Contents

1 Graph
1.1 Dijkstra_algorithm 1
1.2 eular_circut 1
1.3 Floyd_Warshell 2
1.4 KM 2
1.5 Kruskal_algorithm 3
1.6 LCA 3
1.7 MaxFlow 4
1.8 maximum_dis_onTree 5
1.9 minimumCyclewithDirectGraph_byFloyd 5
1.10 SPFA 5
1.11 SPFA_withNagtiveCycle 6

2 DataStructure
2.1 BIT 6
2.2 DisjoinSet 6
2.3 lower_bound 7
2.4 Priority_Queue 7
2.5 Segment_Tree 7

3 DP
3.1 Bag 8
3.2 DP_coin_change 8
3.3 LCS 9
3.4 LIS 9
3.5 minCoinChange 9

4 Math
4.1 baby-giant-step 9
4.2 china_mod 10
4.3 Convex Hull-Andrew's $_{M}onotone_{C}hain$ 10
4.4 eularphi_request 10
4.5 eularphi_table 11
4.6 Fibonacci_log(n)_requset 11
4.7 josephus 12
4.8 mod_inv 12
4.9 prime_table 12
4.10 prime_table_linear 13
4.11 topology_sort 13
4.12 line_intersection 13

5 Misc
5.1 big_integer 13
5.2 romanToInt 13
5.3 SubFactorial 14
5.4 template 14
5.5 pymodel 14
5.6 remain 15
5.7 avltree 15

1 Graph

1.1 Dijkstra_algorithm

```cpp
#define MP make_pair
#define PII pair<int, int>
#define maxn

int dis[maxn];
vector<PII> e[maxn];

void dijk(int cur)
{
  int d;
  priority_queue<PII,vector<PII>,greater<PII>> q;
  q.push( MP(0, cur) );

  while (!q.empty())
  {
    tie(d, cur) = q.top();
    q.pop();
    if (dis[cur] != 1e9)
        continue;

    dis[cur] = d;

    for (auto i: e[cur])
        if (dis[i.first] == 1e9)
        {
            q.push( MP(d+i.second, i.first) );
        }
    }
}

void init(void)
{
    fill(dis, dis+maxn, 1e9);

    for(int i = 0; i < maxn; i++)
    {
        e[i].clear();
    }
}
```

1.2 eular_circut

```cpp
#define ll long long
#define PB push_back
#define EB emplace_back
#define PII pair<int, int>
#define MP make_pair
#define all(x) x.begin(), x.end()
#define maxn 50000+5

//structure
struct Eular{
    vector<PII> adj[maxn];
    vector<bool> edges;
    vector<PII> path;
    int chk[maxn];
    int n;

    void init(int _n){
        n = _n;

        for(int i = 0; i <= n; i++)
            adj[i].clear();
        edges.clear();
        path.clear();
        memset(chk, 0, sizeof(chk));
    }

    void dfs(int v)
    {
        for(auto i : adj[v])
        {
            if(edges[i.first] == true)
            {
                edges[i.first] = false;
                dfs(i.second);
                path.EB(MP(i.second, v));
            }
        }
    }

    void add_Edge(int from, int to){
        edges.PB(true);

        // for bi-directed graph
        adj[from].PB(MP(edges.size()-1, to));
        adj[to].PB(MP(edges.size()-1, from));
        chk[from]++;
        chk[to]++;

        // for directed graph
        // adj[from].PB(MP(edges.size()-1, to));
        // check[from]++;
    }

    bool eular_path(){
        int st = -1;
        for(int i = 1; i <= n; i++){
            if(chk[i]%2 == 1){
                st = i;
                break;
```

```
            }
        }

        if(st == -1){
            return false;
        }

        dfs(st);

        return true;
    }

    void print_path(void){
        for(auto i : path){
            printf("%d %d\n", i.first, i.second);
        }
    }
};
```

1.3 Floyd_Warshell

```
for (int k = 1; k <= n; k++)
    for (int i = 1; i <= n; i++)
      for (int j = 1; j <= n; j++)
          if (dis[i][j] > dis[i][k] + dis[k][j])
            dis[i][j] = dis[i][k] + dis[k][j];
```

1.4 KM

```
template <typename T>
struct KM
{
    int n;
    // Left: y_i match x_Left[i]
    int Left[maxn];
    // w: weight array
    T w[maxn][maxn], Lx[maxn], Ly[maxn];
    bitset<maxn> vx, vy;

    // initialize with vertex
    void init(int _n)
    {
        n = _n;
    }

    bool match(int i)
    {
        vx[i] = true;
        for (int j = 1; j <= n; j++)
        {
            if ((fabs(Lx[i] + Ly[j] - w[i][j]) < 1e-9)
                && !vy[j])
            {
                vy[j] = 1;
                if (!Left[j] || match(Left[j]))
                {
                    Left[j] = i;
                    return true;
                }
            }
        }
        return false;
    }

    void update()
    {
        T a = 1e9;
        for (int i = 1; i <= n; i++)
        {
            if (vx[i])
            {
                for (int j = 1; j <= n; j++)
                {
                    if (!vy[j])
                    {
                        a = min(a, Lx[i] + Ly[j] - w[i
                            ][j]);
                    }
                }
            }
        }
        for (int i = 1; i <= n; i++)
        {
            if (vx[i])
            {
                Lx[i] -= a;
            }
            if (vy[i])
            {
                Ly[i] += a;
            }
        }
    }

    void hungarian()
    {
        for (int i = 1; i <= n; i++)
        {
            Left[i] = Lx[i] = Ly[i] = 0;
            for (int j = 1; j <= n; j++)
            {
                Lx[i] = max(Lx[i], w[i][j]);
            }
        }
        for (int i = 1; i <= n; i++)
        {
            while (1)
            {
                vx.reset();
                vy.reset();
                if (match(i))
                {
                    break;
                }
                update();
            }
        }
    }
};
```

1.5 Kruskal_algorithm

```
#define maxn

template <typename T>
struct Kruskal{

  struct Edge
  {
    int from, to;
    T cost;

    Edge(int _from, int _to, int _cost)
    {
      from = _from;
      to = _to;
      cost = _cost;
    }

    bool operator< (const Edge &r) const
    {
      return cost < r.cost;
    }
  };

  int par[maxn];
  int n, m;
  T cost;
  vector<Edge> edges;
```

```
28
29    int find(int x){
30      return par[x] < 0 ? x : (par[x] = find(par[x]));
31    }
32
33    void conn(int x,int y){
34      int xx = find(x);
35      int yy = find(y);
36
37      if(xx == yy)
38        return;
39
40      par[xx] += par[yy];
41      par[yy] = xx;
42    }
43
44    void add_Edge(int from, int to, T w){
45      edges.emplace_back((Edge){from, to, w});
46    }
47
48    T kruskal_algorithm(void)
49    {
50      cost = 0;
51      memset(par, -1, sizeof(par));
52      sort(edges.begin(), edges.end());
53
54      for(int i = 0; i < m; i++)
55      {
56        Edge tmp = edges[i];
57
58        if(find(tmp.to) == find(tmp.from))
59        {
60          //不能形成環的邊
61          continue;
62        }
63
64        else
65        {
66          cost += tmp.cost;
67          conn(tmp.from, tmp.to);
68
69        }
70      }
71
72      return cost;
73    }
74  };
```

1.6 LCA

```
1  #include <bits/stdc++.h>
2  using namespace std;
3
4  #define maxn
5  #define LG //LG = log2n
6  #define PB push_back
7  #define MP make_pair
8
9  int f[LG][maxn];
10 int dep[maxn]; // dep[i] 是點 i 的深度，root 深度是 0，下一層的深度是 1...
11 int depw[maxn];
12 int n, m;
13
14 // if no weight
15 // int e[maxn];
16
17 //if the edge with weight
18 vector< pair<int, int> > e[maxn];
19
20 void dfs(int cur,int fa) { // 多帶一個父節點的參數，是在樹上 dfs 常見的技巧，可以省去平常 dfs 需要的 vis 陣列
21     f[0][cur] = fa;
22     for (auto i: e[cur]) if (i != fa) {
23         dep[i] = dep[cur]+1;
24         dfs(i, cur);
25     }
26 }
27
28 int lca(int x,int y) {
29     // 跟 swap(x,y) 是一樣的意思
30     if (dep[x] < dep[y]) return lca(y,x);
31     // 這裡開始 dep[x] >= dep[y] 一定成立
32
33     for (int i=LG-1; i>=0; i--)
34         if (dep[x]-(1<<i) >= dep[y]) // 先想辦法把 x,y 調到同深度
35             x = f[i][x];
36         if (x==y) return x; // 如果發現同深度時，是同一個點就回傳找到 LCA 了
37
38     // 否則盡量想辦法往上走，只要 x,y 同時往上走 2^i 步還不是相同的點，就 greedy 走
39     for (int i=LG-1; i>=0; i--)
40         if (f[i][x] != f[i][y])
41         {
42             x = f[i][x];
43             y = f[i][y];
44         }
45   assert(f[0][x] == f[0][y]); // 走完以後，會發現 x,y 停在 lca 的正下方一個點
46   return f[0][x];
47 }
48 void make_lca() {
49   dep[1] = depw[1] = 0;
50   dfs(1, 1); // 拿 1 當 root，且 1 的父節點是 1
51   for (int i=1; i<LG; i++)
52     for (int j=1; j<=n; j++)
53       f[i][j] = f[i-1][f[i-1][j]]; // j 往上走 2^(i-1) 再往上走 2^(i-1) = 往上走 2^i 步
54 }
55
56 int main(void)
57 {
58     while(cin >> n >> m)
59     {
60         //init
61         for(int i = 0; i < maxn; i++)
62             e[i].clear();
63
64         int x, y, z;
65
66         for(int i = 0; i < n-1; i++)
67         {
68             // if no weight
69             // cin >> x >> y;
70             // e[x].PB(y);
71             // e[y].PB(x);
72
73             cin >> x >> y >> z;
74             e[x].PB(MP(y, z));
75             e[y].PB(MP(x, z));
76         }
77
78         //make LCA
79         make_lca();
80
81         for(int i = 0; i < m; i++)
82         {
83             cin >> x >> y;
84             cout << dep[x]+dep[y]-2*dep[lca(x, y)] << '\n';
85         }
86     }
87     return 0;
88 }
```

1.7 MaxFlow

```cpp
template <typename T>
struct Dinic
{
    int n, s, t, level[M], now[M];
    struct Edge
    {
        int v;
        T rf; // rf: residual flow
        int re;
    };
    vector<Edge> e[M];
    void init(int _n, int _s, int _t)
    {
        n = _n;
        s = _s;
        t = _t;
        for (int i = 0; i <= n; i++)
        {
            e[i].clear();
        }
    }
    void add_edge(int u, int v, T f)
    {
        e[u].push_back({v, f, (int)e[v].size()});
        e[v].push_back({u, f, (int)e[u].size() - 1});
        // for directional graph
        // e[v].push_back({u, 0, (int)e[u].size() - 1})
            ;
    }
    bool bfs()
    {
        fill(level, level + n + 1, -1);
        queue<int> q;
        q.push(s);
        level[s] = 0;
        while (!q.empty())
        {
            int u = q.front();
            q.pop();
            for (auto it : e[u])
            {
                if (it.rf > 0 && level[it.v] == -1)
                {
                    level[it.v] = level[u] + 1;
                    q.push(it.v);
                }
            }
        }
        return level[t] != -1;
    }
    T dfs(int u, T limit)
    {
        if (u == t)
            return limit;
        T res = 0;
        while (now[u] < (int)e[u].size())
        {
            Edge &it = e[u][now[u]];
            if (it.rf > 0 && level[it.v] == level[u] +
                1)
            {
                T f = dfs(it.v, min(limit, it.rf));
                res += f;
                limit -= f;
                it.rf -= f;
                e[it.v][it.re].rf += f;
                if (limit == 0)
                {
                    return res;
                }
            }
            else
            {
                ++now[u];
            }
        }
        if (!res)
        {
            level[u] = -1;
        }
        return res;
    }
    T flow(T res = 0)
    {
        while (bfs())
        {
            T tmp;
            memset(now, 0, sizeof(now));
            do{
                tmp = dfs(s, INF);
                res += tmp;
            }while(tmp);
        }
        return res;
    }
};
```

1.8 maximum_dis_onTree

```cpp
#include <bits/stdc++.h>
using namespace std;

#define ll long long
#define PB push_back
#define PII pair<int, int>
#define MP make_pair
#define IOS ios_base::sync_with_stdio(false); cin.tie
    (0)
#define all(x) x.begin(), x.end()
#define REP(x, y, z) for(int x = y; x <= z; x++)
#define maxn 100000+5

//structure

//declaration
int n;
vector<PII> e[maxn];
PII first;

//functions
void dfs(int ver, int fa, int dep)
{
    if(dep > first.second)
    {
        first.first = ver;
        first.second = dep;
    }

    for(auto i : e[ver])
    {
        if(i.first != fa)
        {
            dfs(i.first, ver, dep+i.second);
        }
    }

    return;
}

int main(void)
{
    IOS;

    while(cin >> n)
    {
        //init
        REP(i, 1, n)
        {
            e[i].clear();
        }

```

```
        first.second = 0;


        int x, y, z;

        REP(i, 1, n-1)
        {
            cin >> x >> y >> z;

            e[x].PB(MP(y, z));
            e[y].PB(MP(x, z));
        }

        dfs(1, 1, 0);

        first.second = 0;

        dfs(first.first, first.first, 0);

        printf("%.1f\n", (float)(first.second)/2);
    }

    return 0;
}
```

1.9 minimumCyclewithDirectGraph_byFloyd

```
#include <bits/stdc++.h>
using namespace std;

#define maxn 100+5

int n;
int ans;
int dis[maxn][maxn];

int main(void)
{
  while(cin >> n)
  {
    ans = 1e9;
    for(int i = 1; i <= n; i++)
    {
      for(int j = 1; j <= n; j++)
      {
        cin >> dis[i][j];
        if(!dis[i][j])  dis[i][j] = 1e9;
      }
    }

    for (int k = 1; k <= n; k++)
        for (int i = 1; i <= n; i++)
          for (int j = 1; j <= n; j++)
          {
            if (dis[i][j] > dis[i][k] + dis[k][j])
              {
                // 如果可以以 k 為中繼點，更新 i,j 的最短距離
                dis[i][j] = dis[i][k] + dis[k][j];
            }

            if(i == j)
              ans = min(ans, dis[i][j]);
            }

    if(ans == 1e9)
      cout << -1 << '\n';
    else
      cout << ans << '\n';
  }
  return 0;
}
```

1.10 SPFA

```
#include <bits/stdc++.h>
using namespace std;

#define MP make_pair
#define PII pair<int, int>
#define maxn 500+5

const int INF = 1e9; //比最大可能的距離更大

bool inq[maxn]; // inq[i] 代表 i 在 queue 裡面
int dis[maxn]; // 預設都是 INF
vector<PII> e[maxn]; // (連到的點， 邊的距離)

void spfa(int cur)
{
  queue<int> q;
  dis[cur] = 0;
  q.push(cur);

  while (!q.empty())
  {
      cur = q.front();
      q.pop();
      inq[cur] = false;

      for (auto i: e[cur])
      {
        // 如果點 cur，經過權重 i.S 這條邊，走到 i.F 可以更短，就更新
          if (i.second + dis[cur] < dis[i.first])
          {
            dis[i.first] = dis[cur] + i.second;
            if (!inq[i.first])
            {
              inq[i.first] = true;
              q.push( i.first );
            }
        }
      }
  }
}

void init(void)
{
  fill(dis, dis+maxn, INF);
  for(int i = 0; i < maxn; i++)
  {
    e[i].clear();
  }

  memset(inq, false, sizeof(inq));
}
```

1.11 SPFA_withNagtiveCycle

```
#include <bits/stdc++.h>
using namespace std;

#define MP make_pair
#define PII pair<int, int>
#define maxn 500+5

const int INF = 1e9; //比最大可能的距離更大

bool inq[maxn]; // inq[i] 代表 i 在 queue 裡面
int dis[maxn]; // 預設都是 INF
int updateCount[maxn];
int vis[maxn];
vector<PII> e[maxn]; // (連到的點， 邊的距離)
int n, m;

void spfa(int cur)
```

```
{
  queue<int> q;
  dis[cur] = 0;
  q.push(cur);

  while (!q.empty())
  {
      cur = q.front();
      q.pop();
      inq[cur] = false;

      for (auto i: e[cur])
      {
        // 如果點 cur，經過權重 i.S 這條邊，走到 i.F 可以更短，就更新
        if (i.second + dis[cur] < dis[i.first])
        {
          dis[i.first] = dis[cur] + i.second;
          if (!inq[i.first])
          {
            // updateCount 紀錄一個點被放到 queue 幾次
            updateCount[i.first]++;
            if(updateCount[i.first] > n)
            {
              continue;
            }
              inq[i.first] = true;
              q.push( i.first );
          }
        }
      }
  }
}

void init(void)
{
  fill(dis, dis+maxn, INF);
  for(int i = 0; i < maxn; i++)
  {
    e[i].clear();
  }
  memset(updateCount, 0, sizeof(updateCount));
  memset(inq, false, sizeof(inq));
}

bool dfs(int cur)
{
    vis[cur]=true;
    if(cur==n)return true;

    for(int i = 0; i < e[cur].size(); i++)
        if(!vis[e[cur][i].first])
            if(dfs(e[cur][i].first))
                return true;
    return false;
}

bool check()
{
    memset(vis,false, sizeof(vis));
    for(int i = 1; i <= n; i++)
        if(updateCount[i]>n && dfs(i))
            return true;
    return false;
}

int main(void)
{
  int x, y, z;
  while(cin >> n >> m)
  {
    init();

    for(int i = 0; i < m; i++)
    {
      cin >> x >> y >> z;
      e[x].push_back(MP(y, z));
    }

    spfa(1);

    if(dis[n]!=INF && !check())
      cout << dis[n] << '\n';
    else
      cout << "There a negative cycle or no path\n";
  }
  return 0;
}
```

2 DataStructure

2.1 BIT

```
// C++ code to demonstrate operations of Binary ind Tree
#include <bits/stdc++.h>
using namespace std;
#define maxn 10000

int BIT[maxn+5];
int node[maxn+5] = {2, 1, 1, 3, 2, 3, 4, 5, 6, 7, 8, 9};
int n = 12; //# of node

int getsum(int ind)
{
  int sum = 0;
  ind = ind + 1;
  while (ind>0) {
    sum += BIT[ind];
    ind -= ind & (-ind);
  }
  return sum;
}

void update(int ind, int val)
{
  ind = ind + 1; //bcz BIT ind from 1
  while (ind <= n) {
    BIT[ind] += val;
    ind += ind & (-ind);
  }
}

void init()
{
  memset(BIT,0,sizeof(BIT));
  for (int i=0; i<n; i++)
    update(i, node[i]);
}

int main()
{
  init();
  getsum(5);//sum of arr[0~5]
  // update both arr && BIT of node[3]
  node[3] += 100;
  update(3, 100);
  return 0;
}
```

2.2 DisjoinSet

```
#define SIZE 10000

struct disjoint{

  int arr[SIZE+10];
```

```
6
7   void init(){
8     memset(arr,-1,sizeof(arr));
9   }
10
11  int find(int k){
12    return arr[k]<0 ? k:(arr[k]=find(arr[k]));
13  }
14
15  void uni(int a, int b){
16    a = find(a);
17    b = find(b);
18    arr[a] += arr[b];
19    arr[b] = a;
20  }
21
22  int siz(int k){
23    return (-arr[find(k)])
24  }
25
26 }D;
```

2.3 lower_bound

```
1  #include <iostream>
2  #include <cstdio>
3  #include <algorithm>
4  #include <vector>
5  using namespace std;
6
7  #define maxn 10000
8
9  int main(void)
10 {
11   int n;
12   int find;
13   vector <int> arr;
14
15   scanf("%d", &n);
16
17   for(int i = 0; i < n; i++)
18   {
19     int tmp;
20     scanf("%d", &tmp);
21     arr.push_back(tmp);
22   }
23
24   sort(arr.begin(), arr.end());
25
26   scanf("%d", &find);
27
28   //兩者都在<algorithm> header file
29
30   //找數字是否在array裡面 true = 1, false = 0
31   cout << binary_search(arr.begin(), arr.end(), find)
         << endl;
32
33   //找大於或等於那個數的最小'位子'
34   printf("%d\n", *lower_bound(arr.begin(), arr.end(),
         find));
35
36   return 0;
37 }
```

2.4 Priority_Queue

```
1  priority_queue<int>
2  priority_queue<int , vector<int>, greater<int> > // 由
       小到大 int 可替換為其他形態
```

2.5 Segment_Tree

```
1  int st[maxn];
2  int st_val[maxn];
3
4  void build(int now, int l, int r){
5    if(l == r)
6      st[now] = st_val[l];
7    else{
8      build(now*2, l, (l+r)/2);
9      build(now*2+1, (l+r)/2+1), r);
10     st[now] = max(st[now*2], st[now*2]+1);
11   }
12 }
13
14 int query(int now, int l, int r, int t_l, int t_r){
15
16   int tmp = -1e9;
17
18   if(t_l <= l && r <= t_r)
19     return st[now];
20
21   if(t_l <= (l+r)/2)
22     tmp = max(tmp, query(l, (l+r)/2, t_l, t_r));
23
24   if(t_r > (l+r)/2)
25     tmp = max(tmp, query((l+r)/2+1, r, t_l, t_r));
26
27   return tmp;
28 }
29
30 void update(int now, int l, int r, int tar){
31
32   if(l == r){
33     st[now] = st_val[l];
34     return;
35   }
36
37   if(tar <= (l+r)/2)
38     update(l, (l+r)/2, tar);
39
40   if(tar > (l+r)/2)
41     update((l+r)/2+1, r, tar);
42
43   st[now] = max(st[now*2], st[now*2]+1);
44 }
```

3 DP

```
1  #include<bits/stdc++.h>
2
3  using namespace std;
4
5  #define size 4
6
7  int arr[size][size];
8
9  int maxSubArr(){
10
11   int b[size];
12   int MAX = -11111111;
13
14   for(int i = 0 ; i < size; i++){
15
16     memset(b, 0, sizeof(b));
17     for(int j = i ; j < size ; j++){
18
19       int s = 0;
20       for(int k = 0 ; k < size ; k++){
21
22         b[k] += arr[j][k];
23         s += b[k];
24         if(s <= 0)
25           s = b[k];
```

```
        if(s > MAX)
          MAX = s;
      }
    }
  }
  return MAX;
}

int main(){

  #ifdef DBG
  freopen("1.in", "r", stdin);
  freopen("2.out", "w", stdout);
    #endif

  for(int i = 0 ; i < size ; i++)
    for(int j = 0 ; j < size ; j++)
      cin >> arr[i][j];

  maxSubArr();

  return 0;
}
```

3.1 Bag

```
#include <bits/stdc++.h>

#define W 1000 // 背包最重 W
#define N 100  // 最多 N 種物品

int weight[N]; //物品重量
int value[N];  //物品價值
int bag[W][2];

// -----01背包-----
void ZeroOne(){
  memset(bag, 0, sizeof(bag));
  for (int i = 0; i < N; i++){
    for (int j = 0; j < W; j++)
      if (j >= weight[i])
        bag[j][1] = max(bag[j][0], bag[j - weight[i
            ]][0] + value[i]);
    for (int j = 0; j < W; j++)
      bag[j][0] = bag[j][1];
  } // end for i
}// end ZeroOne

// -----無限背包-----
void Unlimited(){
  memset(bag, 0, sizeof(bag));
  for (int i = 0; i < N; i++){
    for (int j = 0; j < W; j++)
      if (j >= weight[i])
        bag[j][1] = max(bag[j][0], bag[j - weight[i
            ]][1] + value[i]);
    for (int j = 0; j < W; j++)
      bag[j][0] = bag[j][1];
  } // end for i
} // end Unlimited

// -----多重背包-----
int limit[N]; // 物品上限
void Multiple(){
  for (int i = 0; i < N; i++){
    int tmp = 1;
    while (tmp <= weight[i]){
      for (int j = 0; j < W; j++)
        if (j >= weight[i] * tmp)
          bag[j][1] = max(bag[j - weight[i] * tmp][0] +
              value[i] * tmp, bag[j][0]);
      for (int j = 0; j < W; j++)
        bag[j][0] = bag[j][1];
      weight[i] = weight[i] - tmp;
      tmp = tmp * 2;
    } // end while
    if (weight[i] > 0){
      for (int j = 0; j < W; j++)
        if (j >= weight[i] * tmp)
          bag[j][1] = max(bag[j - weight[i] * tmp][0] +
              value[i] * tmp, bag[j][0]);
      for (int j = 0; j < W; j++)
        bag[j][0] = bag[j][1];
    } // end if
  } // end for i
} // end Multiple

// -----分組背包-----
int group;    // 有幾組
int how_many; // 一組幾個
int WEIGHT,VALUE;

void Grouping(){
  memset(bag,0,sizeof(bag));
  for(int i = 0 ; i < group ; i++ ){
    for(int j = 0 ; j < how_many ; j++ ){
      scanf("%d %d", &WEIGHT, &VALUE);
      for(int k = 0 ; k < W ; k++ ){
        if( j >= WEIGHT ){
          bag[j][1] = max( bag[j][1] , bag[j][0] );
          bag[j][1] = max( bag[j][1] , bag[j-WEIGHT][0]
              + VALUE );
        } // end if
      } // end for k
    } // end for j
    for(int j = 0 ; j < W ; j++ )
      bag[j][0] = bag[j][1];
    } // end for i
} // end Goruping
```

3.2 DP_coin_change

```
#include<bits/stdc++.h>

using namespace std;

int coin[] = {1, 5, 10, 25, 50};
int arr[100000];

void build(){

  memset(arr, 0, sizeof(arr));
  arr[0] = 1;
  for (int i = 0; i < 5; i++){

    for (int j = 1; j < 10000; j++){

      if (j >= coin[i]){

        arr[j] = arr[j] + arr[j - coin[i]];
      }
    }
  }
}
```

3.3 LCS

```
#include<bits/stdc++.h>

using namespace std;

int arr[1500][1500];

void LCS(string str1, string str2){

  memset(arr, 0, sizeof(arr));
  for (int i = 1; i <= str1.size(); i++){

```

```
    for (int j = 1; j <= str2.size(); j++){

      if (str1[i - 1] == str2[j - 1]){

        arr[i][j] = arr[i - 1][j - 1] + 1;
      }
      else{

        arr[i][j] = max(arr[i - 1][j], arr[i][j - 1]);
      }
    }
  }
}
```

3.4 LIS

```
if(lis.size()==0||tmp>lis.back())
    lis.push_back(tmp);
else
    *lower_bound(lis.begin(),lis.end(),tmp)=tmp;
```

3.5 minCoinChange

```
#include <bits/stdc++.h>
using namespace std;

#define maxn 100000
#define maxm 100000

int coins[maxn];
int table[maxm];
int n, target;

int minCoins(int n, int tar)
{
    table[0] = 0;

    for (int i = 1; i <= tar; i++)
        table[i] = 1e9;

    // Compute minimum coins required for all
    // values from 1 to V
    for (int i = 1; i <= tar; i++)
    {
        // Go through all coins smaller than i
        for (int j=  0; j < n; j++)
            if (coins[j] <= i)
            {
                int sub_res = table[i-coins[j]];
                if (sub_res != 1e9 && sub_res + 1 <
                    table[i])
                    table[i] = sub_res + 1;
            }
    }
}

int main(void)
{
    while(cin >> n >> target)
    {
        for(int i = 0; i < n; i++)
        {
            cin >> coins[i];
        }

        minCoins(n, target);
    }
}
```

4 Math

4.1 baby-giant-step

```
#include <bits/stdc++.h>
using namespace std;
#define PROBLEM "10225"
#define LL long long
#define USE_CPPIO() ios_base::sync_with_stdio(0); cin.
    tie(0)

LL pow(LL a, LL n, LL p)    // a^n % p
{
    LL ans = 1;
    while(n)
    {
        if(n & 1) ans = ans * a % p;
        a = a * a % p;
        n >>= 1;
    }
    return ans;
}

int main(int argc, char const *argv[])
{
  LL P,A,B; //query i that A^i := B (mod P)
  while(cin>>P>>A>>B){
    LL sqp = sqrt(P);
    map <LL,int> m;
    bool fians = false;
    LL t = 1;
    for (int i = 0; i < sqp; ++i) //baby step
    {
      if (t==B)
      {
        printf("%d\n",i );
        fians = true;
        break;
      }
      m[t]=i+1; //bcz case i=0
      t = t*A % P;
    }
    if (fians)
      continue;
    fians = false;

    LL inv_an = pow(t,P-2,P);
    t = inv_an*B % P;
    //assume A^(i*n+k):=B -> A^k := B*A*(a^n)^-i
    for (int i = 1; i <= sqp; ++i) //giant step
    {
      int te = m[t % P];
      if (te!=0)
      {
        printf("%d\n",i*sqp+(te-1) );
        fians = true;
        break;
      }
      t = t*inv_an % P;
    }
    if (!fians)
    {
      printf("no solution\n");
    }
  }
  return 0;
}
```

4.2 china_mod

```
#include <iostream>

using namespace std;
//参数可为负数的扩展欧几里德定理
```

```cpp
void exOJLD(int a, int b, int & x, int & y) {
    //根据欧几里德定理
    if (b == 0) { //任意数与0的最大公约数为其本身。
        x = 1;
        y = 0;
    } else {
        int x1, y1;
        exOJLD(b, a % b, x1, y1);
        if (a * b < 0) { //异号取反
            x = -y1;
            y = a / b * y1 - x1;
        } else { //同号
            x = y1;
            y = x1 - a / b * y1;
        }
    }
}
//剩余定理
int calSYDL(int a[], int m[], int k) {
    int N[k]; //这个可以删除
    int mm = 1; //最小公倍数
    int result = 0;
    for (int i = 0; i < k; i++) {
        mm *= m[i];
    }
    for (int j = 0; j < k; j++) {
        int L, J;
        exOJLD(mm / m[j], -m[j], L, J);
        N[j] = m[j] * J + 1; //1
        N[j] = mm / m[j] * L; //2 1和2这两个值应该是相等的。
        result += N[j] * a[j];
    }
    return (result % mm + mm) % mm; //落在(0, mm)之间，这么写是为了防止result初始为负数，本例中不可能为负可以直接 写成：return result%mm;即可。
}

int main() {
    int a[3] = {2,3,6}; //a[i]=n%m[i]
    int m[3] = {3,5,7};
    cout << calSYDL(a, m, 3) << endl; //輸出為滿足兩條陣列的最小n,第3參數為陣列長度
    //所有滿足答案的數字集合為n+gcd(m0,m1,m2...)*k, k為正數
    return 0;
}
```

4.3 Convex Hull-Andrew's $_{M}onotone_{C}hain$

```cpp
#include <bits/stdc++.h>
using namespace std;
#define N 10
// P為平面上散佈的點。設定為N點。
// CH為凸包上的頂點。設定為逆時針方向排列。可以視作一個stack。
struct Point {int x, y;} P[N], CH[N+1];

// 向量OA叉積向量OB。大於零表示從OA到OB為逆時針旋轉。
double cross(Point o, Point a, Point b)
{
    return (a.x - o.x) * (b.y - o.y) - (a.y - o.y) * (b.x - o.x);
}

// 小於。依座標大小排序，先排 x 再排 y。
bool compare(Point a, Point b)
{
    return (a.x < b.x) || (a.x == b.x && a.y < b.y);
}

int Andrew_monotone_chain()
{
    // 將所有點依照座標大小排序
    sort(P, P+N, compare);

    int m = 0;  // m 為凸包頂點數目

    // 包下半部
    for (int i=0; i<N; ++i)
    {
        while (m >= 2 && cross(CH[m-2], CH[m-1], P[i]) <= 0) m--;
        CH[m++] = P[i];
    }

    // 包上半部，不用再包入剛才包過的終點，但會再包一次起點
    for (int i=N-2, t=m+1; i>=0; --i)
    {
        while (m >= t && cross(CH[m-2], CH[m-1], P[i]) <= 0) m--;
        CH[m++] = P[i];
    }
    m--;    // 最後一個點是重複出現兩次的起點，故要減一。

    return m;
}

double cac_area(int m){ //有m個點
    double re=0.0,t=0.0;
    while(m>=2)
    {
        t = cross( CH[0] ,CH[m-1], CH[m-2])/2.0;
        if(t<0) t=-t;
        m--;
        re=re+t;
    }
    return re;
}

bool inside(Point pd, int m){ //pd為查詢點, 凸包有m個點
    int t,pt=INT_MAX;
    while(m>=2)
    {
        int tmp = cross( pd ,CH[m-1], CH[m-2]);
        if( tmp > 0){
            t=1;
        }
        else if (tmp < 0 ){
            t=-1;
        }

        if ( t!=pt && pt!=INT_MAX)
        {
            return false;
        }
        pt = t;
        m--;
    }
    return true;
}
```

4.4 eularphi_request

```cpp
#include<bits/stdc++.h>
using namespace std;
#define maxn 46340 //bcz sqrt(2^31-1)~=46340.95 and 46341 not prime
bool prime[maxn];
void prime_table(){
  memset(prime,true,sizeof(prime));
  prime[0]=prime[1]=false;
  for (int i = 2; i < maxn; ++i)
    if (prime[i])
      for (int j = i*i; j < maxn; j+=i)
        prime[j]=false;
```

```
    }

int eularphi(int n)
{
  if (n==0) return n;
  int ans=n;
  for (int i = 2; i < maxn; ++i)
  {
    if(prime[i] && n%i==0){
      ans=ans/i*(i-1);
      while(n%i==0&&n)
        n/=i;
    }
  }
  if (n!=1){
    ans=ans/n*(n-1);
  }
  return ans;
}

int main()
{
  prime_table();
  int in;
  while(~scanf("%d",&in)){
    printf("%d\n", eularphi(in));
  }
  return 0;
}
```

4.5 eularphi_table

```
// all numbers smaller than or equal to n.
#include<iostream>
using namespace std;
#define maxn 250000
// Computes and prints totien of all numbers
// smaller than or equal to n.
void eularphi_table(int n)
{
  // Create and initialize an array to store
  // phi or totient values
  long long phi[n+1];
  for (int i=1; i<=n; i++)
    phi[i] = i; // indicates not evaluated yet
          // and initializes for product
          // formula.

  // Compute other Phi values
  for (int p=2; p<=n; p++)
  {
    // If phi[p] is not computed already,
    // then number p is prime
    if (phi[p] == p)
    {
      // Phi of a prime number p is
      // always equal to p-1.
      phi[p] = p-1;

      // Update phi values of all
      // multiples of p
      for (int i = 2*p; i<=n; i += p)
      {
      // Add contribution of p to its
      // multiple i by multiplying with
      // (1 - 1/p)
      phi[i] = (phi[i]/p) * (p-1);
      }
    }
  }

  // Print precomputed phi values
  for (int i=1; i<=n; i++)
  cout << "Totient of " << i << " is "<< phi[i] << '\n'
      ;
}

int main()
{
  freopen("o.out","w",stdout); //for test
  int n = maxn;
  eularphi_table(n);
  return 0;
}
```

4.6 Fibonacci_log(n)_requset

```
#include <iostream>
#include <cstring>
using namespace std;
#define LL long long
//注意，f0=1,f1=1,f2=2...
const LL mod=1e9+7; // 避免數值過大造成 overflow，因此將所有數值都 mod 10^9+7

struct Matrix  {
  LL a[2][2];
  void all_0() // 清空矩陣
  {
    memset(a, 0, sizeof(a));
  }
  void I() // 讓矩陣變成單位方陣
  {
    a[0][0]=1; a[0][1]=0;
    a[1][0]=0; a[1][1]=1;
  }
  void X() // 讓矩陣變成文章中的矩陣 A
  {
    a[0][0]=1; a[0][1]=1;
    a[1][0]=1; a[1][1]=0;
  }
};

Matrix operator*(const Matrix &a, const Matrix &b) // 矩陣乘法
{
  Matrix ret;
  ret.all_0();
  for (LL i=0; i<2; i++)  {
    for (LL j=0; j<2; j++)  {
      for (LL k=0; k<2; k++)  {
        ret.a[i][j]+=a.a[i][k]*b.a[k][j];
        ret.a[i][j]%=mod;
      }
    }
  }
  return ret;
}

Matrix power(Matrix a, LL n) // 快速冪
{
  Matrix ret;
  ret.I();
  if (n==0)  return ret;
  ret.X();
  if (n==1)  return ret;
  ret=power(a, n/2);
  ret=ret*ret;
  if (n%2==1)  ret=ret*a;
  return ret;
}

LL query(LL n)
{
  Matrix tmp;
  tmp.X();
  tmp=power(tmp, n);
  LL ret=tmp.a[1][0]+tmp.a[1][1]; // 因為初始的矩陣 X[0] 的兩個元的值都是 1，所以矩陣相乘的結果相當於把矩陣 A 下面的兩個元加起來
```

```
  ret%=mod;
  return ret;
}

int main()
{
  LL n;
  while (cin >> n)  {
    cout << query(n) << endl;
  }
  return 0;
}
```

4.7 josephus

```
int josephus(int n, int k){ // 有n個人圍成一圈,每k個一次
  return n > 1 ? (josephus(n-1,k)+k)%n : 0;
}// 回傳最後一人的編號, 0 index
```

4.8 mod_inv

```
#include <stdio.h>
#define LL long long
const LL maxn(1000005), mod(1000000000 + 7);
//ax == 1 (mod m),query x -> use exgcd
//if m is prime ,than x = a^(p-2) % p

LL inv[maxn]; //for mod_inv table
LL Jc[maxn];  //for factorial table
void calJc()    //factorial table
{
    Jc[0] = Jc[1] = 1;
    for(LL i = 2; i < maxn; i++)
        Jc[i] = Jc[i - 1] * i % mod;
}

//section 1 for exgcd start
void exgcd(LL a, LL b, LL &x, LL &y)
{
    if(!b) x = 1, y = 0;
    else
    {
        exgcd(b, a % b, y, x);
        y -= x * (a / b);
    }
}
LL niYuan(LL a, LL b)//ax+by = 1
{
    LL x, y;
    exgcd(a, b, x, y);
    return (x + b) % b;
}
//section 1 for exgcd end

//section 2 for fema's theorem start
LL pow(LL a, LL n, LL p)    // a^n % p
{
    LL ans = 1;
    while(n)
    {
        if(n & 1) ans = ans * a % p;
        a = a * a % p;
        n >>= 1;
    }
    return ans;
}
// LL niYuan(LL a, LL b)   //if mod(b) is prime we use fema's theorem
// {
//     return pow(a, b - 2, b);
// }
// // section 2 for fema's theorem end

//section 3 for tableing inverse start
void mod_inv_table(){
    inv[1]=1;
    for (int i = 2; i < maxn; ++i)
    {
        inv[i] = (mod-mod/i)*inv[mod%i] %mod;
    }
}
//section 3 for tableing end

LL C(LL a, LL b)    //cac C(a, b)
{
    // printf("Jc[%lld] = %lld\n",a,Jc[a] );
    return Jc[a] * niYuan(Jc[b], mod) % mod
        * niYuan(Jc[a - b], mod) % mod;
}

int main(int argc, char const *argv[])
{
    calJc();    //if need cac C(a,b)
    // printf("%lld\n",C(1000,2) );
    mod_inv_table();    //if need table of mod_inv from 1~mod-1
    return 0;
}
```

4.9 prime_table

```
#define maxn 46340
//bcz sqrt(2^31-1)~=46340.95 and 46341 46340 not prime
bool prime[maxn];
void prime_table(){
  memset(prime,true,sizeof(prime));
  prime[0]=prime[1]=false;
  for (int i = 2; i < maxn; ++i)
    if (prime[i])
      for (int j = i*i; j < maxn; j+=i)
        prime[j]=false;
    }
```

4.10 prime_table_linear

```
#include <bits/stdc++.h>
using namespace std;
#define N 1000000
long long int not_prime[N];
vector<long long int> prime;
void prime_sieve(){
    memset(not_prime,0,sizeof(not_prime));
    not_prime[1]=1;
    for(long long int i=2;i<N;i++){
        if(!not_prime[i]){
            prime.push_back(i);
        }
        for(long long int j=0;j<prime.size()&&i*prime[j]<N;j++){
            not_prime[i*prime[j]]=1;
            if(i%prime[j]==0)
                break;
        }
    }
}
```

4.11 topology_sort

```
#include <bits/stdc++.h>
#define maxn 50005
using namespace std;
struct edge
{
```

```
    int t,next;
}   in[maxn*4];
//n vertex has n*4 maximum edges

int n,m,e,first[maxn],s[maxn],top;
//  first   紀錄是否有固定順序
//  s 紀錄順序

bool fail,ins[maxn],vis[maxn];
//  vis 是否訪問
//  ins 在做dfs的當下 那點是否被訪問過

void add(int x,int y)
{
    in[e].t=y;
    in[e].next=first[x];
    first[x]=e++;
}
void dfs(int cur)
{
    ins[cur]=vis[cur]=true;
    for(int i=first[cur]; ~i; i=in[i].next)
    {
        if(!vis[in[i].t])
            dfs(in[i].t);
        else if(ins[in[i].t])
            fail=true;
    }
    ins[cur]=false;
    s[top++]=cur;
}
int main(void)
{
    int x,y;
    while(cin >> n >> m)
    {
        //init
        e = 0;
        top = 0;
        fail = false;
        memset(first, -1, sizeof(first));
        memset(ins, false, sizeof(ins));
        memset(vis, false, sizeof(vis));

        for(int i = 1; i <= m; i++)
        {
            scanf("%d %d",&x,&y);
            add(x,y);
        }

        for(int i = 1; i <= n; i++)
            if(!vis[i])
                dfs(i);

        if(fail)
            puts("-1");
        else
            for(int i = top-1; i >= 0; i--)
                printf("%d\n",s[i]);
    }
    return 0;
}
```

4.12 line_intersection

```
struct Node{
    double x,y;
};
struct Line {
    double ax,ay,bx,by;
    Line(){}
    Line(Node A, Node B){
        ax = A.x,ay = A.y;
        bx = B.x,by = B.y;
    }
};
double cross(Line K, double a, double b) {
    double cross1 = (a - K.ax)*(K.by - K.ay);
    double cross2 = (b - K.ay)*(K.bx - K.ax);
    return (cross1 - cross2);
}
bool intersection(Line A, Line B){
    //快速排斥實驗
    if ( max(A.ax,A.bx) < min(B.ax,B.bx) ||
         max(A.ay,A.by) < min(B.ay,B.by) ||
         max(B.ax,B.bx) < min(A.ax,A.bx) ||
         max(B.ay,B.by) < min(A.ay,A.by) ){
        return false;
    }
    //跨立實驗
    if ( cross(B,A.ax,A.ay)*cross(B,A.bx,A.by) > 0 ||
         cross(A,B.ax,B.ay)*cross(A,B.bx,B.by) > 0 ){
        return false;
    }
    return true;
}
```

5 Misc

5.1 big_integer

```
import java.io.*;
import java.util.Scanner;
import java.math.BigInteger;

public class Main
{
  public static void main(String[] argv)
  {
    Scanner scanner = new Scanner(System.in);

    while(scanner.hasNext())
    {
      String input = scanner.next();
      String input2 = scanner.next();

      BigInteger a = new BigInteger(input);
      BigInteger b = new BigInteger(input2);

      System.out.println("Add: " + a.add(b));
      System.out.println("Sub: " + a.subtract(b));
      System.out.println("Mul: " + a.multiply(b));
      System.out.println("Div: " + a.divide(b));
    }
  }
}
```

5.2 romanToInt

```
unordered_map<char, int> value{{'I', 1}, {'V', 5}, {'X', 10}, {'L', 50}, {'C', 100}, {'D', 500}, {'M', 1000}};

int romanToInt(string s){

  if(s.empty())
    return 0;

  int maxDigit = -1;
  int ans = 0;
  for(int i = s.size()-1 ; i >= 0 ; i--){

    const int current = value[s[i]];
    if(current >= maxDigit){

      ans += value[s[i]];
      maxDigit = current;
```

```
    }
    else{

      ans -= value[s[i]];
    }
  }
  return ans;
}
```

5.3 SubFactorial

```
FAC = list()
MAXN = 100

OUTFILE = open("output.out","w")

# n!
def build_fac():
    global FAC
    FAC.append(1) # FAC[0] = 1
    for i in range(1,MAXN+5):
        FAC.append(FAC[i-1]*i)
    for i in range(1,MAXN):
        print(FAC[i],end=" ",file=OUTFILE)
        if i%10 == 0:
            print("",file=OUTFILE)

# !n
# !2 : (2,1)
# !3 : (3,1,2),(2,3,1)
# !4 : (2,1,4,3),(2,3,4,1),(2,4,1,3),(3,1,4,2)
    ,(3,4,1,2),
#       (3,4,2,1),(4,1,2,3),(4,2,1,2),(4,3,2,1)
SUBFAC = list()
def build_subfac() :
    global SUBFAC
    SUBFAC.append(1) # SUBFAC[0] = 1
    SUBFAC.append(0) # SUBFAC[1] = 0
    for i in range(2,MAXN+5):
        SUBFAC.append(0)
        for j in range(0,i+1):
            tmp = 1
            for k in range(i,j,-1):
                tmp *= k
            if j&1 :
                SUBFAC[i] -= tmp
            else :
                SUBFAC[i] += tmp

    for i in range(1,MAXN):
        print(SUBFAC[i],end=" ",file=OUTFILE)
        if i%10 == 0:
            print("",file=OUTFILE)

build_fac()
print("",file=OUTFILE)
print("",file=OUTFILE)
build_subfac()
```

5.4 template

```
#include <bits/stdc++.h>
using namespace std;

#define ll long long
#define PB push_back
#define PII pair<int, int>
#define MP make_pair
#define all(x) x.begin(), x.end()
#define REP(x, y, z) for(int x = y; x <= z; x++)
#define maxn

//structure

//declaration

//functions

int main(void)
{


  return 0;
}
```

5.5 pymodel

```
### EOF
while True:
  try:
    pass
  except EOFError:
    break

### input output

fp = open("in.txt", "r")
fout = open("out.txt", "w")

for testData in fp.readlines():
  pass

#在這裡end的動作會把換行去掉
print("name: {}".format(1), end="", file=fout)

aa = input()
##input: ICEJJ RealYami
ICEJJ, RealYami = input.split()

##input: 3 5
num1, num2 = input.split()
# 做轉型

###初始化list
arr = [0 for i in range(10)]
###陣列元素轉型
arr = [int(x) for i in arr]

###numpy
import numpy as np

arr = np.zeros(shpape = (10, 20))
arr_2 = np.ones(shpape = (20))
arr.T#轉至矩陣

A * B#位置1對1相乘
A.dot(B)#矩陣A * 矩陣B

A + B #矩陣相加
np.array_equal(A, B)#兩矩陣是否一模一樣
np.sum(A) #A所有元素相加
np.sum(A, axis=0)#col 相加
np.sum(A, axis=1)#row 相加W
np.min(A, axis=0)
np.max(A, axis=1)

np.count_nonzero(A)#計算0的數量

###math
import math

math.ceil(x)#上高斯
math.floor(x)#下高斯
math.factorial(x)#接乘
math.fabs(x)#絕對值
math.fsum(arr)#跟sum一樣但更精確(小數點問題)
math.gcd(x, y)#bj4
```

```
math.exp(x)#e^x
math.log(x, base)
math.log2(x)#2為底
math.log10(x)#10為底
math.sqrt(x)
math,pow(x, y)#精確些(float型態)
math.sin(x)# cos tan asin acos atan atan2(弧度) sinh
    cosh tanh acosh asinh atanh
math.hypot(x, y)#歐幾里德範數
math.degrees(x)#x從弧度轉角度
math.radians(x)#x從角度轉弧度
math.gamma(x)#x的gamma函數
math.pi#常數
math.e#常數
math.inf

### ascii

ord(x)#char to asc
chr(x)#asc to char

x.encode().hex()#string to hex


### bin oct hex
x = "11"

int(x, 2)

x = "F"

int(x, 16)

### str

string.isdigit()
string.isalpha()
### reverse string
string = "abc"
string_reverse = string[::-1]
```

5.6 remain

```
#include <bits/stdc++.h>
using namespace std;

// BKDR Hash Function
unsigned int BKDRHash(const char *str)
{
    unsigned int seed = 131; // 31 131 1313 13131
        131313 etc..
    unsigned int hash = 0;

    while (*str){
        hash = hash * seed + (*str++);
    }

    return (hash & 0x7FFFFFFF);
}

int main(){

    bitset<40> b;//建10格大小的bitset
    int k = 20;
    bitset<40> bk(k);//將int直接轉為bitset
    string str = "10100110"
    bitset<40> bstr(str);//將string直接轉為bitset
    // bstr[0] = 1, bstr[1] = 0, bstr[2] = 1, ... ,
        bstr[6] = 1, bstr[7] = 0;

    b.reset();//每個位元設0
    b.set();//每個位元設1
    b[pos] = 1;//第pos個位元設成1
    b.count();//有幾個1
    b.size();//大小

    cin >> b;
    //input: "0110xx0011"
    //b = "0110"
    bitset<8> bb(string("1001"));
    cout << bb ;// 00001001

    list <int> L;
    int num = 1;
    L.insert(L.begin(),num);
    for(auto i : L)
        cout << i <<endl;
    L.earse(L.begin());
    L.remove(num);

}
```

5.7 avltree

```
#include <bits/stdc++.h>

#include <ext/pb_ds/assoc_container.hpp> // Common file
#include <ext/pb_ds/tree_policy.hpp>

using namespace __gnu_pbds;
using namespace std;


/*
<<declare>>
tree<
    key_type,
    mapped_type,
    Cmp_func(default std::less<Key>),
    rb_tree_tag,
    tree_order_statistics_node_update
    >
*/

tree<int, null_type, less<int>, rb_tree_tag,
    tree_order_statistics_node_update> p;
//tree<int, null_type, less<int>, rb_tree_tag,
    tree_order_statistics_node_update>::iterator it;
auto it = t.begin();
//支援操作：lower_bound, upper_bound, insert, erase,
    find, begin, end, size
//<<insert>>
tree.insert(key)
p.insert(1);

// The index is from 0 to ...
// <<find by order>>
// 找第k+1小的數字，如果沒有，返回end().+'*'取數字
*p.find_by_order(k) ->返回該數字
p.find_by_order(k) ->返回該數字的位置

// <<order of key>>
// 查詢比k小的數字的個數
p.order_of_key(k)

//<find>
t.find(x) // 返回x所在的it
```