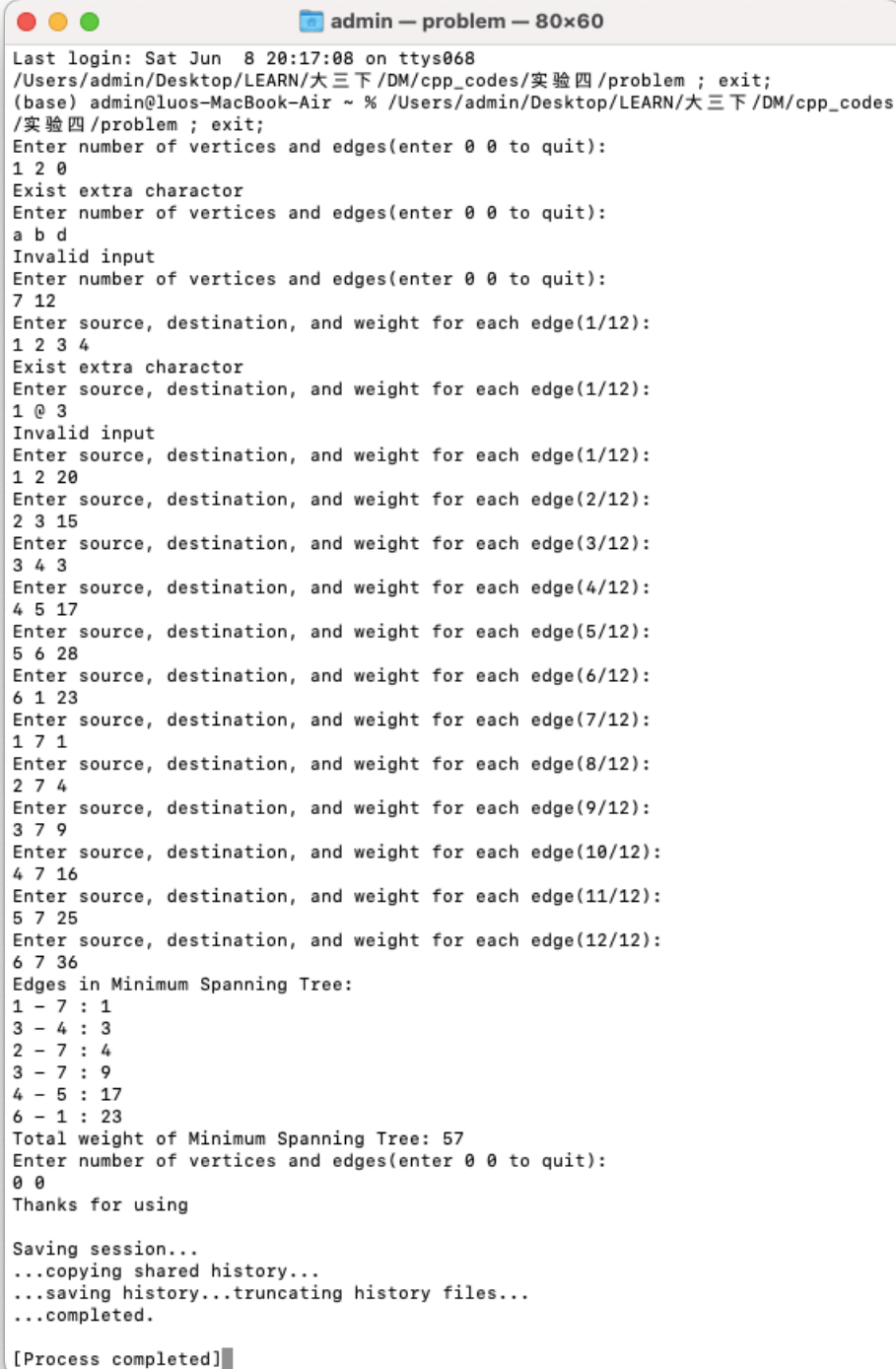
```
        cout << "Exist extra character\n";
        continue;
    }
    i++;
}

vector< Edge > minSpanningTree = kruskal(edges, V);

cout << "Edges in Minimum Spanning Tree:\n";
int totalWeight = 0;
Edge edge;
for (int i = 0; i < minSpanningTree.size(); i++) {
    edge = minSpanningTree[i];
    cout << edge.src << " - " << edge.dest << " : " <<
edge.weight << "\n";
    totalWeight += edge.weight;
}
cout << "Total weight of Minimum Spanning Tree: " <<
totalWeight << "\n";
}

return 0;
}
```

## 6 实验数据及结果分析



```
admin — problem — 80x60
Last login: Sat Jun  8 20:17:08 on ttys068
/Users/admin/Desktop/LEARN/大三下/DM/cpp_codes/实验四/problem ; exit;
(base) admin@luos-MacBook-Air ~ % /Users/admin/Desktop/LEARN/大三下/DM/cpp_codes
/实验四/problem ; exit;
Enter number of vertices and edges(enter 0 0 to quit):
1 2 0
Exist extra charactor
Enter number of vertices and edges(enter 0 0 to quit):
a b d
Invalid input
Enter number of vertices and edges(enter 0 0 to quit):
7 12
Enter source, destination, and weight for each edge(1/12):
1 2 3 4
Exist extra charactor
Enter source, destination, and weight for each edge(1/12):
1 @ 3
Invalid input
Enter source, destination, and weight for each edge(1/12):
1 2 20
Enter source, destination, and weight for each edge(2/12):
2 3 15
Enter source, destination, and weight for each edge(3/12):
3 4 3
Enter source, destination, and weight for each edge(4/12):
4 5 17
Enter source, destination, and weight for each edge(5/12):
5 6 28
Enter source, destination, and weight for each edge(6/12):
6 1 23
Enter source, destination, and weight for each edge(7/12):
1 7 1
Enter source, destination, and weight for each edge(8/12):
2 7 4
Enter source, destination, and weight for each edge(9/12):
3 7 9
Enter source, destination, and weight for each edge(10/12):
4 7 16
Enter source, destination, and weight for each edge(11/12):
5 7 25
Enter source, destination, and weight for each edge(12/12):
6 7 36
Edges in Minimum Spanning Tree:
1 - 7 : 1
3 - 4 : 3
2 - 7 : 4
3 - 7 : 9
4 - 5 : 17
6 - 1 : 23
Total weight of Minimum Spanning Tree: 57
Enter number of vertices and edges(enter 0 0 to quit):
0 0
Thanks for using

Saving session...
...copying shared history...
...saving history...truncating history files...
...completed.

[Process completed]
```

Figure 4: 运行结果

---

当输入非法时，程序会给出提示且不会崩溃。

见结点和边个数的输入以及第一个边的输入，当有多余字符和非法字符时程序分别给出 *Exist extra character* 和 *Invalid input* 的提示。