

Contents

1 Basic

- 1.1 BigNumber
- 1.2 BigNumber+ 小數轉分數
- 1.3 Gcd
- 1.4 Prime

2 DP

- 2.1 LCS
- 2.2 LIS(輸出幾個)
- 2.3 LIS(輸出序列)

3 Graph

- 3.1 DfsTemplate
- 3.2 BfsTemplate
- 3.3 Kruskal

4 Tree

- 4.1 SegmentTree(完整)
- 4.2 SegmentTree(建立)

1 Basic

1.1 BigNumber

```

1 #include <iostream>
2 #include <vector>
3 #include <algorithm>
4 #include <cstdio>
5 #include <cstdlib>
6 #include <cstring>
7 using namespace std;
8 void scan(char s[100], int a[100])
9 {
10     int i = 100 - 1;          // 大數的數字位置
11     int j = 0, n = strlen(s); // 字串的字元位置
12     while (i >= n) a[i--] = 0; // 開頭一律填零
13     while (i >= 0) a[i--] = s[j++] - '0'; // 字串頭尾
14                                     顛倒，存入陣列
15 }
16 void print(int a[100])
17 {
18     int i = 100 - 1;
19     while (i >= 0 && a[i] == 0) i--;
20
21     if (i < 0)
22         cout << '0';
23     else
24         while (i >= 0) cout << a[i--];
25 }
26 bool largertan(int a[100], int b[100])
27 {
28     // 從高位數開始比，對應的位數相比較。
29     for (int i=100-1; i>=0; i--)
30         if (a[i] != b[i]) // 一旦ab不一樣大，馬上回傳
31             return a[i] > b[i]; // 結果。
32     return false; // 完全相等
33 }
34 void add(int a[100], int b[100], int c[100])
35 {
36     for (int i=0; i<100; i++) // 對應的位數相加
37         c[i] = a[i] + b[i];
38
39     for (int i=0; i<100-1; i++) // 一口氣進位
40     {
41         c[i+1] += c[i] / 10; // 進位
42         c[i] %= 10; // 進位後餘下的數
43     }
44 }
45 void sub(int a[100], int b[100], int c[100])
46 {
47     for (int i=0; i<100; i++)
48         c[i] = a[i] - b[i];

```

```

48
49     for (int i=0; i<100-1; i++) // 一口氣借位和補位
50     {
51         if (c[i] < 0)
52         {
53             c[i+1]--; // 借位
54             c[i] += 10; // 補位
55         }
56     }
57 void mul(int a[100], int b[100], int c[100])
58 {
59     for (int i=0; i<100; i++)
60         c[i] = 0;
61
62     for (int i=0; i<100; i++)
63     {
64         for (int j=0; j<100; j++)
65             if (i+j < 100)
66                 c[i+j] += a[i] * b[j];
67     }
68
69     for (int i=0; i<100-1; i++) // 一口氣進位
70     {
71         c[i+1] += c[i] / 10;
72         c[i] %= 10;
73     }
74 }
75 void mul(int a[100], int b, int c[100])
76 {
77     for (int i=0; i<100; i++)
78         c[i] = a[i] * b;
79
80     for (int i=0; i<100-1; i++) // 一口氣進位
81     {
82         c[i+1] += c[i] / 10;
83         c[i] %= 10;
84     }
85 }
86 void div(int a[100], int b[100], int c[100])
87 {
88     int t[100];
89
90     for (int i=100-1; i>=0; i--)
91     {
92         for (int k=9; k>0; k--) // 嘗試商數
93         {
94             mul(b+i, k, t);
95             if (largertan(a+i, t))
96             {
97                 sub(a+i, t, c+i);
98                 break;
99             }
100         }
101     }
102     void div(int a[100], int b, int c[100])
103     {
104         int r = 0;
105         for (int i=100-1; i>=0; i--)
106         {
107             r = r * 10 + a[i];
108             c[i] = r / b;
109             r %= b;
110         }
111     }
112 }
113 int main()
114 {
115     char Str[100];
116     int BigNum[100];
117     cin >> Str;
118     scan(Str, BigNum);
119     print(BigNum);
120     return 0;

```

1.2 BigNumber+ 小數轉分數

```

1 //2017 NCPU Problem.1
2 #include <bits/stdc++.h>
3 using namespace std;
4 long long gcd(long long a, long long b)

```

```

5 {
6     while(b)
7         b ^= a ^= b ^= a ^= b;
8
9     return a;
10 }
11 int main(int argc, char const *argv[])
12 {
13     double L;
14     while(~scanf("%Lf",&L) && L){
15         long long rp = L,rq = 1,tmp;
16
17         while(L != (long long)L){
18             L *= 10;
19             rp = (long long)L;
20             rq *= 10;
21         }
22         tmp = gcd(rp,rq);
23         rp /= tmp;
24         rq /= tmp;
25         printf("%LLd/%LLd\n",rp,rq);
26     }
27
28     return 0;
29 }
30 // unsigned int 0~4294967295
31 // int 2147483648~2147483647
32 // unsigned long 0~4294967295
33 // Long 2147483648~2147483647
34 // Long Long的最大值: 9223372036854775807
35 // Long Long的最小值: -9223372036854775808
36 // unsigned Long Long的最大值: 1844674407370955161
37 // float的最大值和最小值分别为3.40282e+038 (10的38次
38 // 方) , 1.17549e-038 (10的38次方)
39 // double的最大值和最小值分别为1.79769e+308 (10的308次
40 // 方) , 2.22507e-308 (10的308次方)
41 //INPUT 3 3.14 1.234567890123456 0
42 //OUTPUT 3/1 157/50 19290123283179/15625000000000

```

1.3 Gcd

```

1 int gcd(int a, int b)
2 {
3     while(b)
4         b ^= a ^= b ^= a ^= b;
5
6     return a;
7 }
8 // gcd(a,b) ⊗ lcm(a,b) = ab
9 int extgcd(int a,int b,int &x,int &y){
10     int d=a;
11     if(b){d=extgcd(b,a%b,y,x),y-=(a/b)*x;}
12     else x=1,y=0;
13     return d;
14 } //ax+by=1 ax同餘 1 mod b

```

1.4 Prime

```

1 #define N 100000
2 bool pr[N+5];
3 long long int i , a;
4 void buildPr()
5 {
6
7     pr[0] = pr[1] = false;
8     for( i = 2; i <= N; i++)
9         pr[i] = true;
10
11     for( i = 2; i <= N; i++)
12         if(pr[i])
13             for( a = i*i; a < N; a += i)

```

```

14         pr[a] = false;
15 }
16
17 bool isPrime(int n)
18 {
19     for(int i = 2; i <= sqrt(n); i++)
20         if(n % i == 0)
21             return false;
22     return true;
23 }

```

2 DP

2.1 LCS

```

1 //DP
2 //LCS 最長子字串
3
4 #include <cstdio>
5 #include <cstdlib>
6 #include <cstring>
7 #include <algorithm>
8 #define strMAX
9
10 using namespace std;
11
12 char str1[strMAX];
13 char str2[strMAX];
14
15 scanf("%s", str1);
16 scanf("%s", str2);
17
18 int len1 = strlen(str1);
19 int len2 = strlen(str2);
20
21 int dp[len1+1][len2+2];
22 //dp包含空字串
23
24 memset(dp, 0, sizeof(dp));
25
26 for(int i = 1 ; i <= len1 ; i++)
27 {
28     for(int j = 1 ; j < len2 ; j++)
29     {
30         if(str1[i-1] == str2[j-1])
31             dp[i][j] = dp[i-1][j-1] + 1;
32         else
33             dp[i][j] = max(dp[i-1][j], dp[i][j-1]);
34     }
35 }
36
37 int ans = dp[str1][str2];

```

2.2 LIS(輸出幾個)

```

1 #include <iostream>
2 #include <vector>
3 #include <algorithm>
4 using namespace std;
5
6 int main(){
7     int N;
8     while(cin>>N){
9         vector<int>v;
10         int num;
11         for(int i=0;i<n;i++){
12             cin >> num;
13             if(!v.size() || num>v.back())
14                 v.push_back(num);
15             else
16                 *lower_bound(v.begin(), v.end(),num) =
17                     num;
18         }
19     }
20 }

```

```

17     }
18     cout << v.size() << endl;
19 }
20 }

```

2.3 LIS(輸出序列)

```

1 //Longest Increasing Subsequence (LIS)
2
3 #include <iostream>
4 #include <algorithm>
5 using namespace std;
6
7 int Count; //幾個數字
8 int seq[5];
9 int length[5];
10 int pv[5]; // prev[x] 記錄 s[x] 是接在哪個數字後面
11
12 void trace(int i)
13 {
14     if (pv[i] != -1) {
15         trace(pv[i]);
16     }
17     cout << seq[i] << ' ';
18 }
19
20 void LIS()
21 {
22     for (int i=0; i< Count; i++) length[i] = 1;
23
24     for (int i=0; i< Count; i++) pv[i] = -1; // -1
        代表 s[i] 是開頭數字，沒有接在其他數字後面。
25
26     for (int i=0; i< Count; i++)
27         for (int j=i+1; j< Count; j++)
28             if (seq[i] < seq[j])
29                 if (length[i] + 1 > length[j])
30                 {
31                     length[j] = length[i] + 1;
32                     pv[j] = i;
33                 }
34
35     int n = 0, pos = 0;
36     for (int i=0; i< Count; i++){
37         if (length[i] > n)
38         {
39             n = length[i];
40             pos = i;
41         }
42     }
43     trace(pos); // 印出一個LIS
44 }
45
46
47
48 int main(int argc, char const *argv[])
49 {
50
51     cin >> Count;
52     for(int i = 0 ; i < Count ; i++){
53         cin >> seq[i];
54     }
55     LIS();
56     return 0;
57 }

```

3 Graph

3.1 DfsTemplate

```

1 void dfs_visit (int i , int len){
2     for(int j = 0 ; j < n ; j++){

```

```

3         if(a[i][j]){
4             if(a[i][j]){
5                 a[i][j] = a[j][i] = 0;
6                 dfs_visit(j,len+1);
7                 a[i][j] = a[j][i] = 1;
8             }
9         }
10     if(len > best) best = len;
11 }
12 void dfs(){
13     int i;
14     best = 0 ;
15     for(int i = 0 ; i < n ; i++){
16         dfs_visit(i,0);
17     }
18     cout << best << endl;
19 }

```

3.2 BfsTemplate

```

1 void bfs(int s)
2 {
3     queue <int> que;
4     que.push(s);
5     vis[s] = 1;
6
7     while(!que.empty())
8     {
9         int tmp = que.front();
10        que.pop();
11
12        for(int i : G[s])
13        {
14            if(!vis[i])
15            {
16                que.push(i);
17                vis[i] = 1;
18            }
19        }
20    }
21 }

```

3.3 Dijkstra

```

1 //dijkstra
2 #include <iostream>
3 #include <cstdio>
4 #include <queue>
5 #include <vector>
6 using namespace std;
7
8 #define maxn 51415
9
10 struct Edge
11 {
12     int from, to, dist;
13
14     Edge(int _from, int _to, int _dist)
15     {
16         from = _from;
17         to = _to;
18         dist = _dist;
19     }
20 };
21
22 struct Item
23 {
24     int node;
25     int dist;
26
27     Item(int _node, int _dist)
28     {
29         node = _node;

```

```

30     dist = _dist;
31 }
32
33 bool operator <(const Item& rs) const
34 {
35     return dist > rs.dist;
36 }
37 };
38
39 int main(void)
40 {
41     int n, m;
42     while(~scanf("%d %d", &n, &m))
43     {
44         vector <Edge> edges;
45         vector<int> G[maxn];
46         priority_queue <Item> dij;
47         int visit[maxn] = {-1};
48
49         for(int i = 0; i < m; i++)
50         {
51             int a, b, c;
52             scanf("%d %d %d", &a, &b, &c);
53             edges.push_back(Edge(a, b, c));
54             G[a].push_back(i);
55         }
56
57         int node = 1;
58         dij.push(Item(1, 0));
59         Item hold = Item(0, 0);
60         while(!dij.empty())
61         {
62             hold = dij.top();
63             dij.pop();
64
65             if(visit[hold.node] == 1)
66                 continue;
67
68             visit[hold.node] = 1;
69
70             node = hold.node;
71             if(node == n)
72             {
73                 break;
74             }
75
76             for(int i = 0; i < G[node].size(); i++)
77             {
78                 dij.push(Item(edges[G[node][i]].to, hold.
79                     .dist+edges[G[node][i]].dist));
80             }
81
82             if(node != n) printf("-1\n");
83             if(node == n) printf("%d\n", hold.dist);
84         }
85         return 0;
86     }

```

3.4 Kruskal

```

1 //依題目：eMAX、nMAX
2
3 #define eMAX 205
4 #define nMAX 105
5 #define INF 1000000000
6
7 //n 點數量，e 邊數量，ise 紀錄幾號邊使用過
8 int n, e, ise[eMAX];
9 int father[nMAX];
10
11 struct Edge
12 {
13     int from, to, dist;
14 }edge[eMAX];

```

```

15
16 bool cmp(Edge a, Edge b)
17 {
18     return a.dist < b.dist;
19 }
20
21 //find：找尋是否為同個set
22 int find(int i)
23 {
24     if(father[i] == i) return i;
25     return find(father[i]);
26 }
27
28 int Kruskal()
29 {
30     int ans = 0, num = 0, id = 0;
31     for(int i = 1; i <= n; i++)
32     {
33         father[i] = i;
34     }
35     for(int i = 0; i < e; i++)
36     {
37         int n1 = edge[i].from;
38         int n2 = edge[i].to;
39         int n1f = find(n1);
40         int n2f = find(n2);
41         if(n1f != n2f)
42         {
43             ise[id] = i;
44             id++;
45             father[n1f] = n2f;
46             ans = ans + edge[i].dist;
47         }
48     }
49     for(int i = 1; i <= n; i++)
50     {
51         if(i == find(i))
52         {
53             num++;
54         }
55     }
56     if(num > 1)
57     {
58         return INF; //no solution
59     }
60     else
61     {
62         return ans; //min spanning tree cost
63     }
64 }

```

4 Tree

4.1 SegmentTree(完整)

```

1 #define N 10000
2 // 1-index
3 int t[4*N+5];
4 int in[N+5];
5
6 #define LEFT(x) ((x)<<1)
7 #define RIGHT(x) (((x)<<1)+1)
8 // parent, left, right
9 void buildSeg(int p, int inL, int inR)
10 {
11     if(inL == inR) {
12         t[p] = in[inL];
13         return;
14     }
15     int mid = (inL+inR)/2;
16     buildSeg(LEFT(p), inL, mid); // build left
17     buildSeg(RIGHT(p), mid+1, inR); // build right

```

```

18     t[p] = max(t[LEFT(p)], t[RIGHT(p)]);
19 }
20 // treeIdx, left, right, targetIdx, targetVal
21 void modify(int p, int L, int R, int i, int x)
22 {
23     // stop point
24     if(i == L && L == R) {
25         t[p] = x;
26         return;
27     }
28     int mid = (L+R) / 2;
29     if(i <= mid)
30         modify(LEFT(p), L, mid, i, x);
31     else
32         modify(RIGHT(p), mid+1, R, i, x);
33     // update this node
34     t[p] = max(t[LEFT(p)], t[RIGHT(p)]);
35 }
36 // treeIdx, left, right, queryleft, queryright
37 int query(int p, int L, int R, int quL, int quR)
38 {
39     if(quL <= L && R <= quR) {
40         return t[p];
41     }
42     int mid = (L+R) / 2;
43     if(quR <= mid) // left
44         return query(LEFT(p), L, mid, quL, quR);
45     else if(mid < quL) // right
46         return query(RIGHT(p), mid+1, R, quL, quR);
47     else // middle
48         return max(query(LEFT(p), L, mid, quL, quR),
49                     query(RIGHT(p), mid+1, R, quL, quR));
50 }

```

4.2 SegmentTree(建立)

```

1
2
3 typedef vector<int> vi;
4 #define INF 1e18
5
6 vi ST;
7 vi A;
8
9 void st_build(vi &ST, const vi &A, int vertex, int L,
10              int R)
11 {
12     if(L == R) ST[vertex] = L;
13     else
14     {
15         int nL = 2*vertex, nR = 2*vertex+1;
16
17         st_build(ST, A, nL, L, (L+R)/2);
18         st_build(ST, A, nR, (L+R)/2+1, R);
19
20         int lContent = ST[nL], rContent = ST[nR];
21         int lValue = A[lContent], rValue = A[rContent];
22         ST[vertex] = (lValue <= rValue) ? lContent :
23                     rContent;
24     }
25 }
26
27 void st_create(vi &ST, const vi &A)
28 {
29     int len = (int) 4*A.size();
30     ST.assign(len, 0);
31     st_build(ST, A, 1, 0, (int)A.size() - 1);
32 }

```