

Contents

1	基礎	1
1.1	binarySearch	1
1.2	stringstream	1
2	STL	1
2.1	map	1
3	圖論	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	數學	3
4.1	質數表	3
4.2	gcd-lcm	3
5	anngood	3
5.1	bfs	3
5.2	bwt	4

1 基礎

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;
21 }

```

3 圖論

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 }

```

```

24     if (a == b)
25         return;
26     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()

```

```

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 }

```

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
31     // 將每個字串的位置紀錄並放入 vector 中，請記住 i
32     // = strA.length() - 1 才可以達到逆續效果
33
34     int k = 0; //紀錄生成數列的長度的最長長度
35     for(int i = 0; i < strB.length(); i++) { // 依據
36         // strB 的每個字元來生成數列
37         for(int j = 0; j < dict[strB[i]].size();
38             j++)
39             //將此字元在 strA 出現的位置放入數列
40             num[++k] = dict[strB[i]][j];
41     }
42 }

```

```

34 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
35
36 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 數學

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) * b;
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 }

```

5 anngood

5.1 bfs

```

1 #include<bits/stdc++.h>
2 using namespace std;
3 int a[500][500]={0};
4 int used[500][500] = {0};
5
6 int moveI[4]={-1, 0, 1, 0};
7 int moveJ[4]={0, 1, 0, -1};
8 struct node{
9     int I, J, now;
10    node(int I, int J, int now):I(I), J(J),
        now(now){};
11 };
12 int main(){
13     int N;
14     cin>>N;
15     while(N--){
16         memset(a, 0, sizeof(a));
17         memset(used, 0, sizeof(used));
18         int n, m, sn, sm, en, em;
19         cin>>n>>m>>sn>>sm>>en>>em;
20

```

```

21     for(int i=0; i < n+2; i++){ //建牆壁
22         若不用可跳過
23         a[i][0] = 1;
24         a[i][m+1] = 1;
25     }
26     for(int i=0; i < m+2; i++){
27         a[0][i] = 1;
28         a[n+1][i] = 1;
29     }
30     for(int i=1; i<=n; i++){ // 用char輸入
31         因為輸入無空格
32         for(int j=1; j<=m; j++){
33             char t;
34             cin >> t;
35             a[i][j] = t - '0';
36         }
37     }
38     queue<node> q;
39     q.push(node(sn, sm, 1)); //定義起點為1步
40     int flag = 0;
41     while(!q.empty()){
42         node F=q.front(); //取最優先那個
43         q.pop(); //取完丟掉
44         if(F.I == en && F.J == em){
45             flag = 1;
46             cout << F.now << '\n';
47             break;
48         }
49         for(int i = 0; i < 4; i++){
50             int nowi = F.I+moveI[i]; //
51             找I上下左右
52             int nowj = F.J+moveJ[i]; //
53             找J上下左右
54             if(a[nowi][nowj] == 0 &&
55                 used[nowi][nowj] == 0){ //
56                 如果可走且沒走過
57                 used[nowi][nowj]=1;
58                 q.push(node(nowi, nowj,
59                     F.now+1)); //
60                 放到queue裡面(最不優先)
61             }
62         }
63     }
64     if(!flag) cout << 0 << '\n';
65     //如果找不到路輸出0;
66     return 0;
67 }

```

5.2 bwt

```

1  #include <stdio.h>
2  #include <stdlib.h>
3  #include <string.h>
4  #define MAXSTR 1000
5  #define MARKER '$'
6  #define NUM_ALPHA 26
7
8  int comp(const void *s, const void *t) {
9      return strcmp(*(char**)s, *(char**) t); /*
10         注意char的用法 */
11 }
12
13 /* the last char of s is not MARKER */
14 char* bwtEncode(char *s, const int n) {
15     char *L;
16     char **M;
17     int i, j, l;
18     if ((M = (char**) malloc(sizeof(char*) * n)) ==
19         NULL || \

```

```

18     (L = (char*) malloc(sizeof(char) * (n + 2)))
19     == NULL) {
20         fputs("Error: out of space!\n", stderr);
21         exit(1);
22     }
23     for (i = 0; i < n; i++) {
24         if ((M[i] = (char*) malloc(sizeof(char) * (i
25             + 2))) == NULL) { /*
26             只須保存開頭到標記的那一部份字母串，這樣總共可以省
27             */
28             fputs("Error: out of space!\n", stderr);
29             exit(1);
30         }
31         for (j = 0; j < i + 1; j++)
32             M[i][j] = s[n - 1 - i + j]; /*
33             這裡的字母串沒有存MARKER */
34         M[i][i + 1] = '\0';
35     }
36     qsort(M, n, sizeof(M[0]), comp); /*
37     對旋轉後的多個字母串排列 */
38     for (i = 0, L[0] = s[n - 1]; i < n; i++) {
39         if ((l = strlen(M[i])) < n)
40             L[i + 1] = s[n - 1 - l];
41         else
42             L[i + 1] = MARKER;
43     }
44     L[n + 1] = '\0';
45     for (i = 0; i < n; i++)
46         free(M[i]);
47     free(M);
48     return L;
49 }
50
51 char* bwtDecode(char *L, const int n) {
52     int i;
53     int *a, *b;
54     char *r; /* original string. */
55     int pos;
56     if ((a = (int*) calloc(NUM_ALPHA + 1,
57         sizeof(int))) == NULL || \
58         (b = (int*) calloc(n, sizeof(int))) == NULL
59         || \
60         (r = (char*) malloc(sizeof(char) * (n + 1)))
61         == NULL) {
62         fputs("Error: out of space!\n", stderr);
63         exit(1);
64     }
65     for (i = 0; i < n; i++) { /* L列中每種字母的個數
66         */
67         if (L[i] == MARKER)
68             a[0]++;
69         else
70             a[L[i] - 'a' + 1]++;
71     }
72     for (i = 1; i < NUM_ALPHA + 1; i++) { /*
73         F列中排在每種字母前面的其他字母的個數 */
74         a[i] += a[i - 1];
75     }
76     for (i = 0; i < n; i++) { /*
77         L列中每個字母跳轉到F列中的位置 */
78         if (L[i] == MARKER)
79             b[i] = 0;
80         else
81             b[i] = a[L[i] - 'a']++;
82     }
83     for (i = 0, pos = 0; i < n; i++) {
84         r[n - 1 - i] = L[pos];
85         pos = b[pos];
86     }
87     r[n] = '\0';
88     free(a);
89     free(b);
90     return r;
91 }
92
93 int main(void) {
94     char s[MAXSTR];

```

```
83 |     char *L;
84 |     int n;
85 |     char *r;
86 |     printf("input str: ");
87 |     fgets(s, MAXSTR - 1, stdin);
88 |     n = strlen(s);
89 |     if (s[n - 1] == '\n')
90 |         s[--n] = '\0';
91 |     L = bwtEncode(s, n);
92 |     printf("The L column: %s\n", L);
93 |     r = bwtDecode(L, ++n);
94 |     printf("The original str: %s\n", r);
95 |     free(L);
96 |     free(r);
97 | }
```