

1 Basic

1.1 Codeblock setting

```
1 Settings -> Editor -> Keyboard shortcuts ->
2   Plugins -> Source code formatter (AStyle
3   )
4 Settings -> Source Formatter -> Padding
5 Delete empty lines within a function or
6   method
7 Insert space padding around operators
8 Insert space padding around parentheses on
9   outside
10 Remove extra space padding around
11   parentheses
```

1.2 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)
```

1.3 IO_fast

```
1 void io()
2 {
3   ios::sync_with_stdio(false);
4   cin.tie(nullptr);
5   cout.tie(nullptr);
6 }
```

1.4 常忘記

```
1 round(double f); // 四捨五入
2 ceil(double f); // 無條件捨去
3 floor(double f); // 無條件進入
4
5 /*queue*/
6 queue<datatype> q;
7 front(); /*取出最前面的值(沒有移除掉喔!)*
8 back(); /*取出最後面的值(沒有移除掉!)*
9 pop(); /*移除最前面的值*/
10 push(); /*新增值到最後面*/
11 empty(); /*回傳bool,檢查是不是空的queue*/
12 size(); /*queue 的大小*/
13
14 /*stack*/
15 stack<datatype> s;
```

```
16 top(); /*取出最上面的值(沒有移除掉喔!)*
17 pop(); /*移除最上面的值*/
18 push(); /*新增值到最上面*/
19 empty(); /*回傳bool,檢查是不是空的stack*/
20 size(); /*stack 的大小*/
```

2 DP

2.1 3 維 DP 思路

```
1 解題思路: dp[i][j][k]
2 i 跟 j 代表 range i ~ j 的 value
3 k 在我的理解裡是視题目的要求而定的
4 像是 Remove Boxes 當中 k 代表的是在 i 之前還
5   有多少個連續的箱子
6 所以每次區間消去的值就是(k+1) * (k+1)
7 換言之,我認為可以理解成 k 的意義就是題目今
8   天所關注的重點,就是老師說的題目所規定的
9   運算
```

2.2 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[
9       i]);
10    for (int k = 1; num > 0; k *= 2)
11    {
12      if (k > num)
13        k = num;
14      num -= k;
15      for (int j = w; j >= weight[i] *
16        k; --j)
17        c[j] = max(c[j], c[j -
18          weight[i] * k] + cost[i]
19          * k);
20    }
21  }
22  cout << "Max Prince" << c[w];
23 }
```

2.3 Knapsack sample

```
1 int Knapsack(vector<int> weight, vector<int>
2   value, int bag_Weight)
3 {
4   // vector<int> weight = {1, 3, 4};
5   // 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;
9   j++)
10   dp[0][j] = value[0];
11 // weight 數組的大小就是物品個數
12 for (int i = 1; i < weight.size(); i++)
13 { // 遍歷物品
14   for (int j = 0; j <= bagWeight; j++)
15   { // 遍歷背包容量
16     if (j < weight[i]) dp[i][j] = dp
17       [i - 1][j];
18     else dp[i][j] = max(dp[i - 1][j
19       ], dp[i - 1][j - weight[i]]
20       + value[i]);
21   }
22 }
23 cout << dp[weight.size() - 1][bagWeight]
24   << endl;
25 }
```

2.4 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]
11        ] + cost[i]);
12   }
13   cout << "最高的價值為" << c[w];
14 }
```

2.5 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   {
8     table[j] = 0;
9     for (int i = 0; i < n; i++)
10     {
11       int current = 0;
12       for (int j = 0; j < m; j++)
13       {
14         if (arr1[i] == arr2[j])
15           if (current + 1 > table[j])
16             table[j] = current + 1;
17       }
18       if (arr1[i] > arr2[j])
19         if (table[j] > current)
20           current = table[j];
21     }
22   }
```

```
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 }
```

2.6 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 -
13           1] + 1;
14       else
15         LCS[i][j] = max(LCS[i - 1][j
16           ], LCS[i][j - 1]);
17     }
18   }
19   cout << LCS[N][M] << '\n';
20 //列印 LCS
21 int n = N, m = M;
22 vector<string> k;
23 while (n && m)
24 {
25   if (LCS[n][m] != max(LCS[n - 1][m],
26     LCS[n][m - 1]))
27   {
28     k.push_back(Ans[n - 1]);
29     n--;
30     m--;
31   }
32   else if (LCS[n][m] == LCS[n - 1][m])
33     n--;
34   else if (LCS[n][m] == LCS[n][m - 1])
35     m--;
36 }
37 reverse(k.begin(), k.end());
38 for (auto i : k)
39   cout << i << " ";
40 cout << endl;
41 return LCS[N][M];
42 }
```

2.7 LIS

```

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

```

2.8 LPS

```

1 void LPS(string s)
2 {
3     int maxlen = 0, l, r;
4     int n = s.size();
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)) //even length
22            x++;
23        if (2 * x > maxlen)
24        {
25            maxlen = 2 * x;
26            l = i - x + 1;
27            r = i + x;
28        }
29    }
30    cout << maxlen << '\n'; // 最後長度
31    cout << l + 1 << ' ' << r + 1 << '\n';
32    //頭到尾

```

2.9 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;

```

2.10 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
12        if (c[limit]) cout << "YES\n";
13        else cout << "NO\n";
14    }
15    // 湊得某個價位的湊法總共幾種
16    void change(vector<int> price, int limit)
17    {
18        vector<int> c(limit + 1, 0);
19        c[0] = true;
20        for (int i = 0; i < price.size(); ++i)
21            for (int j = price[i]; j <= limit;
22                ++j)
23                c[j] += c[j - price[i]];
24        cout << c[limit] << '\n';
25    }
26    // 湊得某個價位的最少錢幣用量
27    void change(vector<int> price, int limit)
28    {
29        vector<int> c(limit + 1, 0);
30        c[0] = true;
31        for (int i = 0; i < price.size(); ++i)
32            for (int j = price[i]; j <= limit;
33                ++j)
34                c[j] = min(c[j], c[j - price[i]]
35                    + 1);
36        cout << c[limit] << '\n';
37    }
38    // 湊得某個價位的錢幣用量，有幾種可能性
39    void change(vector<int> price, int limit)
40    {
41        vector<int> c(limit + 1, 0);
42        c[0] = true;
43        for (int i = 0; i < price.size(); ++i)
44            for (int j = price[i]; j <= limit;
45                ++j)
46                c[j] |= c[j - price[i]] << 1; //
47                錢幣數量加一，每一種可能性都
48                加一。
49    }
50    for (int i = 1; i <= 63; ++i)
51        if (c[i] & (1 << i))
52            cout << "用 " << i << "個錢幣可湊
53            得價位 " << i;

```

3 Flow & matching

3.1 Edmonds_karp

```

1 /*Flow - Edmonds-karp*/
2 /*Based on UVa820*/
3 #define inf 1000000
4 int getMaxFlow(vector<vector<int>> &capacity
5     , int s, int t, int n){

```

```

5     int ans = 0;
6     vector<vector<int>> residual(n+1, vector<
7         int>(n+1, 0)); //residual network
8     while(true){
9         vector<int> bottleneck(n+1, 0);
10        bottleneck[s] = inf;
11        queue<int> q;
12        q.push(s);
13        vector<int> pre(n+1, 0);
14        while(!q.empty() && bottleneck[t] != 0){
15            int cur = q.front();
16            q.pop();
17            for(int i = 1; i <= n; i++){
18                if(bottleneck[i] == 0 && capacity[
19                    cur][i] > residual[cur][i]){
20                    q.push(i);
21                    pre[i] = cur;
22                    bottleneck[i] = min(bottleneck[cur
23                        ], capacity[cur][i] - residual
24                        [cur][i]);
25                }
26            }
27        }
28        if(bottleneck[t] == 0) break;
29        for(int cur = t; cur != s; cur = pre[cur
30            ]){
31            residual[pre[cur]][cur] +=
32            bottleneck[t];
33            residual[cur][pre[cur]] -=
34            bottleneck[t];
35        }
36        ans += bottleneck[t];
37    }
38    return ans;
39 }
40 int main(){
41     int testcase = 1;
42     int n;
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;

```

3.2 Maximum_matching

```

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

```

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 = 1000000000000LL;
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]]
50            + 1;
51          que[++tail] = v[i];
52        }
53      }
54      ++head;
55    }
56    return vi[t] == tim;
57  }
58  ll dfs(int x, ll a)
59  {
60    if (x == t || a == 0)
61      return a;
62    ll flw = 0, f;
63    int &i = cur[x];
64    for (i = first[x]; i != -1; i = next
65      [i])
66    {
67      if (dis[x] + 1 == dis[v[i]] && (
68        f = dfs(v[i], min(a, cap[i]
69          - flow[i]))) > 0)
70      {
71        flow[i] += f;
72        flow[i ^ 1] -= f;
73        a -= f;
74        flw += f;
75        if (a == 0)
76          break;

```

```

77      }
78    }
79    return flw;
80  }
81  ll MaxFlow(int s, int t)
82  {
83    this->s = s;
84    this->t = t;
85    ll flw = 0;
86    while (bfs())
87    {
88      for (int i = 1; i <= n; ++i)
89        cur[i] = 0;
90      flw += dfs(s, oo);
91    }
92    return flw;
93  }
94  // MF Net;
95  // Net.n = n;
96  // Net.Clear();
97  // a 到 b (注意從1開始!!!!)
98  // Net.Add(a, b, w, 0);
99  // Net.MaxFlow(s, d)
100 // 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
32    point<T> v = p2 - p1, v1 = p - p1;
33    if (is_segment)
34    {
35      point<T> v2 = p - p2;
36      if (v.dot(v1) <= 0)
37        return v1.abs2();
38      if (v.dot(v2) >= 0)
39        return v2.abs2();
40    }
41    T tmp = v.cross(v1);
42    return tmp * tmp / v.abs2();
43  }
44  T seg_dis2(const line<T> &l) const
45  { //兩線段距離平方
46    return min({dis2(l.p1, 1), dis2(l.p2
47      , 1), l.dis2(p1, 1), l.dis2(p2,
48      1)});
49  }
50  point<T> projection(const point<T> &p)
51  const
52  { //點對直線的投影
53    point<T> n = (p2 - p1).normal();
54    return p - n * (p - p1).dot(n) / n.
55      abs2();
56  }
57  point<T> mirror(const point<T> &p) const
58  {
59    //點對直線的鏡射·要先呼叫pton轉成一般式
60    point<T> R;
61    T d = a * a + b * b;
62    R.x = (b * b * p.x - a * a * p.x - 2
63      * a * b * p.y - 2 * a * c) / d;
64    R.y = (a * a * p.y - b * b * p.y - 2
65      * a * b * p.x - 2 * b * c) / d;
66    return R;
67  }
68  bool equal(const line &l) const
69  { //直線相等
70    return ori(l.p1) == 0 && ori(l.p2)
71      == 0;
72  }
73  bool parallel(const line &l) const
74  {
75    return (p1 - p2).cross(l.p1 - l.p2)
76      == 0;
77  }
78  bool cross_seg(const line &l) const
79  {
80    return (p2 - p1).cross(l.p1 - p1) *
81      (p2 - p1).cross(l.p2 - p1) <= 0;
82    //直線是否交線段
83  }
84  int line_intersect(const line &l) const
85  { //直線相交情況·-1無限多點、1交於一
86    點、0不相交
87    return parallel(l) ? (ori(l.p1) == 0
88      ? -1 : 0) : 1;
89  }
90  int seg_intersect(const line &l) const
91  {
92    T c1 = ori(l.p1), c2 = ori(l.p2);
93    T c3 = l.ori(p1), c4 = l.ori(p2);

```

```

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

```

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(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 {
50     //對x軸的弧度
51     //超過180度會變負
52     T A = atan2(y, x);
53     if (A <= -PI / 2)
54         A += PI * 2;
55     return A;
56 }

```

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;

```

```

63 for (int i = p.size() - 1, j = 0; j
64     < (int)p.size(); i = j++)
65 {
66     T a = p[i].cross(p[j]);
67     cx += (p[i].x + p[j].x) * a;
68     cy += (p[i].y + p[j].y) * a;
69     w += a;
70 }
71 return point<T>(cx / 3 / w, cy / 3 /
72     w);
73 }
74 char ahas(const point<T> &t) const
75 { //點是否在簡單多邊形內，是的話回傳1、
76     在邊上回傳-1、否則回傳0
77     bool c = 0;
78     for (int i = 0, j = p.size() - 1; i
79         < p.size(); j = i++)
80         if (line<T>(p[i], p[j]).
81             point_on_segment(t))
82             return -1;
83         else if ((p[i].y > t.y) != (p[j]
84             .y > t.y) &&
85             t.x < (p[j].x - p[i].x)
86                 * (t.y - p[i].y) /
87                 (p[j].y - p[i].y)
88                 + p[i].x)
89             c = !c;
90     return c;
91 }
92 char point_in_convex(const point<T> &x)
93 const
94 {
95     int l = 1, r = (int)p.size() - 2;
96     while (l <= r)
97     { //點是否在凸多邊形內，是的話回傳1
98         、在邊上回傳-1、否則回傳0
99         int mid = (l + r) / 2;
100         T a1 = (p[mid] - p[0]).cross(x -
101             p[0]);
102         T a2 = (p[mid + 1] - p[0]).cross
103             (x - p[0]);
104         if (a1 >= 0 && a2 <= 0)
105         {
106             T res = (p[mid + 1] - p[mid]
107                 ).cross(x - p[mid]);
108             return res > 0 ? 1 : (res >=
109                 0 ? -1 : 0);
110         }
111         else if (a1 < 0)
112             r = mid - 1;
113         else
114             l = mid + 1;
115     }
116     return 0;
117 }
118 vector<T> getA() const
119 { //凸包邊對x軸的夾角
120     vector<T> res; //一定是遞增的
121     for (size_t i = 0; i < p.size(); ++i
122         )
123         res.push_back((p[(i + 1) % p.
124             size()] - p[i]).getA());
125     return res;
126 }

```

```

bool line_intersect(const vector<T> &A,
const line<T> &l) const
{ //O(logN)
int f1 = upper_bound(A.begin(), A.
end(), (1.p1 - 1.p2).getA()) - A.
begin();
int f2 = upper_bound(A.begin(), A.
end(), (1.p2 - 1.p1).getA()) - A.
begin();
return l.cross_seg(line<T>(p[f1], p[
f2]));
}
polygon cut(const line<T> &l) const
{ //凸包對直線切割，得到直線l左側的凸包
polygon ans;
for (int n = p.size(), i = n - 1, j
= 0; j < n; i = j++)
{
if (1.ori(p[i]) >= 0)
{
ans.p.push_back(p[i]);
if (1.ori(p[j]) < 0)
ans.p.push_back(1.
line_intersection(
line<T>(p[i], p[j]))
);
}
else if (1.ori(p[j]) > 0)
ans.p.push_back(1.
line_intersection(line<T>
>(p[i], p[j])));
}
return ans;
}
static bool graham_cmp(const point<T> &a
, const point<T> &b)
{ //凸包排序函數 // 起始點不同
// return (a.x < b.x) || (a.x == b.x
&& a.y < b.y); //最左下角開始
return (a.y < b.y) || (a.y == b.y &&
a.x < b.x); //Y最小開始
}
void graham(vector<point<T>> &s)
{ //凸包 Convexhull 2D
sort(s.begin(), s.end(), graham_cmp)
;
p.resize(s.size() + 1);
int m = 0;
// cross >= 0 順時針，cross <= 0 逆
時針旋轉
for (size_t i = 0; i < s.size(); ++i
)
{
while (m >= 2 && (p[m - 1] - p[m
- 2]).cross(s[i] - p[m -
2]) <= 0)
--m;
p[m++] = s[i];
}
for (int i = s.size() - 2, t = m +
1; i >= 0; --i)
{
while (m >= t && (p[m - 1] - p[m
- 2]).cross(s[i] - p[m -

```

```

2]) <= 0)
--m;
p[m++] = s[i];
}
if (s.size() > 1) // 重複頭一次需扣掉
--m;
p.resize(m);
}
T diam()
{ //直徑
int n = p.size(), t = 1;
T ans = 0;
p.push_back(p[0]);
for (int i = 0; i < n; i++)
{
point<T> now = p[i + 1] - p[i];
while (now.cross(p[t + 1] - p[i]) > now.cross(p[t] - p[i]))
t = (t + 1) % n;
ans = max(ans, (p[i] - p[t]).abs2());
}
return p.pop_back(), ans;
}
T min_cover_rectangle()
{ //最小覆蓋矩形
int n = p.size(), t = 1, r = 1, l;
if (n < 3)
return 0; //也可以做最小周長矩形
T ans = 1e99;
p.push_back(p[0]);
for (int i = 0; i < n; i++)
{
point<T> now = p[i + 1] - p[i];
while (now.cross(p[t + 1] - p[i]) > now.cross(p[t] - p[i]))
t = (t + 1) % n;
while (now.dot(p[r + 1] - p[i]) > now.dot(p[r] - p[i]))
r = (r + 1) % n;
if (l)
l = r;
while (now.dot(p[l + 1] - p[i]) < now.dot(p[l] - p[i]))
l = (l + 1) % n;
T d = now.abs2();
T tmp = now.cross(p[t] - p[i]) * (now.dot(p[r] - p[i]) - now.dot(p[l] - p[i])) / d;
ans = min(ans, tmp);
}
return p.pop_back(), ans;
}
T dis2(polygon &p1)
{ //凸包最近距離平方
vector<point<T>> &P = p, &Q = p1.p;
int n = P.size(), m = Q.size(), l = 0, r = 0;
for (int i = 0; i < n; ++i)
if (P[i].y < P[l].y)
l = i;
for (int i = 0; i < m; ++i)
if (Q[i].y < Q[r].y)
r = i;

```

```

P.push_back(P[0]), Q.push_back(Q[0])
;
T ans = 1e99;
for (int i = 0; i < n; ++i)
{
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;
}
return P.pop_back(), Q.pop_back(), ans;
}
static char sign(const point<T> &t)
{
return (t.y == 0 ? t.x : t.y) < 0;
}
static bool angle_cmp(const line<T> &A, const line<T> &B)
{
point<T> a = A.p2 - A.p1, b = B.p2 - B.p1;
return sign(a) < sign(b) || (sign(a) == sign(b) && a.cross(b) > 0);
}
int halfplane_intersection(vector<line<T>> &s)
{ //半平面交
sort(s.begin(), s.end(), angle_cmp);
//線段左側為該線段半平面
int L, R, n = s.size();
vector<point<T>> px(n);
vector<line<T>> q(n);
q[L = R = 0] = s[0];
for (int i = 1; i < n; ++i)
{
while (L < R && s[i].ori(px[R - 1]) <= 0)
--R;
while (L < R && s[i].ori(px[L]) <= 0)
++L;
q[++R] = s[i];
if (q[R].parallel(q[R - 1]))
{
--R;
if (q[R].ori(s[i].p1) > 0)
q[R] = s[i];
}
if (L < R)
px[R - 1] = q[R - 1].line_intersection(q[R]);
}
while (L < R && q[L].ori(px[R - 1]) <= 0)
--R;
p.clear();
if (R - L <= 1)
return 0;
px[R] = q[R].line_intersection(q[L]);
for (int i = L; i <= R; ++i)

```

```

p.push_back(px[i]);
return R - L + 1;
}
}
};

4.4 Triangle

template <typename T>
struct triangle
{
point<T> a, b, c;
triangle() {}
triangle(const point<T> &a, const point<T> &b, const point<T> &c) : a(a), b(b), c(c) {}
T area() const
{
T t = (b - a).cross(c - a) / 2;
return t > 0 ? t : -t;
}
point<T> barycenter() const
{ //重心
return (a + b + c) / 3;
}
point<T> circumcenter() const
{ //外心
static line<T> u, v;
u.p1 = (a + b) / 2;
u.p2 = point<T>(u.p1.x - a.y + b.y, u.p1.y + a.x - b.x);
v.p1 = (a + c) / 2;
v.p2 = point<T>(v.p1.x - a.y + c.y, v.p1.y + a.x - c.x);
return u.line_intersection(v);
}
point<T> incenter() const
{ //內心
T A = sqrt((b - c).abs2()), B = sqrt((a - c).abs2()), C = sqrt((a - b).abs2());
return point<T>(A * a.x + B * b.x + C * c.x, A * a.y + B * b.y + C * c.y) / (A + B + C);
}
point<T> perpencenter() const
{ //垂心
return barycenter() * 3 - circumcenter() * 2;
}
};

```

```

#define inf 99999 //define by you maximum
edges weight
vector<vector<int>> edges;
vector<int> dist;
vector<int> ancestor;
void BellmanFord(int start, int node)
{
dist[start] = 0;
for (int it = 0; it < node - 1; it++)
for (int i = 0; i < node; i++)
for (int j = 0; j < node; j++)
if (edges[i][j] != -1)
if (dist[i] + edges[i][j] < dist[j])
{
dist[j] = dist[i] + edges[i][j];
ancestor[j] = i;
}
}
}
for (int i = 0; i < node; i++) //negative cycle detection
for (int j = 0; j < node; j++)
if (dist[i] + edges[i][j] < dist[j])
{
cout << "Negative cycle!" << endl;
return;
}
}
int main()
{
int node;
cin >> node;
edges.resize(node, vector<int>(node, inf));
dist.resize(node, inf);
ancestor.resize(node, -1);
int a, b, d;
while (cin >> a >> b >> d)
{
/*input: source destination weight*/
if (a == -1 && b == -1 && d == -1)
break;
edges[a][b] = d;
}
int start;
cin >> start;
BellmanFord(start, node);
return 0;
}

```

5.2 BFS-queue

```

/*BFS - queue version*/
void BFS(vector<int> &result, vector<pair<int, int>> edges, int node, int start)
{
vector<int> pass(node, 0);
queue<int> q;
queue<int> p;
q.push(start);

```

5 Graph

5.1 Bellman-Ford

```

/*SPA - Bellman-Ford*/

```



```

8 int count = 1;
9 vector<pair<int, int>> newedges;
10 while (!q.empty())
11 {
12     pass[q.front()] = 1;
13     for (int i = 0; i < edges.size(); i++)
14     {
15         if (edges[i].first == q.front()
16             && pass[edges[i].second] == 0)
17         {
18             p.push(edges[i].second);
19             result[edges[i].second] = count;
20         }
21         else if (edges[i].second == q.front()
22                 && pass[edges[i].first] == 0)
23         {
24             p.push(edges[i].first);
25             result[edges[i].first] = count;
26         }
27         else
28             newedges.push_back(edges[i]);
29     }
30     edges = newedges;
31     newedges.clear();
32     q.pop();
33     if (q.empty() == true)
34     {
35         q = p;
36         queue<int> tmp;
37         p = tmp;
38         count++;
39     }
40 }
41 int main()
42 {
43     int node;
44     cin >> node;
45     vector<pair<int, int>> edges;
46     int a, b;
47     while (cin >> a >> b)
48     {
49         /*a = b = -1 means input edges ended*/
50         if (a == -1 && b == -1)
51             break;
52         edges.push_back(pair<int, int>(a, b));
53     }
54     vector<int> result(node, -1);
55     BFS(result, edges, node, 0);
56     return 0;
57 }

```

5.3 DFS-rec

```

1 /*DFS - Recursive version*/
2 map<pair<int,int>,int> edges;
3 vector<int> pass;
4 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.end(); iter++){
9         if((*iter).first.first == start &&
10            (*iter).second == 0 && pass[(*iter).first.second] == 0){
11             route.push_back((*iter).first.second);
12             DFS((*iter).first.second);
13         }
14         else if((*iter).first.second == start && (*iter).second == 0 &&
15            pass[(*iter).first.first] == 0){
16             route.push_back((*iter).first.first);
17             DFS((*iter).first.first);
18         }
19     }
20 }
21 int main(){
22     int node;
23     cin>>node;
24     pass.resize(node,0);
25     int a,b;
26     while(cin>>a>>b){
27         if(a == -1 && b == -1)
28             break;
29         edges.insert(pair<pair<int,int>,int>(pair<int,int>(a,b),0));
30     }
31     int start;
32     cin>>start;
33     route.push_back(start);
34     DFS(start);
35     return 0;
36 }

```

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<pair<int,int>>,greater<pair<int,int>>> pq;
8     pq.push(make_pair(0,start));
9     while(!pq.empty()){
10         int cur = pq.top().second;
11         pq.pop();
12         for(int i = 0; i < weight[cur].size(); i++){
13             if(dist[i] > dist[cur] + weight[cur][i] && weight[cur][i] != -1){

```

```

14         dist[i] = dist[cur] + weight[cur][i];
15         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     while(cin>>a>>b>>d){
27         /*input: source destination weight*/
28         if(a == -1 && b == -1 && d == -1)
29             break;
30         weight[a][b] = d;
31     }
32     ancestor.resize(node,-1);
33     dist.resize(node,inf);
34     int start;
35     cin>>start;
36     dist[start] = 0;
37     dijkstra(start);
38     return 0;
39 }

```

5.5 Floyd-warshall

```

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

```

```

23         break;
24         distance[a][b] = d;
25         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!" << endl;
34             break;
35         }
36     }
37     return 0;
38 }

```

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], union_set);
14     return union_set[x];
15 }
16 void merge(int a,int b,vector<int>& union_set){
17     union_set[find(a, union_set)] = find(b, union_set);
18     if(pa != pb)
19         union_set[pa] = pb;
20 }
21 void kruskal(priority_queue<edges> pq,int n)
22 {
23     vector<int> union_set(n, 0);
24     for (int i = 0; i < n; i++)
25         union_set[i] = i;
26     int edge = 0;
27     int cost = 0; //evaluate cost of mst
28     while(!pq.empty() && edge < n - 1){
29         edges cur = pq.top();
30         int from = find(cur.from, union_set);
31         int to = find(cur.to, union_set);
32         if(from != to){
33             merge(from, to, union_set);
34             edge += 1;
35             cost += cur.weight;
36         }
37         pq.pop();
38     }
39     if(edge < n-1)

```

```

39     cout << "No mst" << endl;
40     else
41         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][i];
38                     pq.push(tmp);
39                 }
40             }
41         }
42     }
43 }

```

```

37     }
38     }
39     pass[cur.to] = true;
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++)

```

```

31     union_set[i] = i;
32     int edge;
33     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;

```

```

19     cout << xy.first << " * " << a << " + "
20         << xy.second << " * " << b << endl;
21     return 0;
22 }
23 // ax + by = gcd(a,b) * r
24 /*find |x|+|y| -> min*/
25 int main()
26 {
27     long long r, p, q; //px+qy = r*/
28     int cases;
29     cin >> cases;
30     while (cases-->0)
31     {
32         cin >> r >> p >> q;
33         pair<long long, long long> xy =
34             extgcd(q, p); // (x0, y0)
35         long long ans = 0, tmp = 0;
36         double k, k1;
37         long long s, s1;
38         k = 1 - (double)(r * xy.first) / p;
39         s = round(k);
40         ans = llabs(r * xy.first + s * p) +
41             llabs(r * xy.second - s * q);
42         k1 = -(double)(r * xy.first) / p;
43         s1 = round(k1);
44         /*cout << k << endl << k1 << endl;
45             cout << s << endl << s1 << endl;
46             */
47         tmp = llabs(r * xy.first + s1 * p) +
48             llabs(r * xy.second - s1 * q);
49         ans = min(ans, tmp);
50     }
51     cout << ans << endl;
52     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          if (num[i] == '0' && num[i] == '9')
8          {
9              temp += (num[i] - '0') * base;
10             base = base * 16;
11         }
12         else if (num[i] == 'A' && num[i] == 'F')
13         {
14             temp += (num[i] - 'A' + 10) * 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 }

```

6.6 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());

```

6.7 PI

```

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

```

6.8 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 }

```

6.9 primeBOOL

```

1 // n < 4759123141   chk = [2, 7, 61]
2 // n < 1122004669633   chk = [2, 13, 23, 1662803]
3 // n < 2^64   chk = [2, 325, 9375, 28178, 450775, 9780504, 1795265022]
4 vector<long long> chk = {};
5 long long fmul(long long a, long long n, long long mod)
6 {
7     long long ret = 0;
8     for (; n >= 1; n /= 2)
9     {
10         if (n & 1)
11             (ret += a) %= mod;
12         (a += a) %= mod;

```

```

13     }
14     return ret;
15 }
16
17 long long fpow(long long a, long long n, long long mod)
18 {
19     long long ret = 1LL;
20     for (; n >= 1; n /= 2)
21     {
22         if (n & 1)
23             ret = fmul(ret, a, mod);
24         a = fmul(a, a, mod);
25     }
26     return ret;
27 }
28
29 bool check(long long a, long long u, long 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
47 bool is_prime(long long n)
48 {
49     if (n < 2)
50         return false;
51     if (n % 2 == 0)
52         return n == 2;
53     long long u = n - 1;
54     int t = 0;
55     for (; u & 1; u /= 2, ++t)
56     {
57         if (!check(u, u, n, t))
58             return false;
59     }
60     return true;
61 }
62
63 // if (is_prime(int num)) // true == prime
64 // 反之亦然

```

6.10 Round(小數)

```

1 double myround(double number, unsigned int bits)
2 {
3     LL integerPart = number;
4     number -= integerPart;
5     for (unsigned int i = 0; i < bits; ++i)

```

```

6         number *= 10;
7         number = (LL)(number + 0.5);
8         for (unsigned int i = 0; i < bits; ++i)
9             number /= 10;
10        return integerPart + number;
11    }
12    //printf("%.1f\n", round(3.4515239, 1));

```

6.11 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 }

```

6.12 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     return DFS(le + 1, ri - 1); //去除剩
31     號
32 if (s[le] == ' ' && s[ri] == ' ')
33     return DFS(le + 1, ri - 1); //去除左
34     右兩邊空格
35 if (s[le] == ' ')
36     return DFS(le + 1, ri); //去除左邊空
37     格
38 if (s[ri] == ' ')
39     return DFS(le, ri - 1); //去除右邊空
40     格
41 }

```

6.13 數字乘法組合

```

1 void dfs(int j, int old, int num, vector<int>
  > com, vector<vector<int>> &ans)
2 {
3     for (int i = j; i <= sqrt(num); i++)
4     {
5         if (old == num)
6             com.clear();
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    vector<vector<int>> ans;
18    vector<int> zero;
19    dfs(2, num, num, zero, ans);
20    /*num 為 input 數字*/
21    for (int i = 0; i < ans.size(); i++)
22    {
23        for (int j = 0; j < ans[i].size() - 1; j
24            ++
25            cout << ans[i][j] << " ";
26        cout << ans[i][ans[i].size() - 1] <<
27            endl;
28    }

```

6.14 數字加法組合

```

1 void recur(int i, int n, int m, vector<int>
  &out, vector<vector<int>> &ans)
2 {
3     if (n == 0)

```

```

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

```

6.15 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 }

```

6.16 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 binary search 三類變化

```

1 // 查找和目標值完全相等的數
2 int find(vector<int> &nums, int target)
3 {
4     int left = 0, right = nums.size();
5     while (left < right)
6     {
7         int mid = left + (right - left) / 2;
8         if (nums[mid] == target)
9             return mid;
10        else if (nums[mid] < target)
11            left = mid + 1;
12        else
13            right = mid;
14    }
15    return -1;
16 }
17 // 找第一個不小於目標值的數 == 找最後一個小
18 // 於目標值的數
19 /*(lower_bound)*/
20 int find(vector<int> &nums, int target)
21 {
22     int left = 0, right = nums.size();
23     while (left < right)
24     {
25         int mid = left + (right - left) / 2;
26         if (nums[mid] < target)
27             left = mid + 1;
28        else
29            right = mid;
30    }
31    return right;
32 }
33 // 找第一個大於目標值的數 == 找最後一個不大
34 // 於目標值的數
35 /*(upper_bound)*/
36 int find(vector<int> &nums, int target)
37 {
38     int left = 0, right = nums.size();
39     while (left < right)
40     {
41         int mid = left + (right - left) / 2;
42         if (nums[mid] <= target)
43             left = mid + 1;
44        else
45            right = mid;

```

```

44     }
45     return right;
46 }

```

7.2 heap sort

```

1 void MaxHeapify(vector<int> &array, int root
  , int length)
2 {
3     int left = 2 * root,
4         right = 2 * root + 1,
5         largest;
6     if (left <= length && array[left] >
7         array[root])
8         largest = left;
9     else
10        largest = root;
11    if (right <= length && array[right] >
12        array[largest])
13        largest = right;
14    if (largest != root)
15    {
16        swap(array[largest], array[root]);
17        MaxHeapify(array, largest, length);
18    }
19 }
20 void HeapSort(vector<int> &array)
21 {
22     array.insert(array.begin(), 0);
23     for (int i = (int)array.size() / 2; i >=
24         1; i--)
25        MaxHeapify(array, i, (int)array.size()
26            - 1);
27    int size = (int)array.size() - 1;
28    for (int i = (int)array.size() - 1; i >=
29        2; i--)
30    {
31        swap(array[1], array[i]);
32        size--;
33        MaxHeapify(array, 1, size);
34    }
35    array.erase(array.begin());
36 }

```

7.3 Merge sort

```

1 void Merge(vector<int> &arr, int front, int
  mid, int end)
2 {
3     vector<int> LeftSub(arr.begin() + front,
4         arr.begin() + mid + 1);
5     vector<int> RightSub(arr.begin() + mid +
6         1, arr.begin() + end + 1);
7     LeftSub.insert(LeftSub.end(), INT_MAX);
8     RightSub.insert(RightSub.end(), INT_MAX)
9     ;
10    int idxLeft = 0, idxRight = 0;
11    for (int i = front; i <= end; i++)
12    {

```

```

11     if (LeftSub[idxLeft] <= RightSub[
12         idxRight])
13     {
14         arr[i] = LeftSub[idxLeft];
15         idxLeft++;
16     }
17     else
18     {
19         arr[i] = RightSub[idxRight];
20         idxRight++;
21     }
22 }
23 void MergeSort(vector<int> &arr, int front,
24     int end)
25 {
26     // 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 }

```

7.4 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 void QuickSort(vector<int> &arr, int front,
18     int end)
19 {
20     // front = 0 , end = arr.size() - 1
21     if (front < end)
22     {
23         int pivot = Partition(arr, front,
24             end);
25         QuickSort(arr, front, pivot - 1);
26         QuickSort(arr, pivot + 1, end);
27     }
28 }

```

7.5 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 }

```

7.6 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

```

```

19     if (!rows[rowNum][i] && !cols[colNum
20         ][i] && !boxes[getSquareIndex(
21             rowNum, colNum, n)][i])
22     {
23         rows[rowNum][i] = true;
24         cols[colNum][i] = true;
25         boxes[getSquareIndex(rowNum,
26             colNum, n)][i] = true;
27         board[rowNum][colNum] = i;
28         if (backtracking(board, rows,
29             cols, boxes, index + 1, n))
30             return true;
31         board[rowNum][colNum] = 0;
32         rows[rowNum][i] = false;
33         cols[colNum][i] = false;
34         boxes[getSquareIndex(rowNum,
35             colNum, n)][i] = false;
36     }
37     return false;
38 }
39 /*用法 main*/
40 int n = sqrt(數獨邊長大小) /*e.g. 9*9 n=3*/
41 vector<vector<int>> board(n * n + 1, vector<
42     int>(n * n + 1, 0));
43 vector<vector<bool>> isRow(n * n + 1, vector<
44     bool>(n * n + 1, false));
45 vector<vector<bool>> isColumn(n * n + 1,
46     vector<bool>(n * n + 1, false));
47 vector<vector<bool>> isSquare(n * n + 1,
48     vector<bool>(n * n + 1, false));
49 for (int i = 0; i < n * n; ++i)
50 {
51     for (int j = 0; j < n * n; ++j)
52     {
53         int number;
54         cin >> number;
55         board[i][j] = number;
56         if (number == 0)
57             continue;
58         isRow[i][number] = true;
59         isColumn[j][number] = true;
60         isSquare[getSquareIndex(i, j, n)][
61             number] = true;
62     }
63 }
64 if (backtracking(board, isRow, isColumn,
65     isSquare, 0, n))
66     /*有解答*/
67 else
68     /*解答*/
69 }

```

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             q = 0;
34         }
35     }
36 }
37 // string s = "abcdabcdebcd";
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 }

```

```

19     return dp[m][n];
20 }

```

9 data structure

9.1 Bigint

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 }

```

```

1 //台大
2 struct Bigint
3 {
4     static const int LEN = 60; //
5     static const int BIGMOD = 10000; //10為
6     static const int maxLEN;
7     static const int 正常位數;
8     int s;
9     int v1, v[LEN];
10    // vector<int> v;
11    Bigint() : s(1) { v1 = 0; }
12    Bigint(long long a)
13    {
14        s = 1;
15        v1 = 0;
16        if (a < 0)
17        {
18            s = -1;
19            a = -a;
20        }
21        while (a)
22        {
23            push_back(a % BIGMOD);
24            a /= BIGMOD;
25        }
26    }
27    Bigint(string str)
28    {
29        s = 1;
30        v1 = 0;
31        int stPos = 0, num = 0;
32        if (!str.empty() && str[0] == '-')
33        {
34            stPos = 1;
35            s = -1;
36        }
37        for (int i = str.length() - 1, q =
38            1; i >= stPos; i--)
39        {
40            num += (str[i] - '0') * q;
41            if ((q *= 10) >= BIGMOD)
42            {
43                push_back(num);
44                num = 0;
45                q = 1;
46            }
47            if (num)
48                push_back(num);
49        }
50        n();
51    }
52    int len() const
53    {
54        return v1; //return SZ(v);
55    }
56    bool empty() const { return len() == 0; }
57    void push_back(int x)
58    {

```

```

59     v[v1++] = x; //v.PB(x);
60 }
61 void pop_back()
62 {
63     v1--; //v.pop_back();
64 }
65 int back() const
66 {
67     return v[v1 - 1]; //return v.back();
68 }
69 void n()
70 {
71     while (!empty() && !back())
72         pop_back();
73 }
74 void resize(int n1)
75 {
76     v1 = n1; //v.resize(n1);
77     fill(v, v + v1, 0); //fill(ALL(v),
78         0);
79 }
80 void print() const
81 {
82     if (empty())
83     {
84         putchar('0');
85         return;
86     }
87     if (s == -1)
88         putchar('-');
89     printf("%d", back());
90     for (int i = len() - 2; i >= 0; i--)
91         printf("%.4d", v[i]);
92 }
93 friend std::ostream &operator<<(std::
94     ostream &out, const Bigint &a)
95 {
96     if (a.empty())
97     {
98         out << "0";
99         return out;
100    }
101    if (a.s == -1)
102        out << "-";
103    out << a.back();
104    for (int i = a.len() - 2; i >= 0; i
105        --)
106    {
107        char str[10];
108        snprintf(str, 5, "%.4d", a.v[i])
109        ;
110        out << str;
111    }
112    return out;
113 }
114 int cp3(const Bigint &b) const
115 {
116     if (s != b.s)
117         return s - b.s;
118     if (s == -1)
119         return -(*this).cp3(-b);
120     if (len() != b.len())
121         return len() - b.len(); //int
122     for (int i = len() - 1; i >= 0; i--)
123         if (v[i] != b.v[i])
124             return v[i] - b.v[i];
125 }
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 }
143 bool operator>(const Bigint &b) const
144 {
145     return cp3(b) > 0;
146 }
147 bool operator>=(const Bigint &b) const
148 {
149     return cp3(b) >= 0;
150 }
151 Bigint operator-() const
152 {
153     Bigint r = (*this);
154     r.s = -r.s;
155     return r;
156 }
157 Bigint operator+(const Bigint &b) const
158 {
159     if (s == -1)
160         return -(*this) + (-b);
161     if (b.s == -1)
162         return (*this) - (-b);
163     Bigint r;
164     int n1 = max(len(), b.len());
165     r.resize(n1 + 1);
166     for (int i = 0; i < n1; i++)
167     {
168         if (i < len())
169             r.v[i] += v[i];
170         if (i < b.len())
171             r.v[i] += b.v[i];
172         if (r.v[i] >= BIGMOD)
173         {
174             r.v[i + 1] += r.v[i] /
175                 BIGMOD;
176             r.v[i] %= BIGMOD;
177         }
178     }
179     r.n();
180     return r;
181 }
182 Bigint operator-(const Bigint &b) const
183 {

```

```

184     return 0;
185 }
186 bool operator<(const Bigint &b) const
187 {
188     return cp3(b) < 0;
189 }
190 bool operator<=(const Bigint &b) const
191 {
192     return cp3(b) <= 0;
193 }
194 bool operator==(const Bigint &b) const
195 {
196     return cp3(b) == 0;
197 }
198 bool operator!=(const Bigint &b) const
199 {
200     return cp3(b) != 0;
201 }
202 bool operator>(const Bigint &b) const
203 {
204     return cp3(b) > 0;
205 }
206 bool operator>=(const Bigint &b) const
207 {
208     return cp3(b) >= 0;
209 }
210 Bigint operator-() const
211 {
212     Bigint r = (*this);
213     r.s = -r.s;
214     return r;
215 }
216 Bigint operator+(const Bigint &b) const
217 {
218     if (s == -1)
219         return -(*this) + (-b);
220     if (b.s == -1)
221         return (*this) - (-b);
222     Bigint r;
223     int n1 = max(len(), b.len());
224     r.resize(n1 + 1);
225     for (int i = 0; i < n1; i++)
226     {
227         if (i < len())
228             r.v[i] += v[i];
229         if (i < b.len())
230             r.v[i] += b.v[i];
231         if (r.v[i] >= BIGMOD)
232         {
233             r.v[i + 1] += r.v[i] /
234                 BIGMOD;
235             r.v[i] %= BIGMOD;
236         }
237     }
238     r.n();
239     return r;
240 }
241 Bigint operator-(const Bigint &b) const
242 {

```

```

183 r.resize(len());
184 for (int i = 0; i < len(); i++)
185 {
186     r.v[i] += v[i];
187     if (i < b.len())
188         r.v[i] -= b.v[i];
189     if (r.v[i] < 0)
190     {
191         r.v[i] += BIGMOD;
192         r.v[i + 1]--;
193     }
194 }
195 r.n();
196 return r;
197 }
198 BigInt operator*(const BigInt &b)
199 {
200     BigInt r;
201     r.resize(len() + b.len() + 1);
202     r.s = s * b.s;
203     for (int i = 0; i < len(); i++)
204     {
205         for (int j = 0; j < b.len(); j
206             ++){
207             r.v[i + j] += v[i] * b.v[j];
208             if (r.v[i + j] >= BIGMOD)
209             {
210                 r.v[i + j + 1] += r.v[i
211                     + j] / BIGMOD;
212                 r.v[i + j] %= BIGMOD;
213             }
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         int d = 0, u = BIGMOD - 1;
230         while (d < u)
231         {
232             int m = (d + u + 1) >> 1;
233             r.v[i] = m;
234             if ((r * b2) > (*this))
235                 u = m - 1;
236             else
237                 d = m;
238         }
239         r.v[i] = d;
240     }
241     s = oriS;
242     r.s = s * b.s;
243     r.n();
244     return r;
245 }
246 BigInt operator%(const BigInt &b)

```

9.2 matirx

```

245 {
246     return (*this) - (*this) / b * b;
247 }
248 };
249
250 template <typename T>
251 struct Matrix
252 {
253     using rt = std::vector<T>;
254     using mt = std::vector<rt>;
255     using matrix = Matrix<T>;
256     int r, c; // [r][c]
257     mt m;
258     Matrix(int r, int c) : r(r), c(c), m(r,
259         rt(c)) {}
260     Matrix(mt a) { m = a, r = a.size(), c =
261         a[0].size(); }
262     rt &operator[](int i) { return m[i]; }
263     matrix operator+(const matrix &a)
264     {
265         matrix rev(r, c);
266         for (int i = 0; i < r; ++i)
267             for (int j = 0; j < c; ++j)
268                 rev[i][j] = m[i][j] + a.m[i
269                     ][j];
270         return rev;
271     }
272     matrix operator-(const matrix &a)
273     {
274         matrix rev(r, c);
275         for (int i = 0; i < r; ++i)
276             for (int j = 0; j < c; ++j)
277                 rev[i][j] = m[i][j] - a.m[i
278                     ][j];
279         return rev;
280     }
281     matrix operator*(const matrix &a)
282     {
283         matrix rev(r, a.c);
284         matrix tmp(a.c, a.r);
285         for (int i = 0; i < a.r; ++i)
286             for (int j = 0; j < a.c; ++j)
287                 tmp[j][i] = a.m[i][j];
288         for (int i = 0; i < r; ++i)
289             for (int j = 0; j < a.c; ++j)
290                 for (int k = 0; k < c; ++k)
291                     rev.m[i][j] += m[i][k] *
292                         tmp[j][k];
293         return rev;
294     }
295     bool inverse() //逆矩陣判斷
296     {
297         Matrix t(r, r + c);
298         for (int y = 0; y < r; y++)
299         {
300             t.m[y][c + y] = 1;
301             for (int x = 0; x < c; ++x)
302                 t.m[y][x] = m[y][x];
303         }
304         if (!t.gas())
305             return false;
306     }
307 }

```

```

308 for (int y = 0; y < r; y++)
309     for (int x = 0; x < c; ++x)
310         m[y][x] = t.m[y][c + x] / t.
311             m[y][y];
312     return true;
313 }
314 T gas() //行列式
315 {
316     vector<T> lazy(r, 1);
317     bool sign = false;
318     for (int i = 0; i < r; ++i)
319     {
320         if (m[i][i] == 0)
321         {
322             int j = i + 1;
323             while (j < r && !m[j][i])
324                 j++;
325             if (j == r)
326                 continue;
327             m[i].swap(m[j]);
328             sign = !sign;
329         }
330         for (int j = 0; j < r; ++j)
331         {
332             if (i == j)
333                 continue;
334             lazy[j] = lazy[j] * m[i][i];
335             T mx = m[j][i];
336             for (int k = 0; k < c; ++k)
337                 m[j][k] = m[j][k] * m[i
338                     ][i] - m[i][k] * mx;
339         }
340     }
341     T det = sign ? -1 : 1;
342     for (int i = 0; i < r; ++i)
343     {
344         det = det * m[i][i];
345         det = det / lazy[i];
346         for (auto &j : m[i])
347             j /= lazy[i];
348     }
349     return det;
350 }
351 };
352
353 // biginter字典數
354 struct BigInteger{
355     static const int BASE = 100000000;
356     static const int WIDTH = 8;
357     vector<int> s;
358     BigInteger(long long num = 0){
359         *this = num;
360     }
361     BigInteger operator = (long long num){
362         s.clear();
363         do{
364             s.push_back(num % BASE);
365             num /= BASE;
366         }while(num > 0);
367         return *this;
368     }
369 }

```

9.3 Trie

```

369 // biginter字典數
370 struct BigInteger{
371     static const int BASE = 100000000;
372     static const int WIDTH = 8;
373     vector<int> s;
374     BigInteger(long long num = 0){
375         *this = num;
376     }
377     BigInteger operator = (long long num){
378         s.clear();
379         do{
380             s.push_back(num % BASE);
381             num /= BASE;
382         }while(num > 0);
383         return *this;
384     }
385 }
386
387 BigInteger operator = (const string& str
388     ){
389     s.clear();
390     int x, len = (str.length() - 1) /
391         WIDTH + 1;
392     for(int i = 0; i < len; i++){
393         int end = str.length() - i*WIDTH
394             ;
395         int start = max(0, end-WIDTH);
396         sscanf(str.substr(start, end-
397             start).c_str(), "%d", &x);
398         s.push_back(x);
399     }
400     return *this;
401 }
402
403 BigInteger operator + (const BigInteger&
404     b) const{
405     BigInteger c;
406     c.s.clear();
407     for(int i = 0, g = 0; i < s.size() && i
408         < b.s.size()){
409         if(g == 0 && i >= s.size() && i
410             >= b.s.size()) break;
411         int x = g;
412         if(i < s.size()) x+=s[i];
413         if(i < b.s.size()) x+=b.s[i];
414         c.s.push_back(x % BASE);
415         g = x / BASE;
416     }
417     return c;
418 }
419
420 ostream& operator << (ostream &out, const
421     BigInteger& x){
422     out << x.s.back();
423     for(int i = x.s.size()-2; i >= 0; i--){
424         char buf[20];
425         sprintf(buf, "%08d", x.s[i]);
426         for(int j = 0; j < strlen(buf); j++){
427             out << buf[j];
428         }
429     }
430     return out;
431 }
432
433 istream& operator >> (istream &in,
434     BigInteger& x){
435     string s;
436     if(!(in >> s))
437         return in;
438     x = s;
439     return in;
440 }
441
442 struct Trie{
443     int c[5000005][10];
444     int val[5000005];
445     int sz;
446     int getIndex(char c){
447         return c - '0';
448     }
449     void init(){
450         memset(c[0], 0, sizeof(c[0]));
451         memset(val, -1, sizeof(val));
452     }
453 }

```

```

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

```

```

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
25    {
26        return fraction(n * b.d - b.n * d, d * b
27            .d);
28    }
29    fraction operator*(const fraction &b)
30        const
31    {
32        return fraction(n * b.n, d * b.d);
33    }
34    fraction operator/(const fraction &b)
35        const
36    {
37        return fraction(n * b.d, d * b.n);
38    }
39    void print()
40    {
41        cout << n;
42        if (d != 1)
43            cout << "/" << d;
44    }
45 };

```

9.4 分數

```

1 typedef long long ll;
2 struct fraction
3 {

```


Contents

6.16	質因數分解	9
7	Other	9
7.1	binary search 三類變化	9
7.2	heap sort	9
7.3	Merge sort	9
7.4	Quick	10
7.5	Weighted Job Scheduling . .	10
7.6	數獨解法	10
8	String	10
8.1	KMP	10
8.2	Min Edit Distance	10
8.3	Sliding window	11
8.4	Split	11
9	data structure	11
9.1	Bigint	11
9.2	matirx	12
9.3	Trie	12
9.4	分數	13