

Contents

1	basic	9
1.1	binarySearch	13
1.2	stringstream	14
2	graph	15
2.1	並查集	16
2.2	最小生成樹	17
2.3	最長共同子序列-LCS	18
2.4	Floyd-Warshall	19
3	Section1	20
3.1	basic	3
4	Section2	3
4.1	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);

```

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;
55         cin >> a >> b;
56         Union(a, b);

```

```
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
    w < rhs.w; }
11 | };
12 | void init()
13 | {
14 |     for (int i = 1; i <= N; i++)
15 |     {
16 |         p[i] = i;
17 |         sz[i] = 1;
18 |     }
19 | }
20 | int Find(int x)
21 | {
22 |     if (x == p[x])
23 |         return x;
24 |     return p[x] = Find(p[x]);
25 | }
26 | void Union(int a, int b) // group size
27 | {
28 |     a = Find(a);
29 |     b = Find(b);
30 |     if (a == b)
31 |         return;
32 |     if (sz[a] < sz[b])
33 |         swap(a, b);
34 |     sz[a] += sz[b];
35 |     p[b] = a;
36 | }
37 | int kruskal()
38 | {
39 |     int cost = 0;
40 |     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 | }
```

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
    要用到
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
    LIS 長度
10 | map<char, vector<int>> dict ; //記住每個字串出現的
    index 位置
11 |
12 | int bs(int l , int r , int v ){ //binary search
13 |     int m ;
14 |     while(r>l){
15 |         m = (l+r) /2 ;
16 |         if(num[v] > num[t[m]]) l = m+1 ;
17 |         else if (num[v] < num[t[m]]) r = m ;
18 |         else return m ;
19 |     }
20 |     return r ;
21 | }
22 |
23 | int lcs(){
24 |     dict.clear() ; //先將 dict 先清空
25 |     for(int i = strA.length()-1 ; i >= 0 ; i--){
26 |         dict[strA[i]].push_back(i) ;
27 |         // 將每個字串的位置紀錄並放入 vector 中，請記住 i
            = strA.length() -1 才可以達到逆續效果
28 |
29 |         int k = 0 ; //紀錄生成數列的長度的最長長度
30 |         for(int i = 0 ; i < strB.length() ; i++){ // 依據
            strB 的每個字元來生成數列
31 |             for(int j = 0 ; j < dict[strB[i]].size() ;
32 |                 j++){
33 |                 //將此字元在 strA 出現的位置放入數列
34 |                 num[++k] = dict[strB[i]][j] ;
35 |             }
36 |             if(k==0) return 0 ; //如果 k = 0
                就表示他們沒有共同字元都沒有於是就直接輸出 0
37 |
38 |             d[1] = -1 , t[1] = 1 ; //LIS init
39 |             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
40 |
41 |             // 標準的 LIS 作法，不斷嘗試將 LCS 生長
42 |             for(int i = 1 ; i <= k ; i++){
43 |                 if(num[i] > num[t[len]]) t[++len] = i , d[i]
                    = t[len-1] ;
44 |                 else{
45 |                     cur = bs(1,len,i);
46 |                     t[cur] = i ;
47 |                     d[i] = t[cur-1];
48 |                 }
49 |                 //debug
50 |                 // for(int i = 1 ; i <= k ; i++){
51 |                 //     cout << num[t[i]] << ' ' ;
52 |                 //     cout << '\n' ;
53 |             }
54 |             return len ;
55 | }
56 |
57 | int main()
58 | {
59 |     getline(cin, strA);
60 |     getline(cin, strB);
61 |     cout << lcs() << endl;
62 |     return 0;
63 | }
```

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 Section1

3.1 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 }
```

4 Section2

4.1 thm

- 中文測試

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