

## Contents

1	basic	1
1.1	binarySearch	1
1.2	stringstream	1
2	STL	1
2.1	map	1
3	graph	1
3.1	並查集	1
3.2	最小生成樹	2
3.3	最長共同子序列-LCS	2
3.4	Floyd-Warshall	3
4	math	3
4.1	質數表	3
4.2	gcd-lcm	3

## 1 basic

### 1.1 binarySearch

```

1 #include <bits/stdc++.h>
2 using namespace std;
3
4 int binary_search(const vector<int> &data, int key)
5 {
6     int low = 0;
7     int high = data.size()-1;
8     while (low <= high)
9     {
10         int mid = int((low + high) / 2);
11         if (key == data[mid])
12             return mid;
13         else if (key > data[mid])
14             low = mid + 1;
15         else
16             high = mid - 1;
17     }
18     return -1;
19 }
20
21 int main()
22 {
23     vector<int> data = {1, 9, 2, 7, 4, 10, 3, 8, 5,
24                         6};
25     int key = 7;
26
27     sort(data.begin(), data.end());
28
29     int ret = binary_search(data, key);
30     if (ret == -1)
31         cout << "找不到\n";
32     else
33         cout << "找到索引值" << ret << "\n";
34     //lower_bound(a, a + n, k);    //最左邊 ≥ k 的位置
35     //upper_bound(a, a + n, k);    //最左邊 > k 的位置
36     //upper_bound(a, a + n, k) - 1; //最右邊 ≤ k 的位置
37     //lower_bound(a, a + n, k) - 1; //最右邊 < k 的位置
38     //[lower_bound, upper_bound) //等於 k 的範圍
39     //equal_range(a, a+n, k);
40 }

```

### 1.2 stringstream

```

1 #include <sstream>
2 using namespace std;
3
4 int main()
5 {
6     stringstream ss;
7     int num = 1234;

```

```

8     string output;
9
10    ss << num;
11    ss >> output; //integer to string
12
13    string_to_int << ss;
14    string_to_int >> num; //string to integer
15
16    ss.str("");
17    ss.clear(); //initialization
18
19    return 0;
20 }

```

## 2 STL

### 2.1 map

```

1 #include <bits/stdc++.h> // #include <map>
2 using namespace std;
3
4 int main()
5 {
6     map<int, string> m = {{1, "Tom"}};
7     m.insert(pair<int, string>(7, "Jack"));
8     m[15] = "John";
9     for (const auto& s : m) // map<int,
10         // string>::iterator it = m.begin()
11     {
12         cout << s.first << " " << s.second << endl;
13     }
14     m.erase(7);
15     for (auto it = m.rbegin(); it!= m.rend(); it++)
16     {
17         cout << it->first << " " << it->second << endl;
18     }
19     m.clear();
20     return 0;

```

## 3 graph

### 3.1 並查集

```

1 #include <bits/stdc++.h>
2 using namespace std;
3
4 #define N 10
5 int p[N], rank[N]; // rank -> tree height, sz ->
6 // group size
7 void init()
8 {
9     for (int i = 0; i < N; i++)
10     {
11         p[i] = i;
12         rank[i] = 1;
13     }
14 }
15 int Find(int x)
16 {
17     if (x == p[x])
18         return x;
19     return p[x] = Find(p[x]);
20 }
21 void Union(int a, int b) // tree height
22 {
23     a = Find(a);
24     b = Find(b);
25     if (a == b)
26         return;
27     if (rank[a] < rank[b])

```

```

27     p[a] = b;
28     else if (rank[a] > rank[b])
29         p[b] = a;
30     else
31     {
32         p[a] = b;
33         rank[a]++;
34     }
35 }
36 // void Union(int a, int b) // group size
37 // {
38 //     a = Find(a);
39 //     b = Find(b);
40 //     if (a == b)
41 //         return;
42 //     if (sz[a] < sz[b])
43 //         swap(a, b);
44 //     sz[a] += sz[b];
45 //     p[b] = a;
46 // }
47
48 int main()
49 {
50     init();
51     for (int i = 0; i < N; i++)
52     {
53         int a, b;
54         cin >> a >> b;
55         Union(a, b);
56     }
57     return 0;
58 }

```

### 3.2 最小生成樹

```

1 #include <bits/stdc++.h>
2 using namespace std;
3
4 #define N 10
5 int p[N], sz[N];
6 struct Edge
7 {
8     int s, t, w;
9     Edge(int s, int t, int w) : s(s), t(t), w(w) {}
10    bool operator < (const Edge &rhs) const { return
11        w < rhs.w; }
12 };
13 void init()
14 {
15     for (int i = 1; i <= N; i++)
16     {
17         p[i] = i;
18         sz[i] = 1;
19     }
20 }
21 int Find(int x)
22 {
23     if (x == p[x])
24         return x;
25     return p[x] = Find(p[x]);
26 }
27 void Union(int a, int b) // group size
28 {
29     a = Find(a);
30     b = Find(b);
31     if (a == b)
32         return;
33     if (sz[a] < sz[b])
34         swap(a, b);
35     sz[a] += sz[b];
36     p[b] = a;
37 }
38 int kruskal()
39 {
40     int cost = 0;
41     vector<Edge> E;

```

```

41     init();
42     for (int i = 0; i < N; i++)
43     {
44         int s, t, w;
45         cin >> s >> t >> w;
46         E.push_back(Edge(s, t, w));
47     }
48     sort(E.begin(), E.end());
49     for (auto it : E)
50     {
51         it.s = Find(it.s);
52         it.t = Find(it.t);
53         if (it.s == it.t)
54             continue;
55         cost += it.w;
56         Union(it.s, it.t);
57     }
58     return cost;
59 }
60
61 int main()
62 {
63     init();
64     int cost = kruskal();
65     cout << cost << endl;
66     return 0;
67 }

```

### 3.3 最長共同子序列-LCS

```

1 #include <bits/stdc++.h>
2 using namespace std;
3
4 #define N 120
5 string strA, strB;
6 int t[N*N], d[N*N], num[N*N]; //t and d 是 LIS
7 // 要用到
8 // d 用來記住 LIS 中此數字的前一個數字
9 // t 當前 LIS 的數列位置
10 // num 則是我們根據 strB 的字元生成數列，用來找出最長
11 // LIS 長度
12 map<char, vector<int>> dict; //記住每個字串出現的
13 // index 位置
14
15 int bs(int l, int r, int v) { //binary search
16     int m;
17     while(r > l) {
18         m = (l+r) / 2;
19         if(num[v] > num[t[m]]) l = m+1;
20         else if (num[v] < num[t[m]]) r = m;
21         else return m;
22     }
23     return r;
24 }
25
26 int lcs() {
27     dict.clear(); //先將 dict 先清空
28     for(int i = strA.length()-1; i >= 0; i--)
29         dict[strA[i]].push_back(i);
30     // 將每個字串的位置紀錄並放入 vector 中，請記住 i
31     // = strA.length() - 1 才可以達到逆序效果
32
33     int k = 0; //紀錄生成數列的長度的最長長度
34     for(int i = 0; i < strB.length(); i++) { // 依據
35         strB 的每個字元來生成數列
36         for(int j = 0; j < dict[strB[i]].size();
37             j++)
38             //將此字元在 strA 出現的位置放入數列
39             num[i+k] = dict[strB[i]][j];
40     }
41     if(k==0) return 0; //如果 k = 0
42     //就表示他們沒有共同字元都沒有於是就直接輸出 0
43
44     d[1] = -1, t[1] = 1; //LIS init

```

```

37 |     int len = 1, cur ; // len 由於前面已經把 LCS = 0
    |     的機會排除，於是這裡則從 1 開始
38 |
39 |     // 標準的 LIS 作法，不斷嘗試將 LCS 生長
40 |     for(int i = 1 ; i <= k ; i++){
41 |         if(num[i] > num[t[len]]) t[++len] = i , d[i]
    |         = t[len-1] ;
42 |         else{
43 |             cur = bs(1,len,i);
44 |             t[cur] = i ;
45 |             d[i] = t[cur-1];
46 |         }
47 |
48 |         //debug
49 |         // for(int i = 1 ; i <= k ; i++){
50 |         //     cout << num[t[i]] << ' ' ;
51 |         //     cout << '\n' ;
52 |     }
53 |     return len ;
54 |
55 | }
56 |
57 | int main()
58 | {
59 |     getline(cin, strA);
60 |     getline(cin, strB);
61 |     cout << lcs() << endl;
62 |     return 0;
63 | }

```

### 3.4 Floyd-Warshall

```

1 | #include <bits/stdc++.h>
2 | using namespace std;
3 |
4 | #define N 10+5
5 |
6 | int main()
7 | {
8 |     int G[N][N];
9 |     for (int i = 0; i < N; i++)
10 |     {
11 |         for (int j = 0; j < N; j++)
12 |         {
13 |             cin >> G[i][j];
14 |             G[j][i] = G[i][j];
15 |         }
16 |     }
17 |     for (k = 0; k < n; k++)
18 |         for (i = 0; i < n; i++)
19 |             for (j = 0; j < n; j++)
20 |                 w[i][j] = w[j][i] = min(w[i][j],
    |                 w[i][k] + w[k][j]);
21 |     return 0;
22 | }

```

## 4 math

### 4.1 質數表

```

1 | #include <bits/stdc++.h>
2 | using namespace std;
3 |
4 | #define maxn 100+5
5 | vector<int> p;
6 | bitset<maxn> is_notp;
7 | void PrimeTable(int n)
8 | {
9 |     is_notp.reset();
10 |    is_notp[0] = is_notp[1] = 1;
11 |    for (int i = 2; i <= n; i++)
12 |    {

```

```

13 |        if (is_notp[i])
14 |            continue;
15 |        p.push_back(i);
16 |        for (int j = i * i; j <= n; j += i)
17 |        {
18 |            is_notp[j] = 1;
19 |        }
20 |    }
21 | }
22 |
23 | int main()
24 | {
25 |     PrimeTable(100);
26 |     cout << is_notp[3];
27 |     return 0;
28 | }

```

### 4.2 gcd-lcm

```

1 | #include <bits/stdc++.h>
2 | using namespace std;
3 |
4 | int gcd(int a, int b)
5 | {
6 |     if (b == 0) return a;
7 |     return gcd(b, a%b);
8 | }
9 |
10 | int lcm(int a, int b)
11 | {
12 |     if (a < b) swap(a, b);
13 |     return a / gcd(a, b) * a;
14 | }
15 |
16 | int main()
17 | {
18 |     int a, b;
19 |     cin >> a >> b;
20 |     cout << gcd(a, b) << endl << lcm(a, b) << endl;
21 |     return 0;
22 | }

```