

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 KMP

```
1 void ComputePrefix(string s, int next[])
2 {
3     int n = s.length();
4     int q, k;
5     next[0] = 0;
6     for (k = 0, q = 1; q < n; q++)
7     {
8         while (k > 0 && s[k] != s[q])
9             k = next[k];
10        if (s[k] == s[q])
11            k++;
12        next[q] = k;
13    }
14 }
15 void KMPMatcher(string text, string pattern)
16 {
17     int n = text.length();
18     int m = pattern.length();
19     int next[pattern.length()];
20     ComputePrefix(pattern, next);
21
22     for (int i = 0, q = 0; i < n; i++)
23     {
24         while (q > 0 && pattern[q] != text[i])
25             q = next[q];
26         if (pattern[q] == text[i])
27             q++;
28         if (q == m)
29         {
30             cout << "Pattern occurs with
31               shift " << i - m + 1 << endl;
32             ;
33         }
34     }
35 }
```

```
31     q = 0;
32 }
33 }
34 }
35 // string s = "abcdabcdebc";
36 // string p = "bcd";
37 // KMPMatcher(s, p);
38 // cout << endl;
```

2.2 LCS

```
1 int LCS(vector<int> Ans, vector<int> num) //
2   Ans 跟 num 都要 index 從1開始放
3 {
4     vector<vector<int>> LCS(N + 1, vector<
5       int>(N + 1, 0));
6     for (int i = 1; i <= N; ++i)
7     {
8         for (int j = 1; j <= N; ++j)
9         {
10            if (Ans[i] == num[j])
11                LCS[i][j] = LCS[i - 1][j - 1] + 1;
12            else
13                LCS[i][j] = max(LCS[i - 1][j],
14                               LCS[i][j - 1]);
15        }
16    }
17    // printf("%d\n", LCS[N][N]);
18    return LCS[N][N];
19    //列印 LCS
20    vector<int> k;
21    while (n && m)
22    {
23        if (LCS[n][m] != max(LCS[n - 1][m],
24                             LCS[n][m - 1]))
25        {
26            k.push_back(arr1[n]);
27            n--;
28            m--;
29        }
30        else if (LCS[n][m] == LCS[n - 1][m])
31            n--;
32        else if (LCS[n][m] == LCS[n][m - 1])
33            m--;
34    }
35    reverse(k.begin(), k.end());
36 }
```

2.3 LIC

```
1 void getMaxElementAndPos(vector<int> &LISTbl
2   , vector<int> &LISlen, int tNum,
3   int tlen, int tStart, int &num, int &pos
4   )
5 {
6     int max = numeric_limits<int>::min();
7     int maxPos;
```

```
for (int i = tStart; i >= 0; i--)
{
    if (LISlen[i] == tlen && LISTbl[i] <
        tNum)
    {
        if (LISTbl[i] > max)
        {
            max = LISTbl[i];
            maxPos = i;
        }
    }
    num = max;
    pos = maxPos;
}
int LIS(vector<int> &LISTbl)
{
    if (LISTbl.size() == 0)
        return 0;
    vector<int> LISlen(LISTbl.size(), 1);
    for (int i = 1; i < LISTbl.size(); i++)
    {
        for (int j = 0; j < i; j++)
        {
            if (LISTbl[j] < LISTbl[i])
                LISlen[i] = max(LISlen[i],
                                LISlen[j] + 1);
        }
    }
    int maxlen = *max_element(LISlen.begin(),
                               LISlen.end());
    int num, pos;
    vector<int> buf;
    getMaxElementAndPos(LISTbl, LISlen,
                        numeric_limits<int>
                          >::max(),
                        maxlen, LISTbl.size()
                          - 1, num, pos);
    buf.push_back(num);
    for (int len = maxlen - 1; len >= 1; len
        --)
    {
        int tnum = num;
        int tpos = pos;
        getMaxElementAndPos(LISTbl, LISlen,
                            tnum, len, tpos
                              - 1, num,
                              pos);
        buf.push_back(num);
    }
    reverse(buf.begin(), buf.end());
    for (int k = 0; k < buf.size(); k++) //
        列印
    {
        if (k == buf.size() - 1)
            cout << buf[k] << endl;
        else
            cout << buf[k] << ",";
    }
    return maxlen;
}
```

2.4 LPS

```
1 void LPS(string s)
2 {
3     int maxlen = 0, l, r;
4     int n = s.length();
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;
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
34 }
```

2.5 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], local_max +
7           nums[i]);
8         global_max = max(local_max,
9           global_max);
10    }
11    return global_max;
12 }
```

2.6 MFlow

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

```

```

65         - flow[i]))) > 0)
66     {
67         flow[i] += f;
68         flow[i ^ 1] -= f;
69         a -= f;
70         flw += f;
71         if (a == 0)
72             break;
73     }
74 }
75 }
76 return flw;
77 }
78 ll MaxFlow(int s, int t)
79 {
80     this->s = s;
81     this->t = t;
82     ll flw = 0;
83     while (bfs())
84     {
85         for (int i = 1; i <= n; ++i)
86             cur[i] = 0;
87         flw += dfs(s, oo);
88     }
89     return flw;
90 }
91 // MF Net;
92 // Net.n = n;
93 // Net.Clear();
94 // a 到 b (注意從1開始!!!!)
95 // Net.Add(a, b, w, 0);
96 // Net.MaxFlow(s, d)
97 // s 到 d 的 MF

```

3 Geometry

3.1 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> &y) : p1(x), p2(y) {}
8     void pton()
9     { //轉成一般式
10         a = p1.y - p2.y;
11         b = p2.x - p1.x;
12         c = -a * p1.x - b * p1.y;
13     }
14     T ori(const point<T> &p) const
15     { //點和有向直線的關係 · >0左邊、=0在線上
16         <0右邊
17         return (p2 - p1).cross(p - p1);
18     }
19     T btw(const point<T> &p) const
20     { //點投影落在線段上<=0
21         return (p1 - p).dot(p2 - p);

```

```

21 }
22 bool point_on_segment(const point<T> &p)
23 {
24     const
25     return ori(p) == 0 && btw(p) <= 0;
26 }
27 T dis2(const point<T> &p, bool
28     is_segment = 0) const
29 { //點跟直線/線段的距離平方
30     point<T> v = p2 - p1, v1 = p - p1;
31     if (is_segment)
32     {
33         point<T> v2 = p - p2;
34         if (v.dot(v1) <= 0)
35             return v1.abs2();
36         if (v.dot(v2) >= 0)
37             return v2.abs2();
38     }
39     T tmp = v.cross(v1);
40     return tmp * tmp / v.abs2();
41 }
42 T seg_dis2(const line<T> &l) const
43 { //兩線段距離平方
44     return min({dis2(l.p1, 1), dis2(l.p2,
45         1), l.dis2(p1, 1), l.dis2(p2,
46         1)});
47 }
48 point<T> projection(const point<T> &p)
49 {
50     const
51     point<T> n = (p2 - p1).normal();
52     return p - n * (p - p1).dot(n) / n.
53         abs2();
54 }
55 point<T> mirror(const point<T> &p) const
56 {
57     //點對直線的鏡射 · 要先呼叫pton轉成一般式
58     point<T> R;
59     T d = a * a + b * b;
60     R.x = (b * b * p.x - a * a * p.x - 2 * a * b * p.y - 2 * a * c) / d;
61     R.y = (a * a * p.y - b * b * p.y - 2 * 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) == 0;
67 }
68 bool parallel(const line &l) const
69 {
70     return (p1 - p2).cross(l.p1 - l.p2) == 0;
71 }
72 bool cross_seg(const line &l) const
73 {
74     return (p2 - p1).cross(l.p1 - p1) * (p2 - p1).cross(l.p2 - p1) <= 0;
75     //直線是否交線段
76 }
77 int line_intersect(const line &l) const
78 {
79 }

```

```

77 { //直線相交情況 · -1無限多點、1交於一點、0不相交
78     return parallel(l) ? (ori(l.p1) == 0 ? -1 : 0) : 1;
79 }
80 int seg_intersect(const line &l) const
81 {
82     T c1 = ori(l.p1), c2 = ori(l.p2);
83     T c3 = l.ori(p1), c4 = l.ori(p2);
84     if (c1 == 0 && c2 == 0)
85     { //共線
86         bool b1 = btw(l.p1) >= 0, b2 = btw(l.p2) >= 0;
87         T a3 = l.btw(p1), a4 = l.btw(p2);
88         if (b1 && b2 && a3 == 0 && a4 >= 0)
89             return 2;
90         if (b1 && b2 && a3 >= 0 && a4 == 0)
91             return 3;
92         if (b1 && b2 && a3 >= 0 && a4 >= 0)
93             return 0;
94         return -1; //無限交點
95     }
96     else if (c1 * c2 <= 0 && c3 * c4 <= 0)
97         return 1;
98     return 0; //不相交
99 }
100 point<T> line_intersection(const line &l)
101 {
102     const
103     { /*直線交點*/
104         point<T> a = p2 - p1, b = l.p2 - l.p1, s = l.p1 - p1;
105         //if(a.cross(b)==0)return INF;
106         return p1 + a * (s.cross(b) / a.cross(b));
107     }
108     point<T> seg_intersection(const line &l)
109     { //線段交點
110         int res = seg_intersect(l);
111         if (res <= 0)
112             assert(0);
113         if (res == 2)
114             return p1;
115         if (res == 3)
116             return p2;
117         return line_intersection(l);
118     }
119 }

```

3.2 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(y) {}
7 point operator+(const point &b) const
8 {
9     return point(x + b.x, y + b.y);
10 }
11 point operator-(const point &b) const
12 {
13     return point(x - b.x, y - b.y);
14 }
15 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 { //向量長度的平方
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 };

```

3.3 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++)
19         {
20             T a = p[i].cross(p[j]);
21             cx += (p[i].x + p[j].x) * a;
22             cy += (p[i].y + p[j].y) * a;
23             w += a;
24         }
25         return point<T>(cx / 3 / w, cy / 3 /
26                          w);
27     }
28     char ahas(const point<T> &t) const
29     { //點是否在簡單多邊形內，是的話回傳1、
30       在邊上回傳-1、否則回傳0
31         bool c = 0;
32         for (int i = 0, j = p.size() - 1; i
33              < p.size(); j = i++)
34             if (line<T>(p[i], p[j]).
35                  point_on_segment(t))
36                 return -1;
37             else if ((p[i].y > t.y) != (p[j]
38                  .y > t.y) &&
39                     t.x < (p[j].x - p[i].x)
40                          * (t.y - p[i].y) /
41                          (p[j].y - p[i].y)
42                          + p[i].x)
43                 c = !c;
44         return c;
45     }
46     char point_in_convex(const point<T> &x)
47     const
48     {
49         int l = 1, r = (int)p.size() - 2;
50         while (l <= r)
51         { //點是否在凸多邊形內，是的話回傳1
52           在邊上回傳-1、否則回傳0
53             int mid = (l + r) / 2;
54             T a1 = (p[mid] - p[0]).cross(x -
55                  p[0]);
56             T a2 = (p[mid + 1] - p[0]).cross
57                  (x - p[0]);
58             if (a1 >= 0 && a2 <= 0)
59             {
60                 T res = (p[mid + 1] - p[mid]
61                          ).cross(x - p[mid]);
62                 return res > 0 ? 1 : (res >=
63                     0 ? -1 : 0);
64             }
65             else if (a1 < 0)
66                 r = mid - 1;
67             else
68                 l = mid + 1;
69         }
70         return 0;
71     }
72     vector<T> getA() const

```

```

73     { //凸包邊對x軸的夾角
74         vector<T> res; //一定是遞增的
75         for (size_t i = 0; i < p.size(); ++i
76              )
77             res.push_back((p[(i + 1) % p.
78                  size()] - p[i]).getA());
79         return res;
80     }
81     bool line_intersect(const vector<T> &A,
82                          const line<T> &l) const
83     { //O(logN)
84         int f1 = upper_bound(A.begin(), A.
85             end(), (l.p1 - l.p2).getA()) -
86             A.begin();
87         int f2 = upper_bound(A.begin(), A.
88             end(), (l.p2 - l.p1).getA()) -
89             A.begin();
90         return l.cross_seg(line<T>(p[f1], p[
91             f2]));
92     }
93     polygon cut(const line<T> &l) const
94     { //凸包對直線切割，得到直線l左側的凸包
95         polygon ans;
96         for (int n = p.size(), i = n - 1, j
97              = 0; j < n; i = j++)
98         {
99             if (l.ori(p[i]) >= 0)
100             {
101                 ans.p.push_back(p[i]);
102                 if (l.ori(p[j]) < 0)
103                     ans.p.push_back(l.
104                         line_intersection(
105                             line<T>(p[i], p[j])),
106                             );
107             }
108             else if (l.ori(p[j]) > 0)
109                 ans.p.push_back(l.
110                     line_intersection(line<T>
111                         >(p[i], p[j])));
112         }
113         return ans;
114     }
115     static bool graham_cmp(const point<T> &a
116                             , const point<T> &b)
117     { //凸包排序函數 //起點點不同
118         // return (a.x < b.x) || (a.x == b.x
119             && a.y < b.y); //最左下角開始
120         return (a.y < b.y) || (a.y == b.y &&
121             a.x < b.x); //Y最小開始
122     }
123     void graham(vector<point<T>> &s)
124     { //凸包 Convexhull 2D
125         sort(s.begin(), s.end(), graham_cmp)
126         ;
127         p.resize(s.size() + 1);
128         int m = 0;
129         // cross >= 0 順時針，cross <= 0 逆
130         // 時針旋轉
131         for (size_t i = 0; i < s.size(); ++i
132              )
133         {
134             while (m >= 2 && (p[m - 1] - p[m
135                 - 2]).cross(s[i] - p[m -
136                     2]) <= 0)

```

```

137                 --m;
138                 p[m++] = s[i];
139             }
140             for (int i = s.size() - 2, t = m +
141                 1; i >= 0; --i)
142             {
143                 while (m >= t && (p[m - 1] - p[m
144                     - 2]).cross(s[i] - p[m -
145                         2]) <= 0)
146                     --m;
147                 p[m++] = s[i];
148             }
149             if (s.size() > 1) //重複頭一次需扣
150                 --m;
151             p.resize(m);
152         }
153     }
154     T diam()
155     { //直徑
156         int n = p.size(), t = 1;
157         T ans = 0;
158         p.push_back(p[0]);
159         for (int i = 0; i < n; i++)
160         {
161             point<T> now = p[i + 1] - p[i];
162             while (now.cross(p[t + 1] - p[i]
163                 ]) > now.cross(p[t] - p[i]))
164                 t = (t + 1) % n;
165             ans = max(ans, (p[i] - p[t]).
166                 abs2());
167         }
168         return p.pop_back(), ans;
169     }
170     T min_cover_rectangle()
171     { //最小覆蓋矩形
172         int n = p.size(), t = 1, r = 1, l;
173         if (n < 3)
174             return 0; //也可以做最小周長矩形
175         T ans = 1e99;
176         p.push_back(p[0]);
177         for (int i = 0; i < n; i++)
178         {
179             point<T> now = p[i + 1] - p[i];
180             while (now.cross(p[t + 1] - p[i]
181                 ]) > now.cross(p[t] - p[i]))
182                 t = (t + 1) % n;
183             while (now.dot(p[r + 1] - p[i])
184                 > now.dot(p[r] - p[i]))
185                 r = (r + 1) % n;
186             if (!l)
187                 l = r;
188             while (now.dot(p[l + 1] - p[i])
189                 <= now.dot(p[l] - p[i]))
190                 l = (l + 1) % n;
191             T d = now.abs2();
192             T tmp = now.cross(p[t] - p[i]) *
193                 (now.dot(p[r] - p[i]) - now
194                     .dot(p[l] - p[i])) / d;
195             ans = min(ans, tmp);
196         }
197         return p.pop_back(), ans;
198     }
199     T dis2(polygon &pl)
200     { //凸包最近距離平方
201         vector<point<T>> &P = p, &Q = pl.p;

```

```

153 int n = P.size(), m = Q.size(), l = 202
    0, r = 0;
154 for (int i = 0; i < n; ++i)
155     if (P[i].y < P[l].y)
156         l = i;
157 for (int i = 0; i < m; ++i)
158     if (Q[i].y < Q[r].y)
159         r = i;
160 P.push_back(P[0]), Q.push_back(Q[0])
    ;
161 T ans = 1e99;
162 for (int i = 0; i < n; ++i)
163 {
164     while ((P[l] - P[l + 1]).cross(Q
        [r + 1] - Q[r]) < 0)
        r = (r + 1) % m;
        ans = min(ans, line<T>(P[l], P[l
            + 1]).seg_dis2(line<T>(Q[r
                ], Q[r + 1])));
        l = (l + 1) % n;
    }
165 return P.pop_back(), Q.pop_back(),
    ans;
166 }
167 static char sign(const point<T> &t)
168 {
169     return (t.y == 0 ? t.x : t.y) < 0;
170 }
171 static bool angle_cmp(const line<T> &A,
172     const line<T> &B)
173 {
174     point<T> a = A.p2 - A.p1, b = B.p2 -
        B.p1;
175     return sign(a) < sign(b) || (sign(a)
        == sign(b) && a.cross(b) > 0);
176 }
177 int halfplane_intersection(vector<line<T
    >> &s)
178 {
179     //半平面交
180     sort(s.begin(), s.end(), angle_cmp);
181     //線段左側為該線段半平面
182     int L, R, n = s.size();
183     vector<point<T>> px(n);
184     vector<line<T>> q(n);
185     q[L = R = 0] = s[0];
186     for (int i = 1; i < n; ++i)
187     {
188         while (L < R && s[i].ori(px[R -
            1]) <= 0)
189             --R;
190         while (L < R && s[i].ori(px[L])
            <= 0)
191             ++L;
192         q[++R] = s[i];
193         if (q[R].parallel(q[R - 1]))
194         {
195             --R;
196             if (q[R].ori(s[i].p1) > 0)
197                 q[R] = s[i];
198         }
199     }
200     if (L < R)
201         px[R - 1] = q[R - 1].
            line_intersection(q[R]);

```

3.4 Triangle

```

1 template <typename T>
2 struct triangle
3 {
4     point<T> a, b, c;
5     triangle() {}
6     triangle(const point<T> &a, const point<
    T> &b, const point<T> &c) : a(a), b(b
        ), c(c) {}
7     T area() const
8     {
9         T t = (b - a).cross(c - a) / 2;
10        return t > 0 ? t : -t;
11    }
12    point<T> barycenter() const
13    { //重心
14        return (a + b + c) / 3;
15    }
16    point<T> circumcenter() const
17    { //外心
18        static line<T> u, v;
19        u.p1 = (a + b) / 2;
20        u.p2 = point<T>(u.p1.x - a.y + b.y,
            u.p1.y + a.x - b.x);
21        v.p1 = (a + c) / 2;
22        v.p2 = point<T>(v.p1.x - a.y + c.y,
            v.p1.y + a.x - c.x);
23        return u.line_intersection(v);
24    }
25    point<T> incenter() const
26    { //內心
27        T A = sqrt((b - c).abs2()), B = sqrt
            ((a - c).abs2()), C = sqrt((a -
                b).abs2());
28        return point<T>(A * a.x + B * b.x +
            C * c.x, A * a.y + B * b.y + C *
                c.y) / (A + B + C);
29    }
30    point<T> perpcenter() const
31    { //垂心
32        return barycenter() * 3 -
            circumcenter() * 2;
33    }
34 }

```

4 Graph

4.1 Bellman-Ford

```

1 /*SPA - Bellman-Ford*/
2 #include<bits/stdc++.h>
3 #define inf 99999 //define by you maximum
    edges weight
4 using namespace std;
5 vector<vector<int>> edges;
6 vector<int> dist;
7 vector<int> ancestor;
8 void BellmanFord(int start, int node){
9     dist[start] = 0;
10    for(int it = 0; it < node-1; it++){
11        for(int i = 0; i < node; i++){
12            for(int j = 0; j < node; j++){
13                if(edges[i][j] != -1){
14                    if(dist[i] + edges[i][j]
                        < dist[j]){
15                        dist[j] = dist[i] +
                            edges[i][j];
16                        ancestor[j] = i;
17                    }
18                }
19            }
20        }
21    }
22    for(int i = 0; i < node; i++) //
        negative cycle detection
23    for(int j = 0; j < node; j++){
24        if(dist[i] + edges[i][j] < dist[
            j]){
25            cout<<"Negative cycle!"<<
                endl;
26            return;
27        }
28    }
29    int main(){
30        int node;
31        cin>>node;
32        edges.resize(node, vector<int>(node, inf))
33        ;
34        dist.resize(node, inf);
35        ancestor.resize(node, -1);
36        int a, b, d;
37        while(cin>>a>>b>>d){
38            /*input: source destination weight*/
39            if(a == -1 && b == -1 && d == -1)
40                break;
41            edges[a][b] = d;
42        }
43        int start;
44        cin>>start;
45        BellmanFord(start, node);
46        return 0;
47    }

```

4.2 BFS-queue

```

1 /*BFS - queue version*/
2 #include<bits/stdc++.h>
3 using namespace std;
4 void BFS(vector<int> &result, vector<pair<int
    ,int>> edges, int node, int start){
5     vector<int> pass(node, 0);
6     queue<int> q;
7     queue<int> p;
8     q.push(start);
9     int count = 1;
10    vector<pair<int, int>> newedges;
11    while(!q.empty()){
12        pass[q.front()] = 1;
13        for (int i = 0; i < edges.size(); i
            ++){
14            if(edges[i].first == q.front()
                && pass[edges[i].second] ==
                    0){
15                p.push(edges[i].second);
16                result[edges[i].second] =
                    count;
17            }
18            else if(edges[i].second == q.
                front() && pass[edges[i].
                    first] == 0){
19                p.push(edges[i].first);
20                result[edges[i].first] =
                    count;
21            }
22            else
23                newedges.push_back(edges[i])
                    ;
24        }
25        edges = newedges;
26        newedges.clear();
27        q.pop();
28        if(q.empty() == true){
29            q = p;
30            queue<int> tmp;
31            p = tmp;
32            count++;
33        }
34    }
35 }
36 int main(){
37     int node;
38     cin >> node;
39     vector<pair<int, int>> edges;
40     int a, b;
41     while(cin>>a>>b){
42         /*a = b = -1 means input edges ended
            */
43         if(a == -1 && b == -1)
44             break;
45         edges.push_back(pair<int, int>(a, b)
            );
46     }
47     vector<int> result(node, -1);
48     BFS(result, edges, node, 0);
49     return 0;
50 }

```

4.3 DFS-rec

```

1 /*DFS - Recursive version*/
2 #include<bits/stdc++.h>
3 using namespace std;
4 map<pair<int,int>,int> edges;
5 vector<int> pass;
6 vector<int> route;
7 void DFS(int start){
8     pass[start] = 1;
9     map<pair<int,int>,int>::iterator iter;
10    for(iter = edges.begin(); iter != edges.
11        end(); iter++){
12        if((*iter).first.first == start &&
13            (*iter).second == 0 && pass[(*)
14            iter).first.second] == 0){
15            route.push_back((*iter).first.
16                second);
17            DFS((*iter).first.second);
18        }
19        else if((*iter).first.second ==
20            start && (*iter).second == 0 &&
21            pass[(*)iter).first.first] == 0){
22            route.push_back((*iter).first.
23                first);
24            DFS((*iter).first.first);
25        }
26    }
27 }
28 int main(){
29     int node;
30     cin>>node;
31     pass.resize(node,0);
32     int a,b;
33     while(cin>>a>b){
34         if(a == -1 && b == -1)
35             break;
36         edges.insert(pair<pair<int,int>,int>
37             >(pair<int,int>(a,b),0));
38     }
39     int start;
40     cin>>start;
41     route.push_back(start);
42     DFS(start);
43     return 0;
44 }

```

4.4 Dijkstra

```

1 /*SPA - Dijkstra*/
2 #include<bits/stdc++.h>
3 #define inf INT_MAX
4 using namespace std;
5 vector<vector<int>> weight;
6 vector<int> ancestor;
7 vector<int> dist;
8 void dijkstra(int start){
9     priority_queue<pair<int,int>,vector<
10     pair<int,int>,greater<pair<int,
11     int>>> pq;
12     pq.push(make_pair(0,start));
13     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;
22             pq.push(make_pair(dist[i],i)
23                 );
24         }
25     }
26 }
27 int main(){
28     int node;
29     cin>>node;
30     int a,b,d;
31     weight.resize(node,vector<int>(node,-1))
32     ;
33     while(cin>>a>b>d){
34         /*input: source destination weight*/
35         if(a == -1 && b == -1 && d == -1)
36             break;
37         weight[a][b] = d;
38     }
39     ancestor.resize(node,-1);
40     dist.resize(node,inf);
41     int start;
42     cin>>start;
43     dist[start] = 0;
44     dijkstra(start);
45     return 0;
46 }

```

4.5 Floyd-warshall

```

1 /*SPA - Floyd-Warshall*/
2 #include<bits/stdc++.h>
3 #define inf 99999
4 using namespace std;
5 void floyd_warshall(vector<vector<int>>&
6     distance, vector<vector<int>>& ancestor,
7     int n){
8     for (int k = 0; k < n; k++){
9         for (int i = 0; i < n; i++){
10            for (int j = 0; j < n; j++){
11                if(distance[i][k] + distance
12                    [k][j] < distance[i][j])
13                    {
14                        distance[i][j] =
15                            distance[i][k] +
16                            distance[k][j];
17                        ancestor[i][j] =
18                            ancestor[k][j];
19                    }
20            }
21        }
22    }
23 }
24 int main(){

```

```

18     int n;
19     cin >> n;
20     int a, b, d;
21     vector<vector<int>> distance(n, vector<
22         int>(n,99999));
23     vector<vector<int>> ancestor(n, vector<
24         int>(n,-1));
25     while(cin>>a>b>d){
26         if(a == -1 && b == -1 && d == -1)
27             break;
28         distance[a][b] = d;
29         ancestor[a][b] = a;
30     }
31     for (int i = 0; i < n; i++)
32         distance[i][i] = 0;
33     floyd_warshall(distance, ancestor, n);
34     /*Negative cycle detection*/
35     for (int i = 0; i < n; i++){
36         if(distance[i][i] < 0){
37             cout << "Negative cycle!" <<
38                 endl;
39             break;
40         }
41     }
42     return 0;
43 }

```

4.6 union_find

```

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

```

```

31     for(int i = 0; i < edge; i++)
32     {
33         int a, b;
34         cin >> a >> b;
35         merge(a, b, union_set,rank);
36     }
37     /*build party*/
38     vector<vector<int>> party(node, vector<
39         int>(0));
40     for (int i = 0; i < node; i++)
41         party[find(i, union_set)].push_back(
42             i);

```

5 Mathematics

5.1 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];

```

5.2 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     }
50     cout << ans << endl;
51     return 0;
52 }

```

5.3 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         if (num[i] == '0' && num[i] == '9')
8         {
9             temp += (num[i] - 48) * base;
10            base = base * 16;
11        }
12        else if (num[i] == 'A' && num[i] == 'F')
13        {
14            temp += (num[i] - 55) * base;
15            base = base * 16;
16        }
17    }
18    return temp;
19 }
20 void DecToHex(int p_intValue) //10 to 16
21 {
22     char l_pCharRes = new (char);
23     sprintf(l_pCharRes, "%X", p_intValue);
24     int l_intResult = stoi(l_pCharRes);

```

```

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

```

5.4 Mod

```

1 int pow_mod(int a, int n, int m)
2 {
3     a = a % m;
4     if (n == 0) return 1;
5     int x = pow_mid(a, n / 2, m);
6     long long ans = (long long)x * x % m;
7     if (n % 2 == 1)
8         ans = ans * a % m;
9     return (int)ans;
10 }
11 // 加法: (a + b) % p = (a % p + b % p) % p;
12 // 減法: (a - b) % p = (a % p - b % p + p) % p;
13 // 乘法: (a * b) % p = (a % p * b % p) % p;
14 // 次方: (a ^ b) % p = ((a % p) ^ b) % p;
15 // 加法結合律: ((a + b) % p + c) % p = (a + (b + c)) % p;
16 // 乘法結合律: ((a * b) % p * c) % p = (a * (b * c)) % p;
17 // 加法交換律: (a + b) % p = (b + a) % p;
18 // 乘法交換律: (a * b) % p = (b * a) % p;
19 // 結合律: ((a + b) % p * c) % p = ((a * c) % p + (b * c) % p) % p;
20

```

5.5 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());

```

5.6 PI

```

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

```

5.7 Prime table

```

1 // 埃拉托斯特尼篩法
2 const int maxn = 1000000;
3 bitset<maxn> is_not_prime; // false 是質數
4 void sieve()
5 {
6     is_not_prime[0] = is_not_prime[1] = 1;

```

```

7     for (int i = 2; i * i < maxn; ++i)
8     {
9         if (is_not_prime[i] == 0)
10        {
11            for (int j = i * i; j < maxn; j += i)
12                is_not_prime[j] = 1;
13        }
14    }
15 }

```

5.8 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 }

```

5.9 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, string final index)
3 {
4     int c = 0;
5     for (int i = ri; i >= le; i--)
6     {
7         if (s[i] == '+')
8             c++;
9         if (s[i] == '-')
10            c--;
11        if (s[i] == '*' && c == 0)
12            return DFS(le, i - 1) + DFS(i + 1, ri);
13        if (s[i] == '/' && c == 0)
14            return DFS(le, i - 1) - DFS(i + 1, ri);
15    }
16    for (int i = ri; i >= le; i--)
17    {
18        if (s[i] == '+')
19            c++;
20        if (s[i] == '-')
21            c--;
22        if (s[i] == '*' && c == 0)
23            return DFS(le, i - 1) * DFS(i + 1, ri);
24        if (s[i] == '/' && c == 0)
25            return DFS(le, i - 1) / DFS(i + 1, ri);

```

```

26        if (s[i] == '%' && c == 0)
27            return DFS(le, i - 1) % DFS(i + 1, ri);
28    }
29    if ((s[le] == '(' && (s[ri] == ')')) ||
30        (s[le] == '[' && (s[ri] == ']')))
31        return DFS(le + 1, ri - 1); //去除刮號
32    if (s[le] == ' ' && s[ri] == ' ')
33        return DFS(le + 1, ri - 1); //去除左右兩邊空格
34    if (s[le] == '(' && s[ri] == ')')
35        return DFS(le + 1, ri - 1); //去除左邊空格
36    if (s[ri] == '[' && s[ri] == ']')
37        return DFS(le, ri - 1); //去除右邊空格
38    long long int num = 0;
39    for (int i = le; i <= ri; i++)
40        num = num * 10 + s[i] - '0';
41    return num;
42 }

```

5.10 數字乘法組合

```

1 void toans(vector<vector<int>> &ans, vector<int> com)
2 {
3     // sort(com.begin(), com.end());
4     ans.push_back(com);
5     // for (auto i : com)
6     //     cout << i << ' ';
7     // cout << endl;
8 }
9 void finds(int j, int old, int num, vector<int> com, vector<vector<int>> &ans)
10 {
11     for (int i = j; i <= sqrt(num); i++)
12     {
13         if (old == num)
14             com.clear();
15         if (num % i == 0)
16         {
17             vector<int> a;
18             a = com;
19             a.push_back(i);
20             finds(i, old, num / i, a, ans);
21             a.push_back(num / i);
22             toans(ans, a);
23         }
24     }
25 }
26 int main()
27 {
28     vector<vector<int>> ans;
29     vector<int> zero;
30     finds(2, num, num, zero, ans);
31     // num 為 input 數字
32     for (int i = 0; i < ans.size(); i++)
33     {
34         for (int j = 0; j < ans[i].size(); j++)

```

```

35     cout << ans[i][j] << " ";
36     cout << ans[i][ans[i].size() - 1] << endl;
37 }
38 }

```

5.11 數字加法組合

```

1 void printCombination(vector<int> const &out
2   , int m, vector<vector<int>> &ans)
3 {
4     for (int i : out)
5         if (i > m)
6             return;
7     ans.push_back(out);
8 }
9 void recur(int i, int n, int m, vector<int>
10    &out, vector<vector<int>> &ans)
11 {
12     if (n == 0)
13         printCombination(out, m, ans);
14     for (int j = i; j <= n; j++)
15     {
16         out.push_back(j);
17         recur(j, n - j, m, out, ans);
18         out.pop_back();
19     }
20 }
21 int main()
22 {
23     vector<vector<int>> ans;
24     vector<int> zero;
25     recur(1, num, num, zero, ans);
26     // num 為 input 數字
27     for (int i = 0; i < ans.size(); i++)
28     {
29         for (int j = 0; j < ans[i].size() - 1; j++)
30             cout << ans[i][j] << " ";
31         cout << ans[i][ans[i].size() - 1] << endl;
32     }
33 }

```

5.12 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 }

```

5.13 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        {
12            n /= p[i];
13        }
14    }
15    if (n != 1)
16    {
17        cout << n << ' ';
18    }
19    cout << '\n';
20 }

```

6 Other

6.1 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 - 1]);
30    }
31    int result = table[n - 1];
32    delete[] table;
33
34    return result;
35 }

```

6.2 數獨解法

```

1 int getSquareIndex(int row, int column, int n)
2 {
3     return row / n * n + column / n;
4 }
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
18     if (board[rowNum][colNum] != 0)
19         return backtracking(board, rows,
20           cols, boxes, index + 1, n);
21
22     for (int i = 1; i <= n2; i++)
23     {
24         if (!rows[rowNum][i] && !cols[colNum
25           ][i] && !boxes[getSquareIndex(
26             rowNum, colNum, n)][i])
27         {
28             rows[rowNum][i] = true;
29             cols[colNum][i] = true;
30             boxes[getSquareIndex(rowNum,
31               colNum, n)][i] = true;
32             board[rowNum][colNum] = i;
33             if (backtracking(board, rows,
34               cols, boxes, index + 1, n))
35                 return true;
36             board[rowNum][colNum] = 0;
37             rows[rowNum][i] = false;
38             cols[colNum][i] = false;
39             boxes[getSquareIndex(rowNum,
40               colNum, n)][i] = false;
41         }
42     }
43     return false;
44 }

```

```

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
46 for (int i = 0; i < n * n; ++i)
47 {
48     for (int j = 0; j < n * n; ++j)
49     {
50         int number;
51         cin >> number;
52         board[i][j] = number;
53         if (number == 0)
54             continue;
55         isRow[i][number] = true;
56         isColumn[j][number] = true;
57         isSquare[getSquareIndex(i, j, n)][
58             number] = true;
59     }
60 }
61 if (backtracking(board, isRow, isColumn,
62   isSquare, 0, n))
63     /*有解答*/
64 else
65     /*解答*/
66 }

```

7 String

7.1 sliding window

```

1 string minWindow(string s, string t) {
2     unordered_map<char, int> letterCnt;
3     for (int i = 0; i < t.length(); i++)
4         letterCnt[t[i]]++;
5     int minLength = INT_MAX, minStart = -1;
6     int left = 0, matchCnt = 0;
7     for (int i = 0; i < s.length(); i++)
8     {
9         if (--letterCnt[s[i]] >= 0)
10             matchCnt++;
11         while (matchCnt == t.length())
12         {
13             if (i - left + 1 < minLength)
14             {
15                 minLength = i - left + 1;
16                 minStart = left;
17             }
18             if (++letterCnt[s[left]] > 0)
19                 matchCnt--;
20             left++;
21         }
22     }
23 }

```

```

23     return minLength == INT_MAX ? "" : s.
24     substr(minStart, minLength);
}

```

7.2 split

```

1 vector<string> mysplit(const string& str,
2   const string& delim)
3 {
4     vector<string> res;
5     if (" " == str)
6         return res;
7
8     char *strs = new char[str.length() + 1];
9     strcpy(strs, str.c_str());
10
11    char *d = new char[delim.length() + 1];
12    strcpy(d, delim.c_str());
13
14    char *p = strtok(strs, d);
15    while (p)
16    {
17        string s = p;
18        res.push_back(s);
19        p = strtok(NULL, d);
20    }
21    return res;
}

```

8 data structure

8.1 Bigint

```

1 //台大
2 struct Bigint{
3     static const int LEN = 60;
4     static const int BIGMOD = 10000;
5     int s;
6     int vl, v[LEN];
7     // vector<int> v;
8     Bigint() : s(1) { vl = 0; }
9     Bigint(long long a) {
10         s = 1; vl = 0;
11         if (a < 0) { s = -1; a = -a; }
12         while (a) {
13             push_back(a % BIGMOD);
14             a /= BIGMOD;
15         }
16     }
17     Bigint(string str) {
18         s = 1; vl = 0;
19         int stPos = 0, num = 0;
20         if (!str.empty() && str[0] == '-') {
21             stPos = 1;
22             s = -1;
23         }
24         for (int i = str.length() - 1, q=1; i
25             >=stPos; i--) {

```

```

25         num += (str[i] - '0') * q;
26         if ((q *= 10) >= BIGMOD) {
27             push_back(num);
28             num = 0; q = 1;
29         }
30     }
31     if (num) push_back(num);
32     n();
33 }
34 int len() const {
35     return vl; //return SZ(v);
36 }
37 bool empty() const { return len() == 0; }
38 void push_back(int x) {
39     v[vl++] = x; //v.PB(x);
40 }
41 void pop_back() {
42     vl--; //v.pop_back();
43 }
44 int back() const {
45     return v[vl-1]; //return v.back();
46 }
47 void n() {
48     while (!empty() && !back()) pop_back
49         ();
50 }
51 void resize(int nl) {
52     vl = nl; //v.resize(nl);
53     fill(v, v+vl, 0); //fill(ALL(v), 0);
54 }
55 void print() const {
56     if (empty()) { putchar('0'); return;
57     }
58     if (s == -1) putchar('-');
59     printf("%d", back());
60     for (int i=len()-2; i>=0; i--)
61         printf("%.4d", v[i]);
62 }
63 friend std::ostream& operator << (std::
64     ostream& out, const Bigint &a) {
65     if (a.empty()) { out << "0"; return
66     out; }
67     if (a.s == -1) out << "-";
68     out << a.back();
69     for (int i=a.len()-2; i>=0; i--) {
70         char str[10];
71         snprintf(str, 5, "%.4d", a.v[i])
72         ;
73         out << str;
74     }
75     return out;
76 }
77 int cp3(const Bigint &b) const {
78     if (s != b.s) return s - b.s;
79     if (s == -1) return -(*this).cp3(-b
80     );
81     if (len() != b.len()) return len()-b
82     .len(); //int
83     for (int i=len()-1; i>=0; i--)
84         if (v[i] != b.v[i]) return v[i]-b
85         .v[i];
86     return 0;
87 }
88 bool operator<(const Bigint &b) const
89 { return cp3(b)<0; }

```

```

81 bool operator<=(const Bigint &b) const
82 { return cp3(b)<=0; }
83 bool operator==(const Bigint &b) const
84 { return cp3(b)==0; }
85 bool operator!=(const Bigint &b) const
86 { return cp3(b)!=0; }
87 bool operator>(const Bigint &b) const
88 { return cp3(b)>0; }
89 bool operator>=(const Bigint &b) const
90 { return cp3(b)>=0; }
91 Bigint operator - () const {
92     Bigint r = (*this);
93     r.s = -r.s;
94     return r;
95 }
96 Bigint operator + (const Bigint &b)
97 const {
98     if (s == -1) return -(*this)+(-b);
99     ;
100     if (b.s == -1) return (*this)-(-b);
101     Bigint r;
102     int nl = max(len(), b.len());
103     r.resize(nl + 1);
104     for (int i=0; i<nl; i++) {
105         if (i < len()) r.v[i] += v[i];
106         if (i < b.len()) r.v[i] += b.v[i]
107         ;
108         if (r.v[i] >= BIGMOD) {
109             r.v[i+1] += r.v[i] / BIGMOD;
110             r.v[i] %= BIGMOD;
111         }
112     }
113     r.n();
114     return r;
115 }
116 Bigint operator - (const Bigint &b)
117 const {
118     if (s == -1) return -(*this)-(-b);
119     ;
120     if (b.s == -1) return (*this)+(-b);
121     if ((*this) < b) return -b-(*this)
122     ;
123     Bigint r;
124     r.resize(len());
125     for (int i=0; i<len(); i++) {
126         r.v[i] += v[i];
127         if (i < b.len()) r.v[i] -= b.v[i]
128         ;
129         if (r.v[i] < 0) {
130             r.v[i] += BIGMOD;
131             r.v[i+1]--;
132         }
133     }
134     r.n();
135     return r;
136 }
137 Bigint operator * (const Bigint &b) {
138     Bigint r;
139     r.resize(len() + b.len() + 1);
140     r.s = s * b.s;
141     for (int i=0; i<len(); i++) {
142         for (int j=0; j<b.len(); j++) {
143             r.v[i+j] += v[i] * b.v[j];
144             if (r.v[i+j] >= BIGMOD) {
145                 r.v[i+j+1] += r.v[i+j] /
146                 BIGMOD;
147             }
148         }
149     }
150     r.n();
151     return r;
152 }

```

```

139         r.v[i+j] %= BIGMOD;
140     }
141 }
142 }
143 r.n();
144 return r;
145 }
146 Bigint operator / (const Bigint &b) {
147     Bigint r;
148     r.resize(max(1, len()-b.len()+1));
149     int oriS = s;
150     Bigint b2 = b; // b2 = abs(b)
151     s = b2.s = r.s = 1;
152     for (int i=r.len()-1; i>=0; i--) {
153         int d=0, u=BIGMOD-1;
154         while(d<u) {
155             int m = (d+u+1)>>1;
156             r.v[i] = m;
157             if((r*b2) > (*this)) u = m
158             -1;
159             else d = m;
160         }
161         r.v[i] = d;
162     }
163     s = oriS;
164     r.s = s * b.s;
165     r.n();
166     return r;
167 }
168 Bigint operator % (const Bigint &b) {
169     return (*this)-(*this)/b*b;
170 }
171 }

```

8.2 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);

```



```

24     s.push_back(x);
25 }
26 return *this;
27 }
28
29 BigInteger operator + (const BigInteger&
30 b) const{
31     BigInteger c;
32     c.s.clear();
33     for(int i = 0, g = 0;;i++){
34         if(g == 0 && i >= s.size() && i
35             >= b.s.size()) break;
36         int x = g;
37         if(i < s.size()) x+=s[i];
38         if(i < b.s.size()) x+=b.s[i];
39         c.s.push_back(x % BASE);
40         g = x / BASE;
41     }
42     return c;
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
57 istream& operator >> (istream &in,
58 BigInteger& x){
59     string s;
60     if(!(in >> s))
61         return in;
62     x = s;
63     return in;
64 }
65
66 struct Trie{
67     int c[5000005][10];
68     int val[5000005];
69     int sz;
70     int getIndex(char c){
71         return c - '0';
72     }
73     void init(){
74         memset(c[0], 0, sizeof(c[0]));
75         memset(val, -1, sizeof(val));
76         sz = 1;
77     }
78
79     void insert(BigInteger x, int v){
80         int u = 0;
81         int max_len_count = 0;
82         int firstNum = x.s.back();
83         char firstBuf[20];
84         sprintf(firstBuf, "%d", firstNum);

```

```

85         for(int j = 0; j < strlen(firstBuf);
86             j++){
87             int index = getIndex(firstBuf[j]
88                 );
89             if(!c[u][index]){
90                 memset(c[sz], 0, sizeof(c[
91                     sz]));
92                 val[sz] = v;
93                 c[u][index] = sz++;
94             }
95             u = c[u][index];
96             max_len_count++;
97         }
98         for(int i = x.s.size()-2; i >= 0;i
99             --){
100             char buf[20];
101             sprintf(buf, "%08d", x.s[i]);
102             for(int j = 0; j < strlen(buf)
103                 && max_len_count < 50;j++){
104                 int index = getIndex(buf[j])
105                     ;
106                 if(!c[u][index]){
107                     memset(c[sz], 0, sizeof
108                         (c[sz]));
109                     val[sz] = v;
110                     c[u][index] = sz++;
111                 }
112                 u = c[u][index];
113                 max_len_count++;
114             }
115             if(max_len_count >= 50){
116                 break;
117             }
118         }
119     }
120
121     int find(const char* s){
122         int u = 0;
123         int n = strlen(s);
124         for(int i = 0; i < n;i++){
125             int index = getIndex(s[i]);
126             if(!c[u][index]){
127                 return -1;
128             }
129             u = c[u][index];
130         }
131         return val[u];
132     }
133 }

```

8.3 分數

```

1 class Rational
2 {
3     friend istream &operator>>(istream &,
4         Rational &);
5     friend ostream &operator<<(ostream &,
6         const Rational &);
7 public:
8     Rational() //constructor one
9     {

```

```

10         m_numeitor = 0;
11         m_denominator = 1;
12     }
13     Rational(int a, int b) //constructor two
14     {
15         if (b < 0 || b == 0) //avoids negative
16             denominators. && prevents a 0
17             denominator
18         {
19             cout << "This Rational number can't be
20                 used.\n\n";
21             m_numeitor = 0;
22             m_denominator = 0;
23         }
24         else
25         {
26             cout << "This Rational number can be
27                 used.\n\n";
28             m_numeitor = a;
29             m_denominator = b;
30         }
31     }
32     Rational operator+(const Rational& a); //
33     加
34     Rational operator-(const Rational& a); //
35     減
36     Rational operator*(const Rational& a); //
37     乘
38     Rational operator/(const Rational& a); //
39     除
40     bool operator==(const Rational& a); //相
41     等
42     void reduce(); //化簡
43 private:
44     int m_numeitor;
45     int m_denominator;
46 };
47
48 istream &operator>>(istream &input, Rational
49 &test )
50 {
51     char temp;
52     input >> test.m_numeitor;
53     input >> temp;
54     input >> test.m_denominator;
55     Rational final(test.m_numeitor, test.
56         m_denominator); //final用來告訴使用者
57     這數字不符合!
58     if (test.m_denominator < 0 || test.
59         m_denominator == 0) //不符合(再輸入
60         一次)
61     {
62         while (test.m_denominator < 0 || test.
63             m_denominator == 0) //有可能輸入的
64             東西還是不符合,所以用迴圈
65     {
66         cout << "Enter another Rational number
67             (n/d): ";
68         input >> test.m_numeitor;
69         input >> temp;
70         input >> test.m_denominator;
71         Rational final(test.m_numeitor, test.
72             m_denominator); //final用來告訴使

```

```

73         }
74         return input;
75     }
76     else
77         return input;
78 }
79 ostream &operator<<(ostream &output, const
80 Rational &test )
81 {
82     output << test.m_numeitor;
83     if(test.m_numeitor == 0)
84         return output;
85     if (test.m_denominator == 1)
86         return output;
87     else
88     {
89         output << "/";
90         output << test.m_denominator;
91     }
92     return output;
93 }
94 Rational Rational::operator+(const Rational&
95 a)
96 {
97     Rational c;
98     c.m_denominator = this->m_denominator * a.
99         m_denominator; //通分(同乘)
100     c.m_numeitor = (this->m_numeitor * a.
101         m_denominator) + (a.m_numeitor * this
102         ->m_denominator);
103     c.reduce();
104     return c;
105 }
106 Rational Rational::operator-(const Rational&
107 a)
108 {
109     Rational c;
110     c.m_denominator = this->m_denominator * a.
111         m_denominator;
112     c.m_numeitor = (this->m_numeitor * a.
113         m_denominator) - (a.m_numeitor * this
114         ->m_denominator);
115     c.reduce();
116     return c;
117 }
118 Rational Rational::operator*(const Rational&
119 a)
120 {
121     Rational c;
122     c.m_denominator = this->m_denominator * a.
123         m_denominator;
124     c.m_numeitor = (this->m_numeitor * a.
125         m_denominator) - (a.m_numeitor * this
126         ->m_denominator);
127     c.reduce();
128     return c;
129 }
130 Rational Rational::operator/(const Rational&
131 a)
132 {
133     Rational c;
134     c.m_denominator = this->m_denominator * a.
135         m_denominator;
136     c.m_numeitor = this->m_numeitor * a.
137         m_numeitor;
138     c.reduce();
139     return c;
140 }
141 Rational Rational::operator/(const Rational&
142 a)
143 {
144     Rational c;
145     c.m_denominator = this->m_denominator * a.
146         m_numeitor;
147     c.m_numeitor = this->m_numeitor * a.
148         m_denominator;

```

用者這數字不符合!

```
102 | c.reduce();
103 | return c;
104 | }
105 | bool Rational::operator==(const Rational& a)
106 | {
107 |     if (m_numeitor == a.m_numeitor)
108 |     {
109 |         if (m_denominator == a.m_denominator)
110 |             return true;
111 |         else
112 |             return false;
113 |     }
114 |     else
115 |         return false;
116 | }
117 | void Rational::reduce()
118 | {
119 |     int i;
120 |     int max;
121 |     if(m_numeitor > m_denominator)
122 |         max = m_numeitor;
123 |     else
124 |         max = m_denominator;
125 |     for (i = 2; i <= max; i++)
126 |     {
127 |         if (m_denominator % i == 0 && m_numeitor
128 |             % i == 0)
129 |         {
130 |             m_denominator /= i;
131 |             m_numeitor /= i;
132 |             i = 1;
133 |             max = m_denominator;
134 |             continue;
135 |         }
136 |     }
```

TO DO WRITING

NOT THINKING

Contents

1 Basic	1	2.2 LCS	1	4.4 Dijkstra	5	5.11 數字加法組合	7
1.1 data range	1	2.3 LIC	1	4.5 Floyd-warshall	5	5.12 羅馬數字	7
1.2 IO_fast	1	2.4 LPS	1	4.6 union_find	5	5.13 質因數分解	7
2 DP	1	2.5 Max_subarray	1	5 Mathematics	5	6 Other	7
2.1 KMP	1	2.6 MFlow	1	5.1 Combination	5	6.1 Weighted Job Scheduling . .	7
		3 Geometry	2	5.2 Extended Euclidean	5	6.2 數獨解法	7
		3.1 Line	2	5.3 Hex to Dec	6	7 String	7
		3.2 Point	2	5.4 Mod	6	7.1 sliding window	7
		3.3 Polygon	3	5.5 Permutation	6	7.2 split	8
		3.4 Triangle	4	5.6 PI	6	8 data structure	8
		4 Graph	4	5.7 Prime table	6	8.1 Bigint	8
		4.1 Bellman-Ford	4	5.8 二分逼近法	6	8.2 Trie	8
		4.2 BFS-queue	4	5.9 四則運算	6	8.3 分數	9
		4.3 DFS-rec	5	5.10 數字乘法組合	6		