

Contents

1	basic	1
1.1	binarySearch	1
1.2	stringstream	1
2	graph	1
2.1	並查集	1
2.2	最小生成樹	2
2.3	最長共同子序列-LCS	2
2.4	Floyd-Warshall	3
3	math	3
3.1	質數表	3
3.2	gcd-lcm	3
4	Section1	3
4.1	basic	3
5	Section2	3
5.1	thm	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
35     //lower_bound(a, a + n, k); //最左邊 ≥ k 的位置
36     //upper_bound(a, a + n, k); //最左邊 > k 的位置
37     //upper_bound(a, a + n, k) - 1; //最右邊 ≤ k 的位置
38     //lower_bound(a, a + n, k) - 1; //最右邊 < k 的位置
39     //[lower_bound, upper_bound) //等於 k 的範圍
40     //equal_range(a, a+n, k);
41 }

```

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 graph

2.1 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])
28         p[a] = b;
29     else if (rank[a] > rank[b])
30         p[b] = a;
31     else
32     {
33         p[a] = b;
34         rank[a]++;
35     }
36 }
37 // void Union(int a, int b) // group size
38 // {
39 //     a = Find(a);
40 //     b = Find(b);
41 //     if (a == b)
42 //         return;
43 //     if (sz[a] < sz[b])
44 //         swap(a, b);
45 //     sz[a] += sz[b];
46 //     p[b] = a;
47 // }
48
49 int main()
50 {
51     init();
52     for (int i = 0; i < N; i++)
53     {
54         int a, b;

```

```

54     cin >> a >> b;
55     Union(a, b);
56 }
57 return 0;
58 }

```

2.2 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;
42     init();
43     for (int i = 0; i < N; i++)
44     {
45         int s, t, w;
46         cin >> s >> t >> w;
47         E.push_back(Edge(s, t, w));
48     }
49     sort(E.begin(), E.end());
50     for (auto it : E)
51     {
52         it.s = Find(it.s);
53         it.t = Find(it.t);
54         if (it.s == it.t)
55             continue;
56         cost += it.w;
57         Union(it.s, it.t);
58     }
59     return cost;
60 }
61 int main()
62 {
63     init();
64     int cost = kruskal();
65     cout << cost << endl;
66     return 0;
67 }

```

2.3 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 // d 用來記住 LIS 中此數字的前一個數字
8 // t 當前 LIS 的數列位置
9 // num 則是我們根據 strB 的字元生成數列，用來找出最長
10 // LIS 長度
11 map<char, vector<int>> dict; //記住每個字串出現的
12 // index 位置
13
14 int bs(int l, int r, int v) { //binary search
15     int m;
16     while(r>l){
17         m = (l+r) / 2;
18         if(num[v] > num[t[m]]) l = m+1;
19         else if (num[v] < num[t[m]]) r = m;
20         else return m;
21     }
22     return r;
23 }
24 int lcs(){
25     dict.clear(); //先將 dict 先清空
26     for(int i = strA.length()-1; i >= 0; i--){
27         dict[strA[i]].push_back(i);
28     }
29     // 將每個字串的位置紀錄並放入 vector 中，請記住 i
30     // = strA.length() - 1 才可以達到逆序效果
31
32     int k = 0; //紀錄生成數列的長度的最長長度
33     for(int i = 0; i < strB.length(); i++){ // 依據
34         strB 的每個字元來生成數列
35         for(int j = 0; j < dict[strB[i]].size();
36             j++){
37             //將此字元在 strA 出現的位置放入數列
38             num[++k] = dict[strB[i]][j];
39         }
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
45     int len = 1, cur; // len 由於前面已經把 LCS = 0
46     //的機會排除，於是這裡則從 1 開始
47
48     // 標準的 LIS 作法，不斷嘗試將 LCS 生長
49     for(int i = 1; i <= k; i++){
50         if(num[i] > num[t[len]]) t[++len] = i, d[i]
51             = t[len-1];
52         else{
53             cur = bs(1, len, i);
54             t[cur] = i;
55             d[i] = t[cur-1];
56         }
57     }
58
59     //debug
60     // for(int i = 1; i <= k; i++){
61     //     cout << num[t[i]] << ' ';
62     //     cout << '\n';
63     // }
64     return len;
65 }
66 int main()
67 {
68     getline(cin, strA);
69     getline(cin, strB);
70     cout << lcs() << endl;
71     return 0;
72 }

```

2.4 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],
21                                         w[i][k] + w[k][j]);
22     return 0;
23 }
```

3 math

3.1 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 }
```

3.2 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 }
```

4 Section1

4.1 basic

```

1 // c++ code
2 #include <bits/stdc++.h>
3 using namespace std;
4
5 int main() {
6     // test comment
7     cout << "test string\n";
8 }
```

5 Section2

5.1 thm

- 中文測試

$$\sum_{i=1}^n i^2 = \frac{n(n+1)(2n+1)}{6}$$