

1 Basic

1.1 data range

```
1 int (-2147483648 to 2147483647)
2 unsigned int(0 to 4294967295)
3 long(-2147483648 to 2147483647)
4 unsigned long(0 to 4294967295)
5 long long(-9223372036854775808 to
6   9223372036854775807)
7 unsigned long long (0 to
8   18446744073709551615)
```

1.2 IO_fast

```
1 ios_base::sync_with_stdio(0);
2 cin.tie(0);
```

2 DP

2.1 Knapsack Bounded

```
1 const int N = 100, W = 100000;
2 int cost[N], weight[N], number[N];
3 int c[W + 1];
4 void knapsack(int n, int w)
5 {
6     for (int i = 0; i < n; ++i)
7     {
8         int num = min(number[i], w / weight[i]);
9         for (int k = 1; num > 0; k *= 2)
10         {
11             if (k > num)
12                 k = num;
13             num -= k;
14             for (int j = w; j >= weight[i] * k; --j)
15                 c[j] = max(c[j], c[j - weight[i] * k] + cost[i] * k);
16         }
17     }
18     cout << "Max Prince" << c[w];
19 }
```

2.2 Knapsack sample

```
1 int Knapsack(vector<int> weight, vector<int>
2   value, int bag_Weight)
3 {
4     // vector<int> weight = {1, 3, 4};
```

```
4     // vector<int> value = {15, 20, 30};
5     // int bagWeight = 4;
6     vector<vector<int>> dp(weight.size(),
7       vector<int>(bagWeight + 1, 0));
8     for (int j = weight[0]; j <= bagWeight; j++)
9         dp[0][j] = value[0];
10    // weight數組的大小就是物品個數
11    for (int i = 1; i < weight.size(); i++)
12    { // 遍歷物品
13        for (int j = 0; j <= bagWeight; j++)
14        { // 遍歷背包容量
15            if (j < weight[i]) dp[i][j] = dp[i - 1][j];
16            else dp[i][j] = max(dp[i - 1][j], dp[i - 1][j - weight[i]] + value[i]);
17        }
18    }
19    cout << dp[weight.size() - 1][bagWeight] << endl;
```

2.3 Knapsack Unbounded

```
1 const int N = 100, W = 100000;
2 int cost[N], weight[N];
3 int c[W + 1];
4 void knapsack(int n, int w)
5 {
6     memset(c, 0, sizeof(c));
7     for (int i = 0; i < n; ++i)
8     {
9         for (int j = weight[i]; j <= w; ++j)
10             c[j] = max(c[j], c[j - weight[i]] + cost[i]);
11    }
12    cout << "最高的價值為" << c[w];
```

2.4 LCIS

```
1 int LCIS_len(vector<int> arr1, vector<int>
2   arr2)
3 {
4     int n = arr1.size(), m = arr2.size();
5     vector<int> table(m, 0);
6     for (int j = 0; j < m; j++)
7         table[j] = 0;
8     for (int i = 0; i < n; i++)
9     {
10        int current = 0;
11        for (int j = 0; j < m; j++)
12        {
13            if (arr1[i] == arr2[j])
14                if (current + 1 > table[j])
15                    table[j] = current + 1;
16        }
17        if (arr1[i] > arr2[j])
18            if (table[j] > current)
```

```
19        current = table[j];
20    }
21    }
22    int result = 0;
23    for (int i = 0; i < m; i++)
24        if (table[i] > result)
25            result = table[i];
26    return result;
27 }
```

2.5 LCS

```
1 int LCS(vector<string> Ans, vector<string>
2   num)
3 {
4     int N = Ans.size(), M = num.size();
5     vector<vector<int>> LCS(N + 1, vector<
6       int>(M + 1, 0));
7     for (int i = 1; i <= N; ++i)
8     {
9         for (int j = 1; j <= M; ++j)
10        {
11            if (Ans[i - 1] == num[j - 1])
12                LCS[i][j] = LCS[i - 1][j - 1] + 1;
13            else
14                LCS[i][j] = max(LCS[i - 1][j], LCS[i][j - 1]);
15        }
16    }
17    cout << LCS[N][M] << '\n';
18    //列印 LCS
19    int n = N, m = M;
20    vector<string> k;
21    while (n && m)
22    {
23        if (LCS[n][m] != max(LCS[n - 1][m], LCS[n][m - 1]))
24        {
25            k.push_back(Ans[n - 1]);
26            n--;
27            m--;
28        }
29        else if (LCS[n][m] == LCS[n - 1][m])
30            n--;
31        else if (LCS[n][m] == LCS[n][m - 1])
32            m--;
33    }
34    reverse(k.begin(), k.end());
35    for (auto i : k)
36        cout << i << " ";
37    cout << endl;
38    return LCS[N][M];
39 }
```

2.6 LIS

```
1 void getMaxElementAndPos(vector<int> &LISTbl
2   , vector<int> &LISlen, int tNum, int
3   tlen, int tStart, int &num, int &pos)
```

```
2 {
3     int max = numeric_limits<int>::min();
4     int maxPos;
5     for (int i = tStart; i >= 0; i--)
6     {
7         if (LISlen[i] == tlen && LISTbl[i] <
8             tNum)
9         {
10            if (LISTbl[i] > max)
11            {
12                max = LISTbl[i];
13                maxPos = i;
14            }
15        }
16    }
17    num = max;
18    pos = maxPos;
19 }
20 int LIS(vector<int> &LISTbl)
21 {
22     if (LISTbl.size() == 0)
23         return 0;
24     vector<int> LISlen(LISTbl.size(), 1);
25     for (int i = 1; i < LISTbl.size(); i++)
26     {
27         for (int j = 0; j < i; j++)
28             if (LISTbl[j] < LISTbl[i])
29                 LISlen[i] = max(LISlen[i], LISlen[j] + 1);
30     }
31     int maxlen = *max_element(LISlen.begin(), LISlen.end());
32     int num, pos;
33     vector<int> buf;
34     getMaxElementAndPos(LISTbl, LISlen,
35       numeric_limits<int>::max(), maxlen,
36       LISTbl.size() - 1, num, pos);
37     buf.push_back(num);
38     for (int len = maxlen - 1; len >= 1; len --)
39     {
40         int tnum = num;
41         int tpos = pos;
42         getMaxElementAndPos(LISTbl, LISlen,
43           tnum, len, tpos - 1, num, pos);
44         buf.push_back(tnum);
45     }
46     reverse(buf.begin(), buf.end());
47     for (int k = 0; k < buf.size(); k++) //
48         列印
49     {
50         if (k == buf.size() - 1)
51             cout << buf[k] << endl;
52         else
53             cout << buf[k] << ",";
54     }
55     return maxlen;
56 }
```

2.7 LPS

```
1 void LPS(string s)
2 {
3     int maxlen = 0, l, r;
4     int n = s;
```

```

5 for (int i = 0; i < n; i++)
6 {
7     int x = 0;
8     while ((s[i - x] == s[i + x]) && (i
9         - x >= 0) && (i + x < n)) //odd
10         length
11         x++;
12     x--;
13     if (2 * x + 1 > maxlen)
14     {
15         maxlen = 2 * x + 1;
16         l = i - x;
17         r = i + x;
18     }
19     x = 0;
20     while ((s[i - x] == s[i + 1 + x]) &&
21         (i - x >= 0) && (i + 1 + x < n)
22         ) //even length
23         x++;
24     if (2 * x > maxlen)
25     {
26         maxlen = 2 * x;
27         l = i - x + 1;
28         r = i + x;
29     }
30 }
31 cout << maxlen << '\n'; // 最後長度
32 cout << l + 1 << ' ' << r + 1 << '\n';
33 //頭到尾

```

2.8 Max_subarray

```

1 /*Kadane's algorithm*/
2 int maxSubArray(vector<int>& nums) {
3     int local_max = nums[0], global_max =
4     nums[0];
5     for (int i = 1; i < nums.size(); i++){
6         local_max = max(nums[i], nums[i] +
7         local_max);
8         global_max = max(local_max,
9         global_max);
10    }
11    return global_max;
12 }

```

2.9 Money problem

```

1 //能否湊得某個價位
2 void change(vector<int> price, int limit)
3 {
4     vector<bool> c(limit + 1, 0);
5     c[0] = true;
6     for (int i = 0; i < price.size(); ++i)
7         // 依序加入各種面額
8         for (int j = price[i]; j <= limit;
9             ++j) // 由低價位逐步到高價位
10             c[j] = c[j] | c[j - price[i]];
11         // 湊、湊、湊

```

```

9     if (c[limit]) cout << "YES\n";
10    else cout << "NO\n";
11 }
12 // 湊得某個價位的湊法總共幾種
13 void change(vector<int> price, int limit)
14 {
15     vector<int> c(limit + 1, 0);
16     c[0] = true;
17     for (int i = 0; i < price.size(); ++i)
18         for (int j = price[i]; j <= limit;
19             ++j)
20             c[j] += c[j - price[i]];
21     cout << c[limit] << '\n';
22 }
23 // 湊得某個價位的最少錢幣用量
24 void change(vector<int> price, int limit)
25 {
26     vector<int> c(limit + 1, 0);
27     c[0] = true;
28     for (int i = 0; i < price.size(); ++i)
29         for (int j = price[i]; j <= limit;
30             ++j)
31             c[j] = min(c[j], c[j - price[i]]
32             + 1);
33     cout << c[limit] << '\n';
34 }
35 //湊得某個價位的錢幣用量・有哪幾種可能性
36 void change(vector<int> price, int limit)
37 {
38     vector<int> c(limit + 1, 0);
39     c[0] = true;
40     for (int i = 0; i < price.size(); ++i)
41         for (int j = price[i]; j <= limit;
42             ++j)
43             c[j] |= c[j - price[i]] << 1; //
44             錢幣數量加一・每一種可能性都
45             加一。
46     for (int i = 1; i <= 63; ++i)
47         if (c[i] & (1 << i))
48             cout << "用" << i << "個錢幣可湊
49             得價位" << i;
50 }

```

3 Flow & matching

3.1 Edmonds_karp

```

1 /*Flow - Edmonds-karp*/
2 /*Based on UVA820*/
3 #include<bits/stdc++.h>
4 #define inf 1000000;
5 using namespace std;
6
7 int getMaxFlow(vector<vector<int>> &capacity
8     , int s, int t, int n){
9     int ans = 0;
10    vector<vector<int>> residual(n+1, vector<
11        int>(n+1, 0)); //residual network
12    while(true){

```

```

11    vector<int> bottleneck(n+1, 0);
12    bottleneck[s] = inf;
13    queue<int> q;
14    q.push(s);
15    vector<int> pre(n+1, 0);
16    while(!q.empty() && bottleneck[t] == 0){
17        int cur = q.front();
18        q.pop();
19        for(int i = 1; i <= n; i++){
20            if(bottleneck[i] == 0 && capacity[
21                cur][i] > residual[cur][i]){
22                q.push(i);
23                pre[i] = cur;
24                bottleneck[i] = min(bottleneck[cur
25                    ], capacity[cur][i] - residual
26                    [cur][i]);
27            }
28        }
29        if(bottleneck[t] == 0) break;
30        for(int cur = t; cur != s; cur = pre[cur
31            ]){
32            residual[pre[cur]][cur] +=
33            bottleneck[t];
34            residual[cur][pre[cur]] -=
35            bottleneck[t];
36        }
37        ans += bottleneck[t];
38    }
39    return ans;
40 }
41 int main(){
42     int testcase = 1;
43     while(cin >> n){
44         if(n == 0)
45             break;
46         vector<vector<int>> capacity(n+1, vector
47             <int>(n+1, 0));
48         int s, t, c;
49         cin >> s >> t >> c;
50         int a, b, bandwidth;
51         for(int i = 0; i < c; ++i){
52             cin >> a >> b >> bandwidth;
53             capacity[a][b] += bandwidth;
54             capacity[b][a] += bandwidth;
55         }
56         cout << "Network " << testcase++ << endl
57             ;
58         cout << "The bandwidth is " <<
59             getMaxFlow(capacity, s, t, n) << "."
60             << endl;
61         cout << endl;
62     }
63     return 0;
64 }

```

3.2 maximum_matching

```

1 /*bipartite - maximum matching*/
2 #include<bits/stdc++.h>
3 using namespace std;

```

```

4 bool dfs(vector<vector<bool>> res, int node,
5     vector<int>& x, vector<int>& y, vector<
6     bool> pass){
7     for (int i = 0; i < res[0].size(); i++){
8         if(res[node][i] && !pass[i]){
9             pass[i] = true;
10            if(y[i] == -1 || dfs(res, y[i], x,
11                y, pass)){
12                x[node] = i;
13                y[i] = node;
14                return true;
15            }
16        }
17    }
18    return false;
19 }
20 int main(){
21     int n, m, l;
22     while(cin >> n >> m >> l){
23         vector<vector<bool>> res(n, vector<
24             bool>(m, false));
25         for (int i = 0; i < l; i++){
26             int a, b;
27             cin >> a >> b;
28             res[a][b] = true;
29         }
30         int ans = 0;
31         vector<int> x(n, -1);
32         vector<int> y(m, -1);
33         for (int i = 0; i < n; i++){
34             vector<bool> pass(m, false);
35             if(dfs(res, i, x, y, pass))
36                 ans += 1;
37         }
38         cout << ans << endl;
39     }
40     return 0;
41 }
42 /*
43 input:
44 4 3 5 //n matching m, l links
45 0 0
46 0 2
47 1 0
48 2 1
49 3 1
50 answer is 3
51 */

```

3.3 MFlow Model

```

1 typedef long long ll;
2 struct MF
3 {
4     static const int N = 5000 + 5;
5     static const int M = 60000 + 5;
6     static const ll oo = 10000000000000LL;
7
8     int n, m, s, t, tot, tim;
9     int first[N], next[M];
10    int u[M], v[M], cur[N], vi[N];
11    ll cap[M], flow[M], dis[N];
12    int que[N + N];

```

```

13 void Clear()
14 {
15     tot = 0;
16     tim = 0;
17     for (int i = 1; i <= n; ++i)
18         first[i] = -1;
19 }
20 void Add(int from, int to, ll cp, ll flw)
21 {
22     u[tot] = from;
23     v[tot] = to;
24     cap[tot] = cp;
25     flow[tot] = flw;
26     next[tot] = first[u[tot]];
27     first[u[tot]] = tot;
28     ++tot;
29 }
30 bool bfs()
31 {
32     ++tim;
33     dis[s] = 0;
34     vi[s] = tim;
35
36     int head, tail;
37     head = tail = 1;
38     que[head] = s;
39     while (head <= tail)
40     {
41         for (int i = first[que[head]]; i
42             != -1; i = next[i])
43         {
44             if (vi[v[i]] != tim && cap[i]
45                 > flow[i])
46             {
47                 vi[v[i]] = tim;
48                 dis[v[i]] = dis[que[head]]
49                     + 1;
50                 que[++tail] = v[i];
51             }
52             ++head;
53         }
54         return vi[t] == tim;
55     }
56     ll dfs(int x, ll a)
57     {
58         if (x == t || a == 0)
59             return a;
60         ll flw = 0, f;
61         int &i = cur[x];
62         for (i = first[x]; i != -1; i = next
63             [i])
64         {
65             if (dis[x] + 1 == dis[v[i]] && (
66                 f = dfs(v[i], min(a, cap[i]
67                     - flow[i]))) > 0)
68             {
69                 flow[i] += f;
70                 flow[i ^ 1] -= f;
71                 a -= f;
72                 flw += f;
73                 if (a == 0)
74                     break;
75             }
76         }
77         return flw;
78     }
79 }

```

```

72     }
73     return flw;
74 }
75 ll MaxFlow(int s, int t)
76 {
77     this->s = s;
78     this->t = t;
79     ll flw = 0;
80     while (bfs())
81     {
82         for (int i = 1; i <= n; ++i)
83             cur[i] = 0;
84         flw += dfs(s, oo);
85     }
86     return flw;
87 }
88 };
89 // MF Net;
90 // Net.n = n;
91 // Net.Clear();
92 // a 到 b (注意從1開始!!!!)
93 // Net.Add(a, b, w, 0);
94 // Net.MaxFlow(s, d)
95 // s 到 d 的 MF

```

4 Geometry

4.1 Line

```

1 template <typename T>
2 struct line
3 {
4     line() {}
5     point<T> p1, p2;
6     T a, b, c; //ax+by+c=0
7     line(const point<T> &x, const point<T> &
8         y) : p1(x), p2(y) {}
9     void pton()
10     { //轉成一般式
11         a = p1.y - p2.y;
12         b = p2.x - p1.x;
13         c = -a * p1.x - b * p1.y;
14     }
15     T ori(const point<T> &p) const
16     { //點和有向直線的關係·>0左邊、=0在線上
17         <0右邊
18         return (p2 - p1).cross(p - p1);
19     }
20     T btw(const point<T> &p) const
21     { //點投影落在線段上<=0
22         return (p1 - p).dot(p2 - p);
23     }
24     bool point_on_segment(const point<T> &p)
25     const
26     { //點是否在線段上
27         return ori(p) == 0 && btw(p) <= 0;
28     }
29     T dis2(const point<T> &p, bool
30         is_segment = 0) const
31     { //點跟直線/線段的距離平方

```

```

28     point<T> v = p2 - p1, v1 = p - p1;
29     if (is_segment)
30     {
31         point<T> v2 = p - p2;
32         if (v.dot(v1) <= 0)
33             return v1.abs2();
34         if (v.dot(v2) >= 0)
35             return v2.abs2();
36     }
37     T tmp = v.cross(v1);
38     return tmp * tmp / v.abs2();
39 }
40 T seg_dis2(const line<T> &l) const
41 { //兩線段距離平方
42     return min({dis2(l.p1, 1), dis2(l.p2
43         , 1), l.dis2(p1, 1), l.dis2(p2,
44         1)});
45 }
46 point<T> projection(const point<T> &p)
47 const
48 { //點對直線的投影
49     point<T> n = (p2 - p1).normal();
50     return p - n * (p - p1).dot(n) / n.
51         abs2();
52 }
53 point<T> mirror(const point<T> &p) const
54 {
55     //點對直線的鏡射·要先呼叫pton轉成一般式
56     point<T> R;
57     T d = a * a + b * b;
58     R.x = (b * b * p.x - a * a * p.x - 2
59         * a * b * p.y - 2 * a * c) / d;
60     R.y = (a * a * p.y - b * b * p.y - 2
61         * a * b * p.x - 2 * b * c) / d;
62     return R;
63 }
64 bool equal(const line &l) const
65 { //直線相等
66     return ori(l.p1) == 0 && ori(l.p2)
67         == 0;
68 }
69 bool parallel(const line &l) const
70 {
71     return (p1 - p2).cross(l.p1 - l.p2)
72         == 0;
73 }
74 bool cross_seg(const line &l) const
75 {
76     return (p2 - p1).cross(l.p1 - p1) *
77         (p2 - p1).cross(l.p2 - p1) <= 0;
78     //直線是否交線段
79 }
80 int line_intersection(const line &l) const
81 { //線段相交情況·-1無限多點、1交於一點、0不相交
82     return parallel(l) ? (ori(l.p1) == 0
83         ? -1 : 0) : 1;
84 }
85 int seg_intersect(const line &l) const
86 {
87     T c1 = ori(l.p1), c2 = ori(l.p2);
88     T c3 = l.ori(p1), c4 = l.ori(p2);
89     if (c1 == 0 && c2 == 0)

```

```

79 { //共線
80     bool b1 = btw(l.p1) >= 0, b2 =
81         btw(l.p2) >= 0;
82     T a3 = l.btw(p1), a4 = l.btw(p2);
83     if (b1 && b2 && a3 == 0 && a4 >=
84         0)
85         return 2;
86     if (b1 && b2 && a3 >= 0 && a4 ==
87         0)
88         return 3;
89     if (b1 && b2 && a3 >= 0 && a4 >=
90         0)
91         return 0;
92     else if (c1 * c2 <= 0 && c3 * c4 <=
93         0)
94         return 1;
95     return 0; //不相交
96 }
97 point<T> line_intersection(const line &l)
98 const
99 { /*直線交點*/
100     point<T> a = p2 - p1, b = l.p2 - l.
101         p1, s = l.p1 - p1;
102     //if(a.cross(b)==0)return INF;
103     return p1 + a * (s.cross(b) / a.
104         cross(b));
105 }
106 point<T> seg_intersection(const line &l)
107 const
108 { //線段交點
109     int res = seg_intersect(l);
110     if (res <= 0)
111         assert(0);
112     if (res == 2)
113         return p1;
114     if (res == 3)
115         return p2;
116     return line_intersection(l);
117 }

```

4.2 Point

```

1 template <typename T>
2 struct point
3 {
4     T x, y;
5     point() {}
6     point(const T &x, const T &y) : x(x), y(
7         y) {}
8     point operator+(const point &b) const
9     {
10         return point(x + b.x, y + b.y);
11     }
12     point operator-(const point &b) const
13     {
14         return point(x - b.x, y - b.y);
15     }
16     point operator*(const T &b) const

```

```

16 {
17     return point(x * b, y * b);
18 }
19 point operator/(const T &b) const
20 {
21     return point(x / b, y / b);
22 }
23 bool operator==(const point &b) const
24 {
25     return x == b.x && y == b.y;
26 }
27 T dot(const point &b) const
28 {
29     return x * b.x + y * b.y;
30 }
31 T cross(const point &b) const
32 {
33     return x * b.y - y * b.x;
34 }
35 point normal() const
36 { //求法向量
37     return point(-y, x);
38 }
39 T abs2() const
40 { //向量长度的平方
41     return dot(*this);
42 }
43 T rad(const point &b) const
44 { //兩向量的弧度
45     return fabs(atan2(fabs(cross(b)),
46                          dot(b)));
47 }
48 T getA() const
49 { //對x軸的弧度
50     T A = atan2(y, x); //超過180度會變負
51     if (A <= -PI / 2)
52         A += PI * 2;
53     return A;
54 };

```

4.3 Polygon

```

1 template <typename T>
2 struct polygon
3 {
4     polygon() {}
5     vector<point<T>> p; //逆時針順序
6     T area() const
7     { //面積
8         T ans = 0;
9         for (int i = p.size() - 1, j = 0; j
10              < (int)p.size(); i = j++)
11             ans += p[i].cross(p[j]);
12         return ans / 2;
13     }
14     point<T> center_of_mass() const
15     { //重心
16         T cx = 0, cy = 0, w = 0;
17         for (int i = p.size() - 1, j = 0; j
18              < (int)p.size(); i = j++)

```

```

17 {
18     T a = p[i].cross(p[j]);
19     cx += (p[i].x + p[j].x) * a;
20     cy += (p[i].y + p[j].y) * a;
21     w += a;
22 }
23 return point<T>(cx / 3 / w, cy / 3 /
24                w);
25 }
26 char ahas(const point<T> &t) const
27 { //點是否在簡單多邊形內·是的話回傳1、
28   在邊上回傳-1、否則回傳0
29     bool c = 0;
30     for (int i = 0, j = p.size() - 1; i
31          < p.size(); j = i++)
32         if (line<T>(p[i], p[j]).
33             point_on_segment(t))
34             return -1;
35         else if ((p[i].y > t.y) != (p[j]
36                .y > t.y) &&
37                t.x < (p[j].x - p[i].x)
38                    * (t.y - p[i].y) /
39                    (p[j].y - p[i].y)
40                    + p[i].x)
41             c = !c;
42     return c;
43 }
44 char point_in_convex(const point<T> &x)
45 const
46 {
47     int l = 1, r = (int)p.size() - 2;
48     while (l <= r)
49     { //點是否在凸多邊形內·是的話回傳1
50       在邊上回傳-1、否則回傳0
51         int mid = (l + r) / 2;
52         T a1 = (p[mid] - p[0]).cross(x -
53            p[0]);
54         T a2 = (p[mid + 1] - p[0]).cross
55            (x - p[0]);
56         if (a1 >= 0 && a2 <= 0)
57         {
58             T res = (p[mid + 1] - p[mid]
59                ).cross(x - p[mid]);
60             return res > 0 ? 1 : (res >=
61                0 ? -1 : 0);
62         }
63         else if (a1 < 0)
64             r = mid - 1;
65         else
66             l = mid + 1;
67     }
68     return 0;
69 }
70 vector<T> getA() const
71 { //凸包邊對x軸的夾角
72     vector<T> res; //一定是遞增的
73     for (size_t i = 0; i < p.size(); ++i
74         )
75         res.push_back((p[(i + 1) % p.
76            size()] - p[i]).getA());
77     return res;
78 }
79 bool line_intersect(const vector<T> &A,
80                     const line<T> &l) const
81 { //O(logN)

```

```

107 int f1 = upper_bound(A.begin(), A.
108 end(), (l.p1 - l.p2).getA()) - A
109 .begin();
110 int f2 = upper_bound(A.begin(), A.
111 end(), (l.p2 - l.p1).getA()) - A
112 .begin();
113 return l.cross_seg(line<T>(p[f1], p[
114 f2]));
115 }
116 polygon cut(const line<T> &l) const
117 { //凸包對直線切割·得到直線l左側的凸包
118     polygon ans;
119     for (int n = p.size(), i = n - 1, j
120          = 0; j < n; i = j++)
121     {
122         if (l.ori(p[i]) >= 0)
123         {
124             ans.p.push_back(p[i]);
125             if (l.ori(p[j]) < 0)
126                 ans.p.push_back(l.
127                     line_intersection(
128                         line<T>(p[i], p[j])),
129                     );
130         }
131         else if (l.ori(p[j]) > 0)
132             ans.p.push_back(l.
133                 line_intersection(line<T>
134                    >(p[i], p[j])));
135     }
136     return ans;
137 }
138 static bool graham_cmp(const point<T> &a
139 , const point<T> &b)
140 { //凸包排序函數 // 起始點不同
141     // return (a.x < b.x) || (a.x == b.x
142     // && a.y < b.y); //最左下角開始
143     return (a.y < b.y) || (a.y == b.y &&
144        a.x < b.x); //Y最小開始
145 }
146 void graham(vector<point<T>> &s)
147 { //凸包 Convexhull 2D
148     sort(s.begin(), s.end(), graham_cmp)
149     ;
150     p.resize(s.size() + 1);
151     int m = 0;
152     // cross >= 0 順時針·cross <= 0 逆
153     時針旋轉
154     for (size_t i = 0; i < s.size(); ++i
155         )
156     {
157         while (m >= 2 && (p[m - 1] - p[m
158            - 2]).cross(s[i] - p[m -
159            2]) <= 0)
160             --m;
161         p[m++] = s[i];
162     }
163     for (int i = s.size() - 2, t = m +
164          1; i >= 0; --i)
165     {
166         while (m >= t && (p[m - 1] - p[m
167            - 2]).cross(s[i] - p[m -
168            2]) <= 0)
169             --m;
170         p[m++] = s[i];
171     }

```

```

172 }
173 if (s.size() > 1) // 重複頭一次需扣
174     掉
175     --m;
176     p.resize(m);
177 }
178 T diam()
179 { //直徑
180     int n = p.size(), t = 1;
181     T ans = 0;
182     p.push_back(p[0]);
183     for (int i = 0; i < n; i++)
184     {
185         point<T> now = p[i + 1] - p[i];
186         while (now.cross(p[t + 1] - p[i]
187            ]) > now.cross(p[t] - p[i]))
188             t = (t + 1) % n;
189         ans = max(ans, (p[i] - p[t]).
190             abs2());
191     }
192     return p.pop_back(), ans;
193 }
194 T min_cover_rectangle()
195 { //最小覆蓋矩形
196     int n = p.size(), t = 1, r = 1, l;
197     if (n < 3)
198         return 0; //也可以做最小周長矩形
199     T ans = 1e99;
200     p.push_back(p[0]);
201     for (int i = 0; i < n; i++)
202     {
203         point<T> now = p[i + 1] - p[i];
204         while (now.cross(p[t + 1] - p[i]
205            ]) > now.cross(p[t] - p[i]))
206             t = (t + 1) % n;
207         while (now.dot(p[r + 1] - p[i])
208            > now.dot(p[r] - p[i]))
209             r = (r + 1) % n;
210         if (l)
211             l = r;
212         while (now.dot(p[l + 1] - p[i])
213            <= now.dot(p[l] - p[i]))
214             l = (l + 1) % n;
215         T d = now.abs2();
216         T tmp = now.cross(p[t] - p[i]) *
217             (now.dot(p[r] - p[i]) - now
218                .dot(p[l] - p[i])) / d;
219         ans = min(ans, tmp);
220     }
221     return p.pop_back(), ans;
222 }
223 T dis2(polygon &p1)
224 { //凸包最近距離平方
225     vector<point<T>> &P = p, &Q = p1.p;
226     int n = P.size(), m = Q.size(), l =
227         0, r = 0;
228     for (int i = 0; i < n; i++)
229         if (P[i].y < P[l].y)
230             l = i;
231     for (int i = 0; i < m; i++)
232         if (Q[i].y < Q[r].y)
233             r = i;
234     P.push_back(P[0]), Q.push_back(Q[0])
235     ;
236     T ans = 1e99;

```

```

162 for (int i = 0; i < n; ++i)
163 {
164     while ((P[l] - P[l + 1]).cross(Q
165         [r + 1] - Q[r]) < 0)
166         r = (r + 1) % m;
167     ans = min(ans, line<T>(P[l], P[l
168         + 1]).seg_dis2(line<T>(Q[r
169         ], Q[r + 1])));
170     l = (l + 1) % n;
171     return P.pop_back(), Q.pop_back(),
172     ans;
173 }
174 static char sign(const point<T> &t)
175 {
176     return (t.y == 0 ? t.x : t.y) < 0;
177 }
178 static bool angle_cmp(const line<T> &A,
179     const line<T> &B)
180 {
181     point<T> a = A.p2 - A.p1, b = B.p2 -
182     B.p1;
183     return sign(a) < sign(b) || (sign(a)
184         == sign(b) && a.cross(b) > 0);
185 }
186 int halfplane_intersection(vector<line<T>
187     >> &s)
188 {
189     //半平面交
190     sort(s.begin(), s.end(), angle_cmp);
191     //線段左側為該線段半平面
192     int L, R, n = s.size();
193     vector<point<T>> px(n);
194     vector<line<T>> q(n);
195     q[L = R = 0] = s[0];
196     for (int i = 1; i < n; ++i)
197     {
198         while (L < R && s[i].ori(px[R -
199             1]) <= 0)
200             --R;
201         while (L < R && s[i].ori(px[L])
202             <= 0)
203             ++L;
204         q[++R] = s[i];
205         if (q[R].parallel(q[R - 1]))
206         {
207             --R;
208             if (q[R].ori(s[i].p1) > 0)
209                 q[R] = s[i];
210         }
211         if (L < R)
212             px[R - 1] = q[R - 1].
213             line_intersection(q[R]);
214     }
215     while (L < R && q[L].ori(px[R - 1])
216         <= 0)
217         --R;
218     p.clear();
219     if (R - L <= 1)
220         return 0;
221     px[R] = q[R].line_intersection(q[L]);
222     for (int i = L; i <= R; ++i)
223         p.push_back(px[i]);
224     return R - L + 1;
225 }

```

4.4 Triangle

```

213 };
214
215 template <typename T>
216 struct triangle
217 {
218     point<T> a, b, c;
219     triangle() {}
220     triangle(const point<T> &a, const point<
221         T> &b, const point<T> &c) : a(a), b(
222         b), c(c) {}
223     T area() const
224     {
225         T t = (b - a).cross(c - a) / 2;
226         return t > 0 ? t : -t;
227     }
228     point<T> barycenter() const
229     { //重心
230         return (a + b + c) / 3;
231     }
232     point<T> circumcenter() const
233     { //外心
234         static line<T> u, v;
235         u.p1 = (a + b) / 2;
236         u.p2 = point<T>(u.p1.x - a.y + b.y,
237             u.p1.y + a.x - b.x);
238         v.p1 = (a + c) / 2;
239         v.p2 = point<T>(v.p1.x - a.y + c.y,
240             v.p1.y + a.x - c.x);
241         return u.line_intersection(v);
242     }
243     point<T> incenter() const
244     { //內心
245         T A = sqrt((b - c).abs2()), B = sqrt
246         ((a - c).abs2()), C = sqrt((a -
247             b).abs2());
248         return point<T>(A * a.x + B * b.x +
249             C * c.x, A * a.y + B * b.y + C *
250             c.y) / (A + B + C);
251     }
252     point<T> perpencenter() const
253     { //垂心
254         return barycenter() * 3 -
255             circumcenter() * 2;
256     }
257 }
258 };

```

5 Graph

5.1 Bellman-Ford

```

259 /*SPA - Bellman-Ford*/
260 #define inf 99999 //define by you maximum
261 edges weight
262 vector<vector<int>> edges;
263 vector<int> dist;

```

```

264 vector<int> ancestor;
265 void BellmanFord(int start, int node){
266     dist[start] = 0;
267     for(int it = 0; it < node-1; it++){
268         for(int i = 0; i < node; i++){
269             for(int j = 0; j < node; j++){
270                 if(edges[i][j] != -1){
271                     if(dist[i] + edges[i][j]
272                         < dist[j]){
273                         dist[j] = dist[i] +
274                         edges[i][j];
275                         ancestor[j] = i;
276                     }
277                 }
278             }
279         }
280     }
281     for(int i = 0; i < node; i++) //
282     negative cycle detection
283     for(int j = 0; j < node; j++){
284         if(dist[i] + edges[i][j] < dist[
285             j]){
286             cout<<"Negative cycle!"<<
287             endl;
288             return;
289         }
290     }
291 }
292 int main(){
293     int node;
294     cin>>node;
295     edges.resize(node, vector<int>(node, inf))
296     ;
297     dist.resize(node, inf);
298     ancestor.resize(node, -1);
299     int a, b, d;
300     while(cin>>a>>b>>d){
301         /*input: source destination weight*/
302         if(a == -1 && b == -1 && d == -1)
303             break;
304         edges[a][b] = d;
305     }
306     int start;
307     cin>>start;
308     BellmanFord(start, node);
309     return 0;
310 }

```

5.2 BFS-queue

```

311 /*BFS - queue version*/
312 void BFS(vector<int> &result, vector<pair<
313     int, int>> edges, int node, int start)
314 {
315     vector<int> pass(node, 0);
316     queue<int> q;
317     queue<int> p;
318     q.push(start);
319     int count = 1;
320     vector<pair<int, int>> newedges;
321     while (!q.empty())
322     {

```

```

323         pass[q.front()] = 1;
324         for (int i = 0; i < edges.size(); i
325             ++){
326             if (edges[i].first == q.front()
327                 && pass[edges[i].second] ==
328                 0)
329             {
330                 p.push(edges[i].second);
331                 result[edges[i].second] =
332                 count;
333             }
334             else if (edges[i].second == q.
335                 front() && pass[edges[i].
336                 first] == 0)
337             {
338                 p.push(edges[i].first);
339                 result[edges[i].first] =
340                 count;
341             }
342             else
343                 newedges.push_back(edges[i])
344                 ;
345         }
346         edges = newedges;
347         newedges.clear();
348         q.pop();
349         if (q.empty() == true)
350         {
351             q = p;
352             queue<int> tmp;
353             p = tmp;
354             count++;
355         }
356     }
357 }
358 int main()
359 {
360     int node;
361     cin >> node;
362     vector<pair<int, int>> edges;
363     int a, b;
364     while (cin >> a >> b)
365     {
366         /*a = b = -1 means input edges ended
367         */
368         if (a == -1 && b == -1)
369             break;
370         edges.push_back(pair<int, int>(a, b)
371             );
372     }
373     vector<int> result(node, -1);
374     BFS(result, edges, node, 0);
375     return 0;
376 }

```

5.3 DFS-rec

```

377 /*DFS - Recursive version*/
378 map<pair<int, int>, int> edges;
379 vector<int> pass;
380 vector<int> route;

```



```

5 void DFS(int start){
6     pass[start] = 1;
7     map<pair<int,int>,int>::iterator iter;
8     for(iter = edges.begin(); iter != edges.
9         end(); iter++){
10         if((*iter).first.first == start &&
11            (*iter).second == 0 && pass[(*)
12            iter).first.second] == 0){
13             route.push_back((*iter).first.
14             second);
15             DFS((*iter).first.second);
16         }
17         else if((*iter).first.second ==
18             start && (*iter).second == 0 &&
19             pass[(*)iter).first.first] == 0){
20             route.push_back((*iter).first.
21             first);
22             DFS((*iter).first.first);
23         }
24     }
25 }
26
27 int main(){
28     int node;
29     cin >> node;
30     pass.resize(node,0);
31     int a,b;
32     while(cin >> a >> b){
33         if(a == -1 && b == -1)
34             break;
35         edges.insert(pair<pair<int,int>,int>
36             >(pair<int,int>(a,b),0));
37     }
38     int start;
39     cin >> start;
40     route.push_back(start);
41     DFS(start);
42     return 0;
43 }

```

5.4 Dijkstra

```

1 /*SPA - Dijkstra*/
2 #define inf INT_MAX
3 vector<vector<int>> > weight;
4 vector<int> ancestor;
5 vector<int> dist;
6 void dijkstra(int start){
7     priority_queue<pair<int,int>,vector<
8         pair<int,int>>,greater<pair<int,
9         int>>> pq;
10     pq.push(make_pair(0,start));
11     while(!pq.empty()){
12         int cur = pq.top().second;
13         pq.pop();
14         for(int i = 0; i < weight[cur].size
15             (); i++){
16             if(dist[i] > dist[cur] + weight[
17                 cur][i] && weight[cur][i] !=
18                 -1){
19                 dist[i] = dist[cur] + weight
20                 [cur][i];
21                 ancestor[i] = cur;

```

```

16         pq.push(make_pair(dist[i],i)
17         );
18     }
19 }
20 }
21 int main(){
22     int node;
23     cin >> node;
24     int a,b,d;
25     weight.resize(node,vector<int>(node,-1))
26     ;
27     while(cin >> a >> b >> d){
28         /*input: source destination weight*/
29         if(a == -1 && b == -1 && d == -1)
30             break;
31         weight[a][b] = d;
32     }
33     ancestor.resize(node,-1);
34     dist.resize(node,inf);
35     int start;
36     cin >> start;
37     dist[start] = 0;
38     dijkstra(start);
39     return 0;

```

5.5 Floyd-warshall

```

1 /*SPA - Floyd-Warshall*/
2 #define inf 99999
3 void floyd_warshall(vector<vector<int>>&
4     distance, vector<vector<int>>& ancestor,
5     int n){
6     for (int k = 0; k < n; k++){
7         for (int i = 0; i < n; i++){
8             for (int j = 0; j < n; j++){
9                 if(distance[i][k] + distance
10                    [k][j] < distance[i][j])
11                     distance[i][j] =
12                     distance[i][k] +
13                     distance[k][j];
14                 ancestor[i][j] =
15                 ancestor[k][j];
16             }
17         }
18     }
19 }
20
21 int main(){
22     int n;
23     cin >> n;
24     int a, b, d;
25     vector<vector<int>> distance(n, vector<
26         int>(n,99999));
27     vector<vector<int>> ancestor(n, vector<
28         int>(n,-1));
29     while(cin >> a >> b >> d){
30         if(a == -1 && b == -1 && d == -1)
31             break;
32         distance[a][b] = d;
33         ancestor[a][b] = a;

```

```

26     }
27     for (int i = 0; i < n; i++)
28         distance[i][i] = 0;
29     floyd_warshall(distance, ancestor, n);
30     /*Negative cycle detection*/
31     for (int i = 0; i < n; i++){
32         if(distance[i][i] < 0){
33             cout << "Negative cycle!" <<
34             endl;
35             break;
36         }
37     }
38     return 0;

```

5.6 Kruskal

```

1 /*mst - Kruskal*/
2 struct edges{
3     int from;
4     int to;
5     int weight;
6     friend bool operator < (edges a, edges b
7         ){
8         return a.weight > b.weight;
9     }
10 };
11 int find(int x,vector<int>& union_set){
12     if(x != union_set[x])
13         union_set[x] = find(union_set[x],
14         union_set);
15     return union_set[x];
16 }
17 void merge(int a,int b,vector<int>&
18     union_set){
19     int pa = find(a, union_set);
20     int pb = find(b, union_set);
21     if(pa != pb)
22         union_set[pa] = pb;
23 }
24 void kruskal(priority_queue<edges> pq,int n)
25 {
26     vector<int> union_set(n, 0);
27     for (int i = 0; i < n; i++)
28         union_set[i] = i;
29     int edge = 0;
30     int cost = 0; //evaluate cost of mst
31     while(!pq.empty() && edge < n - 1){
32         edges cur = pq.top();
33         int from = find(cur.from, union_set)
34         ;
35         int to = find(cur.to, union_set);
36         if(from != to){
37             merge(from, to, union_set);
38             edge += 1;
39             cost += cur.weight;
40         }
41         pq.pop();
42     }
43     if(edge < n-1)
44         cout << "No mst" << endl;
45     else
46         cout << cost << endl;

```

```

42 }
43 int main(){
44     int n;
45     cin >> n;
46     int a, b, d;
47     priority_queue<edges> pq;
48     while(cin >> a >> b >> d){
49         if(a == -1 && b == -1 && d == -1)
50             break;
51         edges tmp;
52         tmp.from = a;
53         tmp.to = b;
54         tmp.weight = d;
55         pq.push(tmp);
56     }
57     kruskal(pq, n);
58     return 0;
59 }

```

5.7 Prim

```

1 /*mst - Prim*/
2 #define inf 99999
3 struct edges{
4     int from;
5     int to;
6     int weight;
7     friend bool operator < (edges a, edges b
8         ){
9         return a.weight > b.weight;
10     }
11 };
12 void Prim(vector<vector<int>> gp,int n,int
13     start){
14     vector<bool> pass(n,false);
15     int edge = 0;
16     int cost = 0; //evaluate cost of mst
17     priority_queue<edges> pq;
18     for (int i = 0; i < n; i++){
19         if(gp[start][i] != inf){
20             edges tmp;
21             tmp.from = start;
22             tmp.to = i;
23             tmp.weight = gp[start][i];
24             pq.push(tmp);
25         }
26     }
27     pass[start] = true;
28     while(!pq.empty() && edge < n-1){
29         edges cur = pq.top();
30         pq.pop();
31         if(!pass[cur.to]){
32             for (int i = 0; i < n; i++){
33                 if(gp[cur.to][i] != inf){
34                     edges tmp;
35                     tmp.from = cur.to;
36                     tmp.to = i;
37                     tmp.weight = gp[cur.to][
38                         i];
39                     pq.push(tmp);

```

```

40     edge += 1;
41     cost += cur.weight;
42 }
43 }
44 if(edge < n-1)
45     cout << "No mst" << endl;
46 else
47     cout << cost << endl;
48 }
49 int main(){
50     int n;
51     cin >> n;
52     int a, b, d;
53     vector<vector<int>> gp(n,vector<int>(n,
54         inf));
55     while(cin>>a>>b>>d){
56         if(a == -1 && b == -1 && d == -1)
57             break;
58         if(gp[a][b] > d)
59             gp[a][b] = d;
60     }
61     Prim(gp,n,0);
62     return 0;

```

5.8 Union_find

```

1 int find(int x, vector<int> &union_set)
2 {
3     if (union_set[x] != x)
4         union_set[x] = find(union_set[x],
5             union_set); //compress path
6     return union_set[x];
7 }
8 void merge(int x, int y, vector<int> &
9     union_set, vector<int> &rank)
10 {
11     int rx, ry;
12     rx = find(x, union_set);
13     ry = find(y, union_set);
14     if (rx == ry)
15         return;
16     /*merge by rank -> always merge small
17     tree to big tree*/
18     if (rank[rx] > rank[ry])
19         union_set[ry] = rx;
20     else
21     {
22         union_set[rx] = ry;
23         if (rank[rx] == rank[ry])
24             ++rank[ry];
25     }
26 }
27 int main()
28 {
29     int node;
30     cin >> node; //Input Node number
31     vector<int> union_set(node, 0);
32     vector<int> rank(node, 0);
33     for (int i = 0; i < node; i++)
34         union_set[i] = i;
35     int edge;
36     cin >> edge; //Input Edge number

```

```

34     for (int i = 0; i < edge; i++)
35     {
36         int a, b;
37         cin >> a >> b;
38         merge(a, b, union_set, rank);
39     }
40     /*build party*/
41     vector<vector<int>> party(node, vector<
42         int>(0));
43     for (int i = 0; i < node; i++)
44         party[find(i, union_set)].push_back(
45             i);

```

6 Mathematics

6.1 Combination

```

1 /*input type string or vector*/
2 for (int i = 0; i < (1 << input.size()); ++i
3 )
4 {
5     string testCase = "";
6     for (int j = 0; j < input.size(); ++j)
7         if (i & (1 << j))
8             testCase += input[j];

```

6.2 Extended Euclidean

```

1 // ax + by = gcd(a,b)
2 pair<long long, long long> extgcd(long long
3     a, long long b)
4 {
5     if (b == 0)
6         return {1, 0};
7     long long k = a / b;
8     pair<long long, long long> p = extgcd(b,
9         a - k * b);
10    //cout << p.first << " " << p.second <<
11        endl;
12    //cout << "商數(k)= " << k << endl <<
13        endl;
14    return {p.second, p.first - k * p.second
15        };
16 }
17 int main()
18 {
19     int a, b;
20     cin >> a >> b;
21     pair<long long, long long> xy = extgcd(a
22         , b); //(x0,y0)
23     cout << xy.first << " " << xy.second <<
24         endl;
25     cout << xy.first << " * " << a << " + "
26         << xy.second << " * " << b << endl;

```

```

20     return 0;
21 }
22 // ax + by = gcd(a,b) * r
23 /*find |x|+|y| -> min*/
24 int main()
25 {
26     long long r, p, q; /*px+qy = r*/
27     int cases;
28     cin >> cases;
29     while (cases--)
30     {
31         cin >> r >> p >> q;
32         pair<long long, long long> xy =
33             extgcd(q, p); //(x0,y0)
34         long long ans = 0, tmp = 0;
35         double k, k1;
36         long long s, s1;
37         k = 1 - (double)(r * xy.first) / p;
38         s = round(k);
39         ans = llabs(r * xy.first + s * p) +
40             llabs(r * xy.second - s * q);
41         k1 = -(double)(r * xy.first) / p;
42         s1 = round(k1);
43         /*cout << k << endl << k1 << endl;
44         cout << s << endl << s1 << endl;
45         */
46         tmp = llabs(r * xy.first + s1 * p) +
47             llabs(r * xy.second - s1 * q);
48         ans = min(ans, tmp);
49         cout << ans << endl;
50     }
51     return 0;

```

6.3 Hex to Dec

```

1 int HextoDec(string num) //16 to 10
2 {
3     int base = 1;
4     int temp = 0;
5     for (int i = num.length() - 1; i >= 0; i
6         --)
7     {
8         if (num[i] == '0' && num[i] == '9')
9         {
10             temp += (num[i] - 48) * base;
11             base = base * 16;
12         }
13         else if (num[i] == 'A' && num[i] == 'F'
14             ')
15         {
16             temp += (num[i] - 55) * base;
17             base = base * 16;
18         }
19     }
20     return temp;
21 }
22 void DecToHex(int p_intValue) //10 to 16
23 {
24     char l_pCharRes = new char;
25     sprintf(l_pCharRes, "%X", p_intValue);
26     int l_intResult = stoi(l_pCharRes);

```

```

25     cout << l_pCharRes << endl;
26     return l_intResult;
27 }

```

6.4 Mod

```

1 int pow_mod(int a, int n, int m) // a ^ n
2     mod m;
3 { // a, n, m < 10 ^ 9
4     if (n == 0)
5         return 1;
6     int x = pow_mid(a, n / 2, m);
7     long long ans = (long long)x * x % m;
8     if (n % 2 == 1)
9         ans = ans * a % m;
10    return (int)ans;
11 }
12 // 加法: (a + b) % p = (a % p + b % p) % p;
13 // 減法: (a - b) % p = (a % p - b % p + p) %
14     p;
15 // 乘法: (a * b) % p = (a % p * b % p) % p;
16 // 次方: (a ^ b) % p = ((a % p) ^ b) % p;
17 // 加法結合律: ((a + b) % p + c) % p = (a +
18     (b + c)) % p;
19 // 乘法結合律: ((a * b) % p * c) % p = (a *
20     (b * c)) % p;
21 // 加法交換律: (a + b) % p = (b + a) % p;
22 // 乘法交換律: (a * b) % p = (b * a) % p;
23 // 結合律: ((a + b) % p * c) = ((a * c) % p
24     + (b * c) % p) % p;
25 // 如果 a ≡ b(mod m) · 我們會說 a,b 在模 m
26     下同餘。
27 // 整除性: a ≡ b(mod m) ⇔ c ⇔ m = a - b, c
28     ⇔ Z ⇔ a ≡ b (mod m) ⇔ m|a-b
29 // 遞移性: 若 a ≡ b (mod c), b ≡ d(mod c) 則
30     a ≡ d (mod c)
31 /*****基本運算****/
32 // a ≡ b (mod m) ⇔ { a ± c ≡ b ± d (mod m) }
33 // c ≡ d (mod m) ⇔ { a * c ≡ b * d (mod m) }
34 // 放大縮小模數: k ∈ Z+, a ≡ b (mod m) ⇔ k * a
35     ≡ k * b (mod k * m)

```

6.5 Permutation

```

1 // 全排列要先 sort !!!
2 // num -> vector or string
3 next_permutation(num.begin(), num.end());
4 prev_permutation(num.begin(), num.end());

```

6.6 PI

```

1 #define PI acos(-1)
2 #define PI M_PI
3 const double PI = atan2(0.0, -1.0);

```

6.7 Prime table

```

1 const int maxn = sqrt(INT_MAX);
2 vector<int> p;
3 bitset<maxn> is_notp;
4 void PrimeTable()
5 {
6     is_notp.reset();
7     is_notp[0] = is_notp[1] = 1;
8     for (int i = 2; i <= maxn; ++i)
9     {
10         if (!is_notp[i])
11             p.push_back(i);
12         for (int j = 0; j < (int)p.size(); ++j)
13         {
14             if (i * p[j] > maxn)
15                 break;
16             is_notp[i * p[j]] = 1;
17             if (i % p[j] == 0)
18                 break;
19         }
20     }
21 }

```

6.8 primeBOOL

```

1 // n < 4759123141    chk = [2, 7, 61]
2 // n < 1122004669633  chk = [2, 13, 23,
3 //                    1662803]
4 // n < 2^64          chk = [2, 325, 9375,
5 //                    28178, 450775, 9780504, 1795265022]
6 vector<long long> chk = {};
7 long long fmul(long long a, long long n,
8 long long mod)
9 {
10     long long ret = 0;
11     for (; n >= 1)
12     {
13         if (n & 1)
14             (ret += a) %= mod;
15         (a += a) %= mod;
16     }
17     return ret;
18 }
19 long long fpow(long long a, long long n,
20 long long mod)
21 {
22     long long ret = 1LL;
23     for (; n >= 1)
24     {
25         if (n & 1)
26             ret = fmul(ret, a, mod);
27         a = fmul(a, a, mod);
28     }
29 }

```

```

26     return ret;
27 }
28 bool check(long long a, long long u, long
29 long n, int t)
30 {
31     a = fpow(a, u, n);
32     if (a == 0)
33         return true;
34     if (a == 1 || a == n - 1)
35         return true;
36     for (int i = 0; i < t; ++i)
37     {
38         a = fmul(a, a, n);
39         if (a == 1)
40             return false;
41         if (a == n - 1)
42             return true;
43     }
44     return false;
45 }
46 bool is_prime(long long n)
47 {
48     if (n < 2)
49         return false;
50     if (n % 2 == 0)
51         return n == 2;
52     long long u = n - 1;
53     int t = 0;
54     for (; u & 1; u >>= 1, ++t)
55         ;
56     for (long long i : chk)
57     {
58         if (!check(i, u, n, t))
59             return false;
60     }
61     return true;
62 }
63 // if (is_prime(int num)) // true == prime
64 // 反之亦然

```

6.9 二分逼近法

```

1 #define eps 1e-14
2 void half_interval()
3 {
4     double L = 0, R = /*區間*/, M;
5     while (R - L >= eps)
6     {
7         M = (R + L) / 2;
8         if (/*函數*/ > /*方程式目標*/)
9             L = M;
10        else
11            R = M;
12    }
13    printf("%.3lf\n", R);
14 }

```

6.10 四則運算

```

1 string s = ""; //開頭是負號要補0
2 long long int DFS(int le, int ri) // (0,
3 string final index)
4 {
5     int c = 0;
6     for (int i = ri; i >= le; i--)
7     {
8         if (s[i] == ')')
9             c++;
10        if (s[i] == '(')
11            c--;
12        if (s[i] == '+' && c == 0)
13            return DFS(le, i - 1) + DFS(i +
14            1, ri);
15        if (s[i] == '-' && c == 0)
16            return DFS(le, i - 1) - DFS(i +
17            1, ri);
18    }
19    for (int i = ri; i >= le; i--)
20    {
21        if (s[i] == ')')
22            c++;
23        if (s[i] == '(')
24            c--;
25        if (s[i] == '*' && c == 0)
26            return DFS(le, i - 1) * DFS(i +
27            1, ri);
28        if (s[i] == '/' && c == 0)
29            return DFS(le, i - 1) / DFS(i +
30            1, ri);
31        if (s[i] == '%' && c == 0)
32            return DFS(le, i - 1) % DFS(i +
33            1, ri);
34    }
35    if ((s[le] == '(' && (s[ri] == ')'))
36        return DFS(le + 1, ri - 1); //去除刮
37        號
38    if (s[le] == '+' && s[ri] == '+')
39        return DFS(le + 1, ri - 1); //去除左
40        右兩邊空格
41    if (s[le] == '-')
42        return DFS(le + 1, ri); //去除左邊空
43        格
44    if (s[ri] == '-')
45        return DFS(le, ri - 1); //去除右邊空
46        格
47    long long int num = 0;
48    for (int i = le; i <= ri; i++)
49        num = num * 10 + s[i] - '0';
50    return num;
51 }

```

6.11 數字乘法組合

```

1 void dfs(int j, int old, int num, vector<int>
2 > com, vector<vector<int>> &ans)
3 {
4     for (int i = j; i <= sqrt(num); i++)
5     {
6         if (old == num)
7             com.clear();
8     }
9 }

```

```

7     if (num % i == 0)
8     {
9         vector<int> a;
10        a = com;
11        a.push_back(i);
12        finds(i, old, num / i, a, ans);
13        a.push_back(num / i);
14        ans.push_back(a);
15    }
16 }
17 }
18 vector<vector<int>> ans;
19 vector<int> zero;
20 dfs(2, num, num, zero, ans);
21 /*num 為 input 數字*/
22 for (int i = 0; i < ans.size(); i++)
23 {
24     for (int j = 0; j < ans[i].size() - 1; j
25         ++
26         cout << ans[i][j] << " ";
27     cout << ans[i][ans[i].size() - 1] <<
28     endl;
29 }

```

6.12 數字加法組合

```

1 void recur(int i, int n, int m, vector<int>
2 &out, vector<vector<int>> &ans)
3 {
4     if (n == 0)
5     {
6         for (int i : out)
7             if (i > m)
8                 return;
9         ans.push_back(out);
10    }
11    for (int j = i; j <= n; j++)
12    {
13        out.push_back(j);
14        recur(j, n - j, m, out, ans);
15        out.pop_back();
16    }
17 }
18 vector<vector<int>> ans;
19 vector<int> zero;
20 recur(1, num, num, zero, ans);
21 // num 為 input 數字
22 for (int i = 0; i < ans.size(); i++)
23 {
24     for (int j = 0; j < ans[i].size() - 1; j
25         ++
26         cout << ans[i][j] << " ";
27     cout << ans[i][ans[i].size() - 1] <<
28     endl;
29 }

```


6.13 羅馬數字

```

1 int romanToInt(string s)
2 {
3     unordered_map<char, int> T;
4     T['I'] = 1;
5     T['V'] = 5;
6     T['X'] = 10;
7     T['L'] = 50;
8     T['C'] = 100;
9     T['D'] = 500;
10    T['M'] = 1000;
11
12    int sum = T[s.back()];
13    for (int i = s.length() - 2; i >= 0; --i)
14    {
15        if (T[s[i]] < T[s[i + 1]])
16            sum -= T[s[i]];
17        else
18            sum += T[s[i]];
19    }
20    return sum;
21 }

```

6.14 質因數分解

```

1 void primeFactorization(int n) // 配合質數表
2 {
3     for (int i = 0; i < (int)p.size(); ++i)
4     {
5         if (p[i] * p[i] > n)
6             break;
7         if (n % p[i])
8             continue;
9         cout << p[i] << ' ';
10        while (n % p[i] == 0)
11            n /= p[i];
12    }
13    if (n != 1)
14        cout << n << ' ';
15    cout << '\n';
16 }

```

7 Other

7.1 heap sort

```

1 void MaxHeapify(vector<int> &array, int root
2     , int length)
3 {
4     int left = 2 * root,
5         right = 2 * root + 1,
6         largest;
7     if (left <= length && array[left] >
8         array[root])
9         largest = left;

```

```

8     else
9         largest = root;
10    if (right <= length && array[right] >
11        array[largest])
12        largest = right;
13    if (largest != root)
14    {
15        swap(array[largest], array[root]);
16        MaxHeapify(array, largest, length);
17    }
18 }
19 void HeapSort(vector<int> &array)
20 {
21     array.insert(array.begin(), 0);
22     for (int i = (int)array.size() / 2; i >=
23         1; i--)
24         MaxHeapify(array, i, (int)array.size
25             () - 1);
26     int size = (int)array.size() - 1;
27     for (int i = (int)array.size() - 1; i >=
28         2; i--)
29     {
30         swap(array[1], array[i]);
31         size--;
32         MaxHeapify(array, 1, size);
33     }
34     array.erase(array.begin());
35 }

```

7.2 Merge sort

```

1 void Merge(vector<int> &arr, int front, int
2     mid, int end)
3 {
4     vector<int> LeftSub(arr.begin() + front,
5         arr.begin() + mid + 1);
6     vector<int> RightSub(arr.begin() + mid +
7         1, arr.begin() + end + 1);
8     LeftSub.insert(LeftSub.end(), INT_MAX);
9     RightSub.insert(RightSub.end(), INT_MAX);
10    ;
11    int idxLeft = 0, idxRight = 0;
12
13    for (int i = front; i <= end; i++)
14    {
15        if (LeftSub[idxLeft] <= RightSub[
16            idxRight])
17        {
18            arr[i] = LeftSub[idxLeft];
19            idxLeft++;
20        }
21        else
22        {
23            arr[i] = RightSub[idxRight];
24            idxRight++;
25        }
26    }
27 }
28 void MergeSort(vector<int> &arr, int front,
29     int end)
30 {
31     // front = 0 , end = arr.size() - 1

```

```

27     if (front < end)
28     {
29         int mid = (front + end) / 2;
30         MergeSort(arr, front, mid);
31         MergeSort(arr, mid + 1, end);
32         Merge(arr, front, mid, end);
33     }
34 }

```

7.3 Quick

```

1 int Partition(vector<int> &arr, int front,
2     int end)
3 {
4     int pivot = arr[end];
5     int i = front - 1;
6     for (int j = front; j < end; j++)
7     {
8         if (arr[j] < pivot)
9         {
10            i++;
11            swap(arr[i], arr[j]);
12        }
13        i++;
14        swap(arr[i], arr[end]);
15        return i;
16    }
17 }
18 void QuickSort(vector<int> &arr, int front,
19     int end)
20 {
21     // front = 0 , end = arr.size() - 1
22     if (front < end)
23     {
24         int pivot = Partition(arr, front,
25             end);
26         QuickSort(arr, front, pivot - 1);
27         QuickSort(arr, pivot + 1, end);
28     }
29 }

```

7.4 Weighted Job Scheduling

```

1 struct Job
2 {
3     int start, finish, profit;
4 };
5 bool jobComparataor(Job s1, Job s2)
6 {
7     return (s1.finish < s2.finish);
8 }
9 int latestNonConflict(Job arr[], int i)
10 {
11     for (int j = i - 1; j >= 0; j--)
12     {
13         if (arr[j].finish <= arr[i].start)
14             return j;
15     }
16     return -1;
17 }

```

```

18 int findMaxProfit(Job arr[], int n)
19 {
20     sort(arr, arr + n, jobComparataor);
21     int *table = new int[n];
22     table[0] = arr[0].profit;
23     for (int i = 1; i < n; i++)
24     {
25         int inclProf = arr[i].profit;
26         int l = latestNonConflict(arr, i);
27         if (l != -1)
28             inclProf += table[l];
29         table[i] = max(inclProf, table[i -
30             1]);
31     }
32     int result = table[n - 1];
33     delete[] table;
34     return result;
35 }

```

7.5 數獨解法

```

1 int getSquareIndex(int row, int column, int
2     n)
3 {
4     return row / n * n + column / n;
5 }
6 bool backtracking(vector<vector<int>> &board
7     , vector<vector<bool>> &rows, vector<
8     vector<bool>> &cols,
9     vector<vector<bool>> &boxes
10     , int index, int n)
11 {
12     int n2 = n * n;
13     int rowNum = index / n2, colNum = index
14         % n2;
15     if (index >= n2 * n2)
16         return true;
17     if (board[rowNum][colNum] != 0)
18         return backtracking(board, rows,
19             cols, boxes, index + 1, n);
20     for (int i = 1; i <= n2; i++)
21     {
22         if (!rows[rowNum][i] && !cols[colNum
23             ][i] && !boxes[getSquareIndex(
24                 rowNum, colNum, n)][i])
25         {
26             rows[rowNum][i] = true;
27             cols[colNum][i] = true;
28             boxes[getSquareIndex(rowNum,
29                 colNum, n)][i] = true;
30             board[rowNum][colNum] = i;
31             if (backtracking(board, rows,
32                 cols, boxes, index + 1, n))
33                 return true;
34             board[rowNum][colNum] = 0;
35             rows[rowNum][i] = false;
36             cols[colNum][i] = false;
37             boxes[getSquareIndex(rowNum,
38                 colNum, n)][i] = false;

```

```

31     }
32 }
33 return false;
34 }
35 /*用法 main*/
36 int n = sqrt(數獨邊長大小) /*e.g. 9*9 n=3*/
37 vector<vector<int>> board(n * n + 1, vector<
38     int>(n * n + 1, 0));
39 vector<vector<bool>> isRow(n * n + 1, vector<
40     bool>(n * n + 1, false));
41 vector<vector<bool>> isColumn(n * n + 1,
42     vector<bool>(n * n + 1, false));
43 vector<vector<bool>> isSquare(n * n + 1,
44     vector<bool>(n * n + 1, false));
45 for (int i = 0; i < n * n; ++i)
46 {
47     for (int j = 0; j < n * n; ++j)
48     {
49         int number;
50         cin >> number;
51         board[i][j] = number;
52         if (number == 0)
53             continue;
54         isRow[i][number] = true;
55         isColumn[j][number] = true;
56         isSquare[getSquareIndex(i, j, n)][
57             number] = true;
58     }
59 }
60 if (backtracking(board, isRow, isColumn,
61     isSquare, 0, n))
62     /*有解答*/
63 else
64     /*解答*/

```

8 String

8.1 KMP

```

1 // 用在一個 s 內查找一個詞 w 的出現位置
2 void ComputePrefix(string s, int next[])
3 {
4     int n = s.length();
5     int q, k;
6     next[0] = 0;
7     for (k = 0, q = 1; q < n; q++)
8     {
9         while (k > 0 && s[k] != s[q])
10             k = next[k];
11         if (s[k] == s[q])
12             k++;
13         next[q] = k;
14     }
15 }
16 void KMPMatcher(string text, string pattern)
17 {
18     int n = text.length();
19     int m = pattern.length();
20     int next[pattern.length()];

```

```

21 ComputePrefix(pattern, next);
22 for (int i = 0, q = 0; i < n; i++)
23 {
24     while (q > 0 && pattern[q] != text[i]
25         ])
26         q = next[q];
27     if (pattern[q] == text[i])
28         q++;
29     if (q == m)
30     {
31         cout << "Pattern occurs with
32             shift " << i - m + 1 << endl;
33         ;
34         q = 0;
35     }
36 }
37 // string s = "abcdabcdeabcd";
38 // string p = "bcd";
39 // KMPMatcher(s, p);
40 // cout << endl;

```

8.2 Min Edit Distance

```

1 int EditDistance(string a, string b)
2 {
3     vector<vector<int>> dp(a.size() + 1,
4         vector<int>(b.size() + 1, 0));
5     int m = a.length(), n = b.length();
6     for (int i = 0; i < m + 1; i++)
7     {
8         for (int j = 0; j < n + 1; j++)
9         {
10             if (i == 0)
11                 dp[i][j] = j;
12             else if (j == 0)
13                 dp[i][j] = i;
14             else if (a[i - 1] == b[j - 1])
15                 dp[i][j] = dp[i - 1][j - 1];
16             else
17                 dp[i][j] = 1 + min(min(dp[i]
18                     - 1][j], dp[i][j - 1]),
19                     dp[i - 1][j - 1]);
20         }
21     }
22     return dp[m][n];
23 }

```

8.3 Sliding window

```

1 string minWindow(string s, string t)
2 {
3     unordered_map<char, int> letterCnt;
4     for (int i = 0; i < t.length(); i++)
5         letterCnt[t[i]]++;
6     int minLength = INT_MAX, minStart = -1;
7     int left = 0, matchCnt = 0;
8     for (int i = 0; i < s.length(); i++)
9     {

```

```

10         if (--letterCnt[s[i]] >= 0)
11             matchCnt++;
12         while (matchCnt == t.length())
13         {
14             if (i - left + 1 < minLength)
15             {
16                 minLength = i - left + 1;
17                 minStart = left;
18             }
19             if (++letterCnt[s[left]] > 0)
20                 matchCnt--;
21             left++;
22         }
23     }
24     return minLength == INT_MAX ? "" : s.
25         substr(minStart, minLength);

```

8.4 Split

```

1 vector<string> mysplit(const string &str,
2     const string &delim)
3 {
4     vector<string> res;
5     if (" " == str)
6         return res;
7     char *strs = new char[str.length() + 1];
8     char *d = new char[delim.length() + 1];
9     strcpy(strs, str.c_str());
10    strcpy(d, delim.c_str());
11    char *p = strtok(strs, d);
12    while (p)
13    {
14        string s = p;
15        res.push_back(s);
16        p = strtok(NULL, d);
17    }
18    return res;
19 }

```

9 data structure

9.1 Bigint

```

1 //台大
2 struct Bigint
3 {
4     static const int LEN = 60;
5     static const int BIGMOD = 10000;
6     int s;
7     int vl, v[LEN];
8     // vector<int> v;
9     Bigint() : s(1) { vl = 0; }
10    Bigint(long long a)
11    {
12        s = 1;
13        vl = 0;

```

```

14        if (a < 0)
15        {
16            s = -1;
17            a = -a;
18        }
19        while (a)
20        {
21            push_back(a % BIGMOD);
22            a /= BIGMOD;
23        }
24    }
25    Bigint(string str)
26    {
27        s = 1;
28        vl = 0;
29        int stPos = 0, num = 0;
30        if (!str.empty() && str[0] == '-')
31        {
32            stPos = 1;
33            s = -1;
34        }
35        for (int i = str.length() - 1, q =
36            1; i >= stPos; i--)
37        {
38            num += (str[i] - '0') * q;
39            if ((q *= 10) >= BIGMOD)
40            {
41                push_back(num);
42                num = 0;
43                q = 1;
44            }
45        }
46        if (num)
47            push_back(num);
48        n();
49    }
50    int len() const
51    {
52        return vl; //return SZ(v);
53    }
54    bool empty() const { return len() == 0; }
55    void push_back(int x)
56    {
57        v[vl++] = x; //v.PB(x);
58    }
59    void pop_back()
60    {
61        vl--; //v.pop_back();
62    }
63    int back() const
64    {
65        return v[vl - 1]; //return v.back();
66    }
67    void n()
68    {
69        while (!empty() && !back())
70            pop_back();
71    }
72    void resize(int nl)
73    {
74        vl = nl; //v.resize(nl);
75        fill(v, v + vl, 0); //fill(ALL(v),
76            0);
77    }
78    void print() const

```

```

77 {
78     if (empty())
79     {
80         putchar('0');
81         return;
82     }
83     if (s == -1)
84         putchar('-');
85     printf("%d", back());
86     for (int i = len() - 2; i >= 0; i--)
87         printf("%.4d", v[i]);
88 }
89 friend std::ostream &operator<<(std::
90     ostream &out, const Bigint &a)
91 {
92     if (a.empty())
93     {
94         out << "0";
95         return out;
96     }
97     if (a.s == -1)
98         out << "-";
99     out << a.back();
100     for (int i = a.len() - 2; i >= 0; i
101         --)
102     {
103         char str[10];
104         snprintf(str, 5, "%.4d", a.v[i])
105         ;
106         out << str;
107     }
108     return out;
109 }
110 int cp3(const Bigint &b) const
111 {
112     if (s != b.s)
113         return s - b.s;
114     if (s == -1)
115         return -(*this).cp3(-b);
116     if (len() != b.len())
117         return len() - b.len(); //int
118     for (int i = len() - 1; i >= 0; i--)
119         if (v[i] != b.v[i])
120             return v[i] - b.v[i];
121     return 0;
122 }
123 bool operator<(const Bigint &b) const
124 {
125     return cp3(b) < 0;
126 }
127 bool operator<=(const Bigint &b) const
128 {
129     return cp3(b) <= 0;
130 }
131 bool operator==(const Bigint &b) const
132 {
133     return cp3(b) == 0;
134 }
135 bool operator!=(const Bigint &b) const
136 {
137     return cp3(b) != 0;
138 }
139 bool operator>(const Bigint &b) const
140 {
141     return cp3(b) > 0;
142 }

```

```

140 bool operator>=(const Bigint &b) const
141 {
142     return cp3(b) >= 0;
143 }
144 Bigint operator-(const
145 {
146     Bigint r = (*this);
147     r.s = -r.s;
148     return r;
149 }
150 Bigint operator+(const Bigint &b) const
151 {
152     if (s == -1)
153         return -(-(*this) + (-b));
154     if (b.s == -1)
155         return (*this) - (-b);
156     Bigint r;
157     int nl = max(len(), b.len());
158     r.resize(nl + 1);
159     for (int i = 0; i < nl; i++)
160     {
161         if (i < len())
162             r.v[i] += v[i];
163         if (i < b.len())
164             r.v[i] += b.v[i];
165         if (r.v[i] >= BIGMOD)
166         {
167             r.v[i + 1] += r.v[i] /
168             BIGMOD;
169             r.v[i] %= BIGMOD;
170         }
171     }
172     r.n();
173     return r;
174 }
175 Bigint operator-(const Bigint &b) const
176 {
177     if (s == -1)
178         return -(-(*this) - (-b));
179     if (b.s == -1)
180         return (*this) + (-b);
181     if ((*this) < b)
182         return -(b - (*this));
183     Bigint r;
184     r.resize(len());
185     for (int i = 0; i < len(); i++)
186     {
187         r.v[i] += v[i];
188         if (i < b.len())
189             r.v[i] -= b.v[i];
190         if (r.v[i] < 0)
191         {
192             r.v[i] += BIGMOD;
193             r.v[i + 1]--;
194         }
195     }
196     r.n();
197     return r;
198 }
199 Bigint operator*(const Bigint &b)
200 {
201     Bigint r;
202     r.resize(len() + b.len() + 1);
203     r.s = s * b.s;
204     for (int i = 0; i < len(); i++)

```

```

205     for (int j = 0; j < b.len(); j
206         ++)
207     {
208         r.v[i + j] += v[i] * b.v[j];
209         if (r.v[i + j] >= BIGMOD)
210         {
211             r.v[i + j + 1] += r.v[i
212             + j] / BIGMOD;
213             r.v[i + j] %= BIGMOD;
214         }
215     }
216     r.n();
217     return r;
218 }
219 Bigint operator/(const Bigint &b)
220 {
221     Bigint r;
222     r.resize(max(1, len() - b.len() + 1)
223     );
224     int oriS = s;
225     Bigint b2 = b; // b2 = abs(b)
226     s = b2.s = r.s = 1;
227     for (int i = r.len() - 1; i >= 0; i
228         --)
229     {
230         int d = 0, u = BIGMOD - 1;
231         while (d < u)
232         {
233             int m = (d + u + 1) >> 1;
234             r.v[i] = m;
235             if ((r * b2) > (*this))
236                 u = m - 1;
237             else
238                 d = m;
239         }
240         r.v[i] = d;
241     }
242     s = oriS;
243     r.s = s * b.s;
244     r.n();
245     return r;
246 }
247 Bigint operator%(const Bigint &b)
248 {
249     return (*this) - (*this) / b * b;
250 }

```

9.2 matirx

```

1 template <typename T>
2 struct Matrix
3 {
4     using rt = std::vector<T>;
5     using mt = std::vector<rt>;
6     using matrix = Matrix<T>;
7     int r, c; // [r][c]
8     mt m;
9     Matrix(int r, int c) : r(r), c(c), m(r,
10         rt(c)) {}
11     Matrix(mt a) { m = a, r = a.size(), c =
12         a[0].size(); }

```

```

13     rt &operator[](int i) { return m[i]; }
14     matrix operator+(const matrix &a)
15     {
16         matrix rev(r, c);
17         for (int i = 0; i < r; ++i)
18             for (int j = 0; j < c; ++j)
19                 rev[i][j] = m[i][j] + a.m[i
20                 ][j];
21         return rev;
22 }
23 matrix operator-(const matrix &a)
24 {
25     matrix rev(r, c);
26     for (int i = 0; i < r; ++i)
27         for (int j = 0; j < c; ++j)
28             rev[i][j] = m[i][j] - a.m[i
29             ][j];
30     return rev;
31 }
32 matrix operator*(const matrix &a)
33 {
34     matrix rev(r, a.c);
35     matrix tmp(a.c, a.r);
36     for (int i = 0; i < a.r; ++i)
37         for (int j = 0; j < a.c; ++j)
38             tmp[j][i] = a.m[i][j];
39     for (int i = 0; i < r; ++i)
40         for (int j = 0; j < a.c; ++j)
41             for (int k = 0; k < c; ++k)
42                 rev.m[i][j] += m[i][k] *
43                 tmp[j][k];
44     return rev;
45 }
46 bool inverse() //逆矩陣判斷
47 {
48     Matrix t(r, r + c);
49     for (int y = 0; y < r; y++)
50     {
51         t.m[y][c + y] = 1;
52         for (int x = 0; x < c; ++x)
53             t.m[y][x] = m[y][x];
54     }
55     if (!t.gas())
56         return false;
57     for (int y = 0; y < r; y++)
58         for (int x = 0; x < c; ++x)
59             m[y][x] = t.m[y][c + x] / t.
60             m[y][y];
61     return true;
62 }
63 T gas() //行列式
64 {
65     vector<T> lazy(r, 1);
66     bool sign = false;
67     for (int i = 0; i < r; ++i)
68     {
69         if (m[i][i] == 0)
70         {
71             int j = i + 1;
72             while (j < r && !m[j][i])
73                 j++;
74             if (j == r)
75                 continue;
76             m[i].swap(m[j]);
77             sign = !sign;

```

```

72     }
73     for (int j = 0; j < r; ++j)
74     {
75         if (i == j)
76             continue;
77         lazy[j] = lazy[j] * m[i][i];
78         T mx = m[j][i];
79         for (int k = 0; k < c; ++k)
80             m[j][k] = m[j][k] * m[i]
81                 [i] - m[i][k] * mx;
82     }
83     T det = sign ? -1 : 1;
84     for (int i = 0; i < r; ++i)
85     {
86         det = det * m[i][i];
87         det = det / lazy[i];
88         for (auto &j : m[i])
89             j /= lazy[i];
90     }
91     return det;
92 }
93 };

```

9.3 Trie

```

1 // biginter字典數
2 struct BigInteger{
3     static const int BASE = 100000000;
4     static const int WIDTH = 8;
5     vector<int> s;
6     BigInteger(long long num = 0){
7         *this = num;
8     }
9     BigInteger operator = (long long num){
10        s.clear();
11        do{
12            s.push_back(num % BASE);
13            num /= BASE;
14        }while(num > 0);
15        return *this;
16    }
17    BigInteger operator = (const string& str
18    ){
19        s.clear();
20        int x, len = (str.length() - 1) /
21            WIDTH + 1;
22        for(int i = 0; i < len; i++){
23            int end = str.length() - i*WIDTH
24                ;
25            int start = max(0, end-WIDTH);
26            sscanf(str.substr(start, end-
27                start).c_str(), "%d", &x);
28            s.push_back(x);
29        }
30        return *this;
31    }
32    BigInteger operator + (const BigInteger&
33        b) const{
34        BigInteger c;
35        c.s.clear();
36        for(int i = 0, g = 0; i++){

```

```

33         if(g == 0 && i >= s.size() && i
34             >= b.s.size()) break;
35         int x = g;
36         if(i < s.size()) x+=s[i];
37         if(i < b.s.size()) x+=b.s[i];
38         c.s.push_back(x % BASE);
39         g = x / BASE;
40     }
41     return c;
42 }
43 };
44 ostream& operator << (ostream &out, const
45     BigInteger& x){
46     out << x.s.back();
47     for(int i = x.s.size()-2; i >= 0; i--){
48         char buf[20];
49         sprintf(buf, "%08d", x.s[i]);
50         for(int j = 0; j < strlen(buf); j++){
51             out << buf[j];
52         }
53     }
54     return out;
55 }
56 istream& operator >> (istream &in,
57     BigInteger& x){
58     string s;
59     if(!(in >> s))
60         return in;
61     x = s;
62     return in;
63 }
64 struct Trie{
65     int c[5000005][10];
66     int val[5000005];
67     int sz;
68     int getIndex(char c){
69         return c - '0';
70     }
71     void init(){
72         memset(c[0], 0, sizeof(c[0]));
73         memset(val, -1, sizeof(val));
74         sz = 1;
75     }
76     void insert(BigInteger x, int v){
77         int u = 0;
78         int max_len_count = 0;
79         int firstNum = x.s.back();
80         char firstBuf[20];
81         sprintf(firstBuf, "%d", firstNum);
82         for(int j = 0; j < strlen(firstBuf);
83             j++){
84             int index = getIndex(firstBuf[j]
85                 );
86             if(!c[u][index]){
87                 memset(c[sz], 0, sizeof(c[
88                     sz]));
89                 val[sz] = v;
90                 c[u][index] = sz++;
91             }
92             u = c[u][index];
93             max_len_count++;
94         }
95     }
96 }

```

```

21     for(int i = x.s.size()-2; i >= 0; i
22         --){
23         char buf[20];
24         sprintf(buf, "%08d", x.s[i]);
25         for(int j = 0; j < strlen(buf)
26             && max_len_count < 50; j++){
27             int index = getIndex(buf[j])
28                 ;
29             if(!c[u][index]){
30                 memset(c[sz], 0, sizeof
31                     (c[sz]));
32                 val[sz] = v;
33                 c[u][index] = sz++;
34             }
35             u = c[u][index];
36             max_len_count++;
37         }
38     }
39     if(max_len_count >= 50){
40         break;
41     }
42 }
43 }
44 int find(const char* s){
45     int u = 0;
46     int n = strlen(s);
47     for(int i = 0; i < n; ++i)
48     {
49         int index = getIndex(s[i]);
50         if(!c[u][index]){
51             return -1;
52         }
53         u = c[u][index];
54     }
55     return val[u];
56 }
57 }

```

9.4 分數

```

1 typedef long long ll;
2 struct fraction
3 {
4     ll n, d;
5     fraction(const ll &n = 0, const ll &d =
6         1) : n(_n), d(_d)
7     {
8         ll t = __gcd(n, d);
9         n /= t, d /= t;
10        if (d < 0)
11            n = -n, d = -d;
12    }
13    fraction operator-() const
14    {
15        return fraction(-n, d);
16    }
17    fraction operator+(const fraction &b)
18        const
19    {
20        return fraction(n * b.d + b.n * d, d * b
21            .d);
22    }
23    fraction operator-(const fraction &b)
24        const

```

TO DO WRITING NOT THINKING

Contents

1 Basic	1	2.6 LIS	1	5.4 Dijkstra	6	6.14 質因數分解	9
1.1 data range	1	2.7 LPS	1	5.5 Floyd-warshall	6	7 Other	9
1.2 IO_fast	1	2.8 Max_subarray	2	5.6 Kruskal	6	7.1 heap sort	9
2 DP	1	2.9 Money problem	2	5.7 Prim	6	7.2 Merge sort	9
2.1 Knapsack Bounded	1	3 Flow & matching	2	5.8 Union_find	7	7.3 Quick	9
2.2 Knapsack sample	1	3.1 Edmonds_karp	2	6 Mathematics	7	7.4 Weighted Job Scheduling	9
2.3 Knapsack Unbounded	1	3.2 maximum_matching	2	6.1 Combination	7	7.5 數獨解法	9
2.4 LCIS	1	3.3 MFlow Model	2	6.2 Extended Euclidean	7	8 String	10
2.5 LCS	1	4 Geometry	3	6.3 Hex to Dec	7	8.1 KMP	10
		4.1 Line	3	6.4 Mod	7	8.2 Min Edit Distance	10
		4.2 Point	3	6.5 Permutation	7	8.3 Sliding window	10
		4.3 Polygon	4	6.6 PI	7	8.4 Split	10
		4.4 Triangle	5	6.7 Prime table	8	9 data structure	10
		5 Graph	5	6.8 primeBOOL	8	9.1 Bigint	10
		5.1 Bellman-Ford	5	6.9 二分逼近法	8	9.2 matirx	11
		5.2 BFS-queue	5	6.10 四則運算	8	9.3 Trie	12
		5.3 DFS-rec	5	6.11 數字乘法組合	8	9.4 分數	12
				6.12 數字加法組合	8		
				6.13 羅馬數字	9		