Team Notebook

Metropolitan University

March 7, 2024

Contents

1 Array

1.1 Kadane in circular array

```
int maxSubarraySumCircular(vector<int> &nums)
{
    int mxsf = -30000, mnsf = 30000, sum = 0, csum = 0;
    for (int x : nums)
    {
        csum += x; sum += x;
        mxsf = max(mxsf, csum);
        csum = max(csum, 0);
    }
    csum = 0;
    for (int x : nums)
    {
        csum += x;
        mnsf = min(mnsf, csum);
        if (csum > 0) csum = 0;
    }
    if (mnsf == sum) mnsf = 0;
    return max(mxsf, sum - mnsf);
}
```

1.2 Kadane

```
int maxSubArraySum(int a[], int size)
{
    int max_so_far = INT_MIN, max_ending_here = 0;

    for (int i = 0; i < size; i++) {
        max_ending_here = max_ending_here + a[i];
        if (max_so_far < max_ending_here)
            max_so_far = max_ending_here;

        if (max_ending_here < 0)
            max_ending_here = 0;
    }
    return max_so_far;
}
```

2 Boilerplate

```
#include<bits/stdc++.h>
using namespace std;
#define pi acos(-1)
#define MOD 1000000007
#define inf 1000000010
#define endl "\n"
#define ull unsigned long long
#define con (f?"YES":"NO")
#define yes cout<<"YES"<<endl
#define no cout<<"NO"<<endl

#define dpos(n) fixed << setprecision(n)

#define clear1(a) memset(a, -1, sizeof(a))
#define clear0(a) memset(a, 0, sizeof(a))

#define sortn(a,x,n) sort(a+x, a+x+n)
#define sortv(s) sort(s.begin(), s.end())
#define reversev(s) reverse(s.begin(), s.end())
#define rsortv(s) sort(s.rbegin(),s.rend())
#define unik(a) unique(a.begin(), a.end()) - a.begin()
#define iotav(s, x) iota(s.begin(), s.end(), x)

#define lowerbound(v,x) lower_bound(v.begin(), v.end(), x)-v
    .begin()
#define upperbound(v,x) upper_bound(v.begin(), v.end(), x)-v
    .begin()

#define pb push_back
#define loj(i,j) "Case "<<i<<": "<<j
#define gap " "

#define auto(x,a) for (auto& x : a)
#define print(x) cout << #x << " = " << x << endl

long long dx[] = {1, -1, 0, 0, 1, 1, -1, -1};
long long dy[] = {0, 0, 1, -1, 1, -1, 1, -1};

int main() {
  ios_base::sync_with_stdio(false); cin.tie(0),cout.tie(0);

  #ifndef ONLINE_JUDGE
   freopen("input.txt", "r", stdin);
   freopen("output.txt", "w", stdout);
  #endif

   long long t;
   cin >> t;
   while (t--) {
       long long n;
       cin>>n;

       long long a[n+2];
       for(long long i=1;i<=n;i++) cin>>a[i];

       for(long long i=1;i<=n;i++) cout<<a[i]<<" ";
   }
}
```

3 Combitatorics

3.1 Formula

```
Summation of squares of n natural numbers: (n*(n+1)*(2n+1))
    /6
C(n,r): n! / (r! * (n-r)!)
C(n,r): (n*(n-1)*..*(n-r+1)) / r!
P(n,k): n! / (n-k)!
-> nCk = nCn-k
-> Ways to go from (0,0) to (r,c):
(r+c)Cr or (r+c)Cc
-> Ways to go from (0,0,0) to (x,y,z): (x+y+z)Cx * (y+z)Cy
-> a1+a2+.+an = k , ai>=0: C(k+n-1,n-1)
-> Catalan Numbers:
C(n) = (2n)! / ((n+1)! * r!)
```

4 DP

4.1 Coin Changing

```
long long minNumberOfCoin(vector<long long>&v, long long val
    , long long n, vector<long long>&dp){
   if(val==0) return 0;
   if(dp[val]!=-1) return dp[val];

   long long ans = LLONG_MAX;

   for(long long i=0;i<n;i++){
       if(v[i]<=val){
           long long subAns = coinChange(v, val-v[i], n, dp)
               ;
           if(subAns!=long long_MAX && subAns+1<ans) ans =
               subAns+1;
       }
   }
```

```
    dp[val] = ans;
    return ans;
}

int numberOfWays(int coins[], int n, int sum)
{

 int dp[sum + 1];
 memset(dp, 0, sizeof(dp));
 dp[0] = 1;
 for (int i = 0; i < n; i++)
  for (int j = coins[i]; j <= sum; j++)
   dp[j] += dp[j - coins[i]];
 return dp[sum];
}
```

4.2 Knapsack

```
long long knapSack(long long w, long long i, long long *
     marks, long long *cap, vector<vector<long long>> &dp)
{
    if (i < 0)
        return 0;
    if (dp[i][w] != -1)
        return dp[i][w];

    if (cap[i] > w) dp[i][w] = knapSack(w, i - 1, marks, cap,
          dp);
    else dp[i][w] = max(marks[i] + knapSack(w - cap[i], i -
         1, marks, cap, dp), knapSack(w, i - 1, marks, cap,
         dp));

    return dp[i][w];
}
```

4.3 LIS

```
int lengthOfLIS(vector<int>& nums)
{
    short totalNums = nums.size();
    int lisSize = 0;
    vector<short>lis(totalNums+1,totalNums+10);
    lis[0] *= -1;
    for(int x:nums)
    {
        short lo = 0, high = lisSize;
        while(lo!=high)
        {
            short mid = (lo + high)/2;
            if(lis[mid]<x)
            {
                lo = mid + 1;
            }
            else
            {
                high = mid;
            }
        }
        lis[lo] = x;
        lisSize += (lo==lisSize);
    }
    return lisSize;
}
```

5 DSU

```
struct DSU
{
 int connected;
 vector<int> par, sz;

 void init(int n)
 {
  par = sz = vector<int> (n + 1, 0);
  for(int i = 1; i <= n; i++)
   par[i] = i, sz[i] = 1;
  connected = n;
 }

 int getPar(int u)
 {
  while(u != par[u])
  {
   par[u] = par[par[u]];
   u = par[u];
  }
  return u;
 }

 int getSize(int u)
 {
  return sz[getPar(u)];
 }

 void unite(int u, int v)
 {
  int par1 = getPar(u), par2 = getPar(v);

  if(par1 == par2)
   return;

  connected--;

  if(sz[par1] > sz[par2])
   swap(par1, par2);

  sz[par2] += sz[par1];
  sz[par1] = 0;
  par[par1] = par[par2];
 }
};
```

6 Data Structure

6.1 BIT with Lazy

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

template <class T>
struct Fenwick { // 1-indexed
    int n;
    vector<T> t;

    Fenwick() {}

    Fenwick(int _n) {
        n = _n;
        t.assign(n + 1, 0);
    }

    T query(int i) {
        T ans = 0;
        for (; i >= 1; i -= (i & -i))
            ans += t[i];
        return ans;
    }

    void upd(int i, T val) {
        if (i <= 0)
            return;
        for (; i <= n; i += (i & -i))
```

```
            t[i] += val;
    }

    void upd(int l, int r, T val) {
        upd(l, val);
        upd(r + 1, -val);
    }

    T query(int l, int r) {
        return query(r) - query(l - 1);
    }
};

int main() {
    long long t;
    // cin >> t;
    t = 1;
    while (t--) {
        long long n, q;
        cin >> n >> q;
        long long a[n + 2];
        Fenwick<long long> tree(n);

        for (long long i = 1; i <= n; i++)
            cin >> a[i];

        for (long long i = 1; i <= n; i++)
            tree.upd(i, i, a[i]);

        while (q--) {
            int type;
            cin >> type;
            if (type == 1) {
                long long x, y, val;
                cin >> x >> y >> val;
                tree.upd(x, y, val);
            } else {
                long long x, y;
                cin >> x;
                cout << tree.query(x) << endl;
            }
        }
    }
    return 0;
}
```

6.2 BIT

```
void update(int i, int val, int n, int *tree)
{
    while (i <= n)
    {
        tree[i] += val;
        i += (i & -i);
    }
}
//sum from 1 to i
int getSum(int i, int *tree)
{
    int sum = 0;
    while (i > 0)
    {
        sum += tree[i];
        i ^= (i & -i);
    }
    return sum;
}
```

6.3 LCA

```
const int N = 1e5 + 10;
int tin[N], tout[N];
int up[N][32];
vector<int> adj[N];
int n, lg, timer;
void dfs(int src, int par)
{
    tin[src] = ++timer;
    up[src][0] = par;
    for(int i = 1; i <= lg; i++)
    {
        up[src][i] = up[up[src][i - 1]][i - 1];
    }
    for(auto child : adj[src])
    {
        if(child != par)
            dfs(child, src);
    }
    tout[src] = ++timer;
}
bool is_ancestor(int u, int v)
{
    return tin[u] <= tin[v] && tout[u] >= tout[v];
}
int lca(int u, int v)
{
    if(is_ancestor(u, v)) return u;
    if(is_ancestor(v, u)) return v;
    for(int i = lg; i >= 0; i--)
    {
        if(!is_ancestor(up[u][i], v))
            u = up[u][i];
    }
    return up[u][0];
}
void pre_process(int root)
{
    timer = 0;
    lg = ceil(log2(n));
    dfs(root, root);
}
```

6.4 MOs

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

const int N = 1e6 + 9, B = 440;

struct query
{
    int l, r, id;
    bool operator<(const query &x) const
    {
        if (l / B == x.l / B)
            return ((l / B) & 1) ? r > x.r : r < x.r;
        return l / B < x.l / B;
    }
} Q[N];
int cnt[N], a[N];
long long sum;
inline void add_left(int i)
{
    int x = a[i];
    sum += 1LL * (cnt[x] + cnt[x] + 1) * x;
    ++cnt[x];
}
inline void add_right(int i)
{
    int x = a[i];
    sum += 1LL * (cnt[x] + cnt[x] + 1) * x;
    ++cnt[x];
}
inline void rem_left(int i)
{
    int x = a[i];
    sum -= 1LL * (cnt[x] + cnt[x] - 1) * x;
```

```
    --cnt[x];
}
inline void rem_right(int i)
{
    int x = a[i];
    sum -= 1LL * (cnt[x] + cnt[x] - 1) * x;
    --cnt[x];
}
long long ans[N];
int32_t main()
{
    ios_base::sync_with_stdio(0);
    cin.tie(0);

    int n, q;
    cin >> n >> q;
    for (int i = 1; i <= n; i++)
        cin >> a[i];
    for (int i = 1; i <= q; i++)
    {
        cin >> Q[i].l >> Q[i].r;
        Q[i].id = i;
    }
    sort(Q + 1, Q + q + 1);
    int l = 1, r = 0;
    for (int i = 1; i <= q; i++)
    {
        int L = Q[i].l, R = Q[i].r;
        if (R < l)
        {
            while (l > L)
                add_left(--l);
            while (l < L)
                rem_left(l++);
            while (r < R)
                add_right(++r);
            while (r > R)
                rem_right(r--);
        }
        else
        {
            while (r < R)
                add_right(++r);
            while (r > R)
                rem_right(r--);
            while (l > L)
                add_left(--l);
            while (l < L)
                rem_left(l++);
        }
        ans[Q[i].id] = sum;
    }
    for (int i = 1; i <= q; i++)
        cout << ans[i] << '\n';
    return 0;
}
```

6.5 Monotonic Stack

```
void printNGE(int arr[], int n)
{
    stack<int> s;

    /* push the first element to stack */
    s.push(arr[0]);

    // iterate for rest of the elements
    for (int i = 1; i < n; i++) {

        if (s.empty()) {
            s.push(arr[i]);
            continue;
        }

        /* if stack is not empty, then
           pop an element from stack.
           If the popped element is smaller
           than next, then
        a) print the pair
        b) keep popping while elements are
        smaller and stack is not empty */
        while (s.empty() == false && s.top() < arr[i]) {
            cout << s.top() << " --> " << arr[i] << endl;
            s.pop();
        }

        /* push next to stack so that we can find
        next greater for it */
        s.push(arr[i]);
    }

    /* After iterating over the loop, the remaining
    elements in stack do not have the next greater
    element, so print -1 for them */
    while (s.empty() == false) {
        cout << s.top() << " --> " << -1 << endl;
        s.pop();
    }
}
```

6.6 Segement Tree

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

void build(long long *tree, long long *a, long long node,
     long long l, long long r)
{
    if (l == r)
    {
        tree[node] = a[l];
        return;
    }

    long long left = 2 * node, right = left + 1;
    long long mid = l + (r-l)/2;

    build(tree, a, left, l, mid);
    build(tree, a, right, mid + 1, r);

    tree[node] = tree[left] + tree[right];
}

long long query(long long *tree, long long *a, long long
     node, long long l, long long r, long long begin, long
     long end)
{
    if (r < begin || end < l)
        return 0;

    if (begin <= l && r <= end)
    {
        return tree[node];
    }

    long long left = 2 * node, right = left + 1;
    long long mid = l + (r-l)/2;

    long long left_value = query(tree, a, left, l, mid, begin
         , end);
    long long right_value = query(tree, a, right, mid + 1, r,
          begin, end);

    return left_value + right_value;
}
```

```
void update(long long *tree, long long *a, long long node,
     long long l, long long r, long long index, long long
     value)
{
    if (l == r)
    {
        a[l] = value;
        tree[node] = a[l];
        return;
    }

    long long left = 2 * node, right = left + 1;
    long long mid = l + (r-l)/2;

    if (index <= mid)
        update(tree, a, left, l, mid, index, value);
    else
        update(tree, a, right, mid + 1, r, index, value);

    tree[node] = tree[left] + tree[right];
}

int main()
{
    long long n;
    cin >> n;
    long long q;
    cin >> q;

    long long a[n + 2], tree[4 * n];
    for (long long i = 1; i <= n; i++)
        cin >> a[i];

    build(tree, a, 1, 1, n);

    while (q--)
    {
        long long tt, x, y;
        cin>>tt>>x>>y;
        if(tt==1){
            update(tree, a, 1, 1, n, x, y);
            continue;
        }
        long long desire_value = query(tree, a, 1, 1, n, x, y
            );
        cout << desire_value << endl;
    }
}
```

6.7 Segment Tree with Lazy

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

void build(long long *tree, long long *lazy, long long *a,
     long long node, long long l, long long r)
{
    lazy[node] = 0;
    if (l == r)
    {
        tree[node] = a[l];
        return;
    }

    long long left = 2 * node, right = left + 1, mid = l + (r
         - l) / 2;

    build(tree, lazy, a, left, l, mid);
    build(tree, lazy, a, right, mid + 1, r);
    tree[node] = tree[left] + tree[right];
}

void propagate(long long *tree, long long *lazy, long long
     node, long long l, long long r)
{
    if (lazy[node])
    {
        tree[node] += (r - l + 1) * lazy[node];
        if (l != r)
        {
            lazy[2 * node] += lazy[node];
            lazy[2 * node + 1] += lazy[node];
        }
        lazy[node] = 0;
    }
}

long long query(long long *tree, long long *lazy, long long
     *a, long long node, long long l, long long r, long long
      b, long long e)
{
    propagate(tree, lazy, node, l, r);
    if (r < b || e < l)
        return 0;

    if (b <= l && r <= e)
        return tree[node];

    long long mid = l + (r - l) / 2;

    return query(tree, lazy, a, 2 * node, l, mid, b, e) +
         query(tree, lazy, a, 2 * node + 1, mid + 1, r, b, e)
         ;
}

void update_range(long long *tree, long long *lazy, long
     long *a, long long node, long long l, long long r, long
      long b, long long e, long long val)
{
    propagate(tree, lazy, node, l, r);
    if (r < b || e < l)
        return;
    if (b <= l && r <= e)
    {
        tree[node] += (r - l + 1) * val;

        if (l != r)
        {
            lazy[2 * node] += val;
            lazy[2 * node + 1] += val;
        }
        return;
    }

    long long mid = l + (r - l) / 2;

    update_range(tree, lazy, a, 2 * node, l, mid, b, e, val);
    update_range(tree, lazy, a, 2 * node + 1, mid + 1, r, b,
         e, val);

    tree[node] = tree[2 * node] + tree[2 * node + 1];
}

int main()
{
    long long t;
    // cin >> t;
    t=1;
    while (t--)
    {
        long long n, q;
        cin >> n >> q;
        long long a[n + 2], tree[4 * n], lazy[4 * n];

        for (long long i = 1; i <= n; i++)
            cin >> a[i];
        build(tree, lazy, a, 1, 1, n);

        while (q--)
```

```
        {
            int type;
            cin >> type;
            if (type == 1)
            {
                long long x, y, val;
                cin >> x >> y >> val;
                update_range(tree, lazy, a, 1, 1, n, x, y, val
                     );
                continue;
            }
            long long x;
            cin >> x;
            cout << query(tree, lazy, a, 1, 1, n, x, x) <<
                 endl;
        }
    }
}
```

6.8 Sparse Table

```
const int N = 2e5 + 5, K = 20; // K = log2(N)
int a[N], t[N][K];
void build(int n)
{
    for (int i = 1; i <= n; i++) t[i][0] = a[i];
    for (int j = 1; j < K; j++)
    {
        for (int i = 1; i + (1 << j) - 1 <= n; i++)
        {
            t[i][j] = min(t[i][j - 1], t[i + (1 << (j - 1))][
                 j - 1]);
        }
    }
}
int query(int l, int r)
{
    int k = __lg(r - l + 1);
    return min(t[l][k], t[r - (1 << k) + 1][k]);
}
```

7 Formula

```
Others:
Decider for a point located left or right of a line:
d=(x3-x2)*(y2-y1)-(y3-y2)*(x2-x1)

Number of digits: log10(n)+1
Depth of road water: (s^2-h^2)/2h

//sum of series n/1+n/2+n/3+....n/n
ll root=sqrt(n);
for(int i=1; i<=root; i++)
sum+=n/i;
sum=(2*sum)-(root*root);


count the numbers that are
divisible by given number in a
certain range:a=2,b=3,c=7;
low=(a+b-1)/a;
high=c/a;
total=high-low+1;

Euler Constant: 0 .5772156649

#Number of squares in a n*n grid:
S=(n*(n+ 1)*(2*n + 1))/6;

#Number of rectangle in a n*n grid:
R=(n+1)*n/2*(n+1)*n/2 - S;

#Total number of rectangle and square in a n*n grid:
ans=[(n^2 + n)^2]/4

#Number of squares in a n*m grid
exp:6*4
S=6*4+5*3+4*2+3*1=50

#Number of rectangles in n*m grid
R=m(m+1)n(n+1)/4

#Number of cubes in a n*n*n grid
formula:n^k-(n-2)^k
C=n*(n+1)/2*n*(n+1)/2;

#Number of boxes in a n*n*n grid:
B=(n+1)*n/2*(n+1)*n/2*(n+1)*n/2-C;

#Number of hypercube in a n^4grid:
start a loop from 1 to <=n;
HC=0;
for(i=1;i<=n;i++)
HC+=i*i*i*i;

#Number of hyper box in a n^4 grid:
HB=(n+1)*n/2*(n+1)*n/2*(n+1)*n/2*(n+1)*n/2 - HC;
```

8 Geometry

8.1 Circle

```
Distance: sqrt((x2-x1)^2 + (y2-y1)^2)
Check if 3 points are in same line:
x1*(y2-y3)-x2(y1-y3)+x3(y1-y2) = 0
Find a circle that covers 2 given:
x3 = (x1+x2)/2 , y3 = (y1+y2)/2
r = dist(x1,y1,x2,y2)
Lattice Points:1+gcd(|x1-x2|,|y1-y2|)
Slope formed by 2 points:(y2-y1)/(x2-x1)
Area of sector of circle:  r^2*theta
Arc Length: r*thetha
```

8.2 Parallelogram

```
Given 3 points find 4th point:
Dx = Ax + (Cx-Bx)
Dy = Ay + (Cy-By)
Area: | ((Ax*By+Bx*Cy+Cx*Dy+Dx*Ay)
-(Ay*Bx + By*Cx + Cx*Dx + Dy*Ax))|
```

8.3 Polygon

```
The sum of the interior angles: (2n  4)  90
*Area of the largest square inside
a pentagon->
s*(sin(108)/(sin(18)+sin(36)))
Area:(s^2*n)/4*tan(180/4)
Area:(r^2*n*sin(360/n))/2
Area:Apo^2*n*tan(180/n)
Area:  *sq(5*(5+2sq(5)))*s^2
Area:(Apo*s)/2
Area:pr/2
Area:  *n*sin(360/n)*s^2
Area:n*apo^2*tan(180/n)
Area:(n*r*sin(2*(180/n)))/2
Area:( *n*s^2)/tan(180/n)
Perimeter:5*s
Diagonal:(s*(1+sq(5)))/2
Height:(s*sq(5+2sq(5)))/2
an:2*R*sin(180/n) ..here:an=side of
regular inscribed polygon,R=radius
of circumscribed circle.
Sum of interior angles of a
```

```
Convex polygon:180(n-2)
Exterior taken one at each
vertex:360
measurement of Exterior Ang:360/n
Measure Interior An:((n-2)*180)/n
No. Of Dia:(n*(n-3))/2
No. Of Tri:N-2
Side:2*R*sin(180/n)
Apo:R*cos(180/n)
Side:2*apo*tan(180/n)
Area of smallest tri:*apo*(s/2)
=   *Apo^2*tan(180/n)
Intersection points of diagonals of n(odd) sided regular
    polygon = nC4
```

8.4 Right Circular Cone

```
Volume:(pie*h/3)*(R^2+R*r+r^2)
Lateral surface Area:pie(r+R)*S
Area of the base:pie*r^2
Lateral area:pie*r*L
Total Surface A:pie*r^2+pie*r*s
Volume:  *pie*r^2*h
s=sq(r^2+h^2)
```

8.5 Trapezium

```
Area:(a+b)/(a-b) * sqrt((s-a)(s-b) (s-b-c)(s-b-d))
-> s = (a+b+c+d)/2
-> a = long parallel side
-> b = short parallel side
-> c,d = non-parallel side
Area:h*((b1+b2)/2)
H:sq(b^2-(b^2-d^2+(a-c)^2)/2(a-c))^2)
```

8.6 Triangle

```
To form: a+b>c, b+c>a, c+a>b
Check if 3 points form triangle: |(x2-x1)(y3-y1)-(y2-y1)(x3-
    x1)| >0
Perimeter: p = a+b+c
Area:    1) (a*b)/2
         2) (abSinC)/2
         3) sqrt(s(s-a)(s-b)(s-c)); ///s=(p/2)
         4) (sqrt(3)/4)*a*a; ///equi triangle
         5) (b*sqrt(4*a*a-b*b))/4; ///isosceles
Pythagoras: a*a+b*b = c*c
SineRule: a/SinA=b/SinB=c/SinC
CosineRule: CosA = (b*b+c*c-a*a)/2bc
Centre: x=(x1+x2+x3)/3, y=(y1+y2+y3)/3
Median: AD=sqrt((2*b*b+2*c*c-a*a)/4)
Centroid: AG=sqrt(2*b*b-2*c*c-a*a)/3
Inradius: A/s
Circumradius: a/(2*sinA)
r=abc/sq((a+b+c)(a+b-c)(a+c-b)(b+c-a))
```

8.7 Truncated Cone

```
z=(H*r2^2)(r1^2-r2^2)
R=(  *r1^2(z+h))/(H+z)
Volume:  *pie*h*(R^2+(R*r2)+r2^2)
Volume of a cylinder: pi*r*r*h
Volume of a triangular prism: 0.5*b*h*H
```

9 Graph

9.1 DFS on Tree

```
vector<vector<int>> p;
vector<long long> depth;
vector<long long> reverse_depth;
void dfs(int u, int par)
{
    if (p[u].size() == 1 && p[u][0] == par){
        depth[u] = depth[par]+1;
        reverse_depth[u] = 1;
    }
    else
    {
        for (auto v : p[u])
        {
            if (v != par)
            {
                depth[v] = 1 + depth[u];
                dfs(v, u);
                reverse_depth[u] = 1 + reverse_depth[v];
            }
        }
    }
}
void solve(int n)
{
    p.assign(n + 2, vector<int>());
    depth.assign(n + 2, 0);
    reverse_depth.assign(n + 2, 0);
    for (int i = 1; i < n; i++)
    {
        int x, y;
        cin >> x >> y;
        p[x].push_back(y);
        p[y].push_back(x);
    }
    depth[1]=1;
    dfs(1, -1);
    int x = 2;
    cout<<depth[x]<<" "<<reverse_depth[x]<<endl;
}
```

9.2 Detect Cycle in Tree

```
void dfs(int u, vector<bool> &vis, vector<vector<int>>&p,
     int prev)
{
    if(vis[u]){
        cycle = true;
        return;
    }

    vis[u] = true;

    for (int i = 0; i < p[u].size(); i++)
    {
        int v = p[u][i];
        if(prev!=v){
            dfs(v, vis, p, u);
        }
    }
}
```

9.3 Dijkstra

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

const long long N = 2e5 + 3;
const long long inf = 1e18;

vector<pair<long long, long long>> edges[N];
```

```
vector<long long> dist(N, inf);

int main()
{
    long long n, m;
    cin >> n >> m;

    while (m--)
    {
        long long x, y, w;
        cin >> x >> y >> w;
        edges[x].push_back({y, w});
    }

    dist[1] = 0;

    priority_queue<pair<long long, long long>, vector<pair<
         long long, long long>>, greater<pair<long long, long
          long>>> pq;

    pq.push({0, 1});

    while (!pq.empty())
    {
        long long u = pq.top().second, d = pq.top().first;
        pq.pop();
        if (dist[u] < d)
            continue;
        for (auto e : edges[u])
        {
            long long w = e.second, v = e.first;
            if (dist[v] > dist[u] + w)
            {
                dist[v] = dist[u] + w;
                pq.push({dist[v], v});
            }
        }
    }

    for (long long i = 1; i <= n; i++)
        cout << dist[i] << " ";
}
```

9.4 Distance of Leaf from root

```
#include<bits/stdc++.h>
using namespace std;
//K= root, n=node
void find(vector<long long>a[], long long n, long long k)
{
  queue<long long>q;
  long long vis[n+2]={};
  long long dis[n+2]={},maxx=0ll;
  vis[k]=0;
  dis[k]=1;
  q.push(k);
  while(!q.empty())
  {
    long long x=q.front();
    q.pop();
    long long l=a[x].size();
    for(long long i=0;i<l;i++)
    {
      long long y=a[x][i];
      if(!vis[y])
      {
        q.push(y);
        vis[y]=1;
        dis[y]=dis[x]+1;
        maxx=max(maxx,dis[y]);
      }
    }
  }
  cout<<maxx<<endl;
}
```

9.5 Topological Sort

```
void topologicalSortUtil(
    int v,
    vector<vector<int>> &adj,
    vector<bool> &visited,
    stack<int> &Stack)
{
    visited[v] = true;
    for (int i : adj[v])
    {
        if (!visited[i])
            topologicalSortUtil(i, adj, visited, Stack);
    }
    Stack.push(v);
}

void topologicalSort(vector<vector<int>> &adj, int N)
{
    stack<int> Stack;
    vector<bool> visited(N, false);
    for (int i = 0; i < N; i++)
    {
        if (!visited[i])
            topologicalSortUtil(i, adj, visited, Stack);
    }
    while (!Stack.empty())
    {
        cout << Stack.top() << " ";
        Stack.pop();
    }
}
```

10 Math

10.1 String and int multiply

```
string multyply(string a, int b)
{
    int carry = 0, l = a.size();

    string ans = "";

    for (int i = l - 1; i >= 0; i--)
    {
        carry = ((a[i] - '0') * b + carry);
        ans += carry % 10 + '0';
        carry /= 10;
    }
    while (carry != 0)
    {
        ans += carry % 10 + '0';
        carry /= 10;
    }
    reverse(ans.begin(), ans.end());
    return ans;
}
```

10.2 Sum of Absolute Diff of All Pairs

```
int
    sum_of_absolute_differences_of_all_pairs_in_a_given_array
    (int a[], int n)
{
    int ans = 0;
    sort(a, a + n);
    for (long long i = 0; i < n; i++)
        ans += a[i] * (2 * i - n + 1);
```

```
    return ans;
}
```

11 Number Theory

11.1 Big Mul

```
long long bigMul(long long n, long long m, long long p)
{
  if(m<=0) return 0;
    long long res = bigMul(n, m/2, p);
    long long ans = (2*res)%p;
    if(m%2) ans = (ans+n)%p;
    return ans;
}
```

11.2 Bigmod , Inverse MOD, ncr

```
#define MOD 1000000007
long long bigMod(long long a, long long b)
{
  a %= MOD;
  if (!b)
    return 1;
  long long res = bigMod(a, b / 2);
  long long ans = (res * res) % MOD;
  if (b % 2)
    ans = (ans * a) % MOD;
  return ans;
}

long long inverseMod(long long a)
{
  return bigMod(a, MOD - 2);
}


long long fact[MOD];
void factorial()
{
  fact[0] = 1;
  for (long long i = 1; i < MOD; i++)
    fact[i] = (((i % MOD) * (fact[i - 1] % MOD)) % MOD);
}

long long nCr(long long n, long long r)
{
  return ((fact[n] % MOD) * (inverseMod((fact[r] * fact[n -
      r]) % MOD) % MOD)) % MOD;
}
```

11.3 Bigmod with Loop

```
#define MOD 1000000007
long long Big(long long x, long long n)
{
  long long ans=1;
  while(n>0){
    x%=MOD;
    if(n&1) ans*=x;
    ans%=MOD;
    x*=x;
    n>>=1;
  }
  return ans;
}
```

11.4 Euler Phi Sieve

```
// P1k1    1(P1    1).P2k2    1(P2    1)....Prkr - 1(Pr - 1)
void computeTotient(int n)
{
    for (int i = 1; i <= n; i++) phi[i] = i;
    for (int j = 2; j <= n; j++)
    {
        if (phi[j] == j)
        {
            phi[j] = j - 1;
            for (int i = 2 * j; i <= n; i += j)
            {
                phi[i] = (phi[i] / j) * (j - 1);
            }
        }
    }
}
```

11.5 Factorial

11.5.1 Digit Count

```
int factDigitCnt(int n)
{
    if (n <= 1)
        return n;
    double digits = 0;
    for (int i = 2; i <= n; i++)
    {
        digits += log10(i);
    }
    return floor(digits) + 1;
}
```

11.5.2 Divisor Count

```
long long factDivisorsCnt(long long n)
{
    long long res = 1;
    for (int i = 0; primes[i] <= n; i++)
    {
        long long exp = 0;
        long long p = primes[i];
        while (p <= n)
        {
            exp += (n / p);
            p *= primes[i];
        }
        res *= (exp + 1);
    }
    return res;
}
```

11.5.3 Tailing Zeros

```
int trailingZeroes(int n)
{
    int c = 0, f = 5;
    while (f <= n)
    {
        c += n / f;
        f *= 5;
    }
    return c;
}
```

11.6 Generate Number of Divisor 1 to N

```
vector<int>generateNumberOfDivisor(int n= 1e6){
```

```
    vector<int>divisor(n+1, 1);
    for(int i=2;i<=n;i++){
        if(divisor[i]==1){
            for(int j=i;j<=n;j+=i){
                int num = j, primeFactor = 0;
                while(num%i==0){
                    num/=i;
                    primeFactor++;
                }
                divisor[j] *= (primeFactor+1);
            }
        }
    }
    return divisor;
}
```

11.7 Get Prime

```
#define INF 1000005
int prime[INF];
bool vis[INF];

void getPrime()
{
    int k = 1;
    prime[k++] = 2;
    for (long long i = 3; i <INF; i += 2)
    {
        if (!vis[i] && i % 2)
            prime[k++] = i;
        for (long long j = i * i; j < INF; j += i)
        {
            vis[j] = true;
        }
    }
}
```

11.8 Is Prime

```
vector<bool> isPrime(long long n = 1e6)
{
    vector<bool> vis(n + 5);
    vis[1] = true;
    for (long long i = 3; i <= n; i+=2)
    {
        if (!vis[i])
            for (long long j = i * i; j <= n; j += i)
                vis[j] = true;
    }
    return vis;
}
```

11.9 MOD Jog Gun

```
#define MOD 1000000007

long long modGunKoro(long long a, long long b){
    return ((a%MOD)*(b%MOD))%MOD;
}

long long modJogKoro(long long a, long long b){
    return ((a%MOD)+(b%MOD))%MOD;
}
```

11.10 Number of Divisor

```
#define nn 1000010115.
long long int notprime[nn] = {}, prime[nn];
long long numberofDivisor(long long n)
{
    long long int c = 1, i, j, ans = 1;
    for (i = 3; i * i <= nn; i += 2)
    {

        if (!notprime[i])
        {
            for (j = i * i; j <= nn; j += i)
                notprime[j] = 1;
        }
    }
    prime[c++] = 2;
    for (i = 3; i <= nn; i += 2)
    {
        if (!notprime[i])
        {

            prime[c++] = i;
        }
    }

    for (i = 1; i <= nn && prime[i] * prime[i] <= n; i++)
    {
        if (n % prime[i] == 0)
        {
            int cnt = 1;
            while (n > 1 && n % prime[i] == 0)
            {
                n /= prime[i];
                cnt++;
            }
            ans *= cnt;
        }
    }
    if (n != 1)
        ans *= 2;
}
```

11.11 Number of Prime Divisor

```
vector<int>generateNumberOfPrimeDivisor(int n = 1e6){
    vector<int>primeDivisor(n+1, 0);
    for(int i=2;i<=n;i++){
        if(primeDivisor[i]==0){
            for(int j=i;j<=n;j+=i){
                primeDivisor[j] ++;
            }
        }
    }
    return primeDivisor;
}
```

11.12 Sum of Divisor

```
vector<int> generateSumOfDivisor(int n = 2e6)
{
    vector<int> divSum(n + 5, 1);
    divSum[1]=0;
    for (int i = 2; i <= n; i++)
    {
        for(int j=i+i;j<=n;j+=i){
            divSum[j]+=i;
        }
    }
    return divSum;
}
```

12 String

12.1 Hashing

```
const long long MOD = 999999999999999989, MOD1=1000001137;
long long base = 7919,base1= 2551, base2 = 6091;
void pre_power()
{
    pw[0] = 1;
    for(int i = 1; i < 300015; i++)
        pw[i] = (pw[i - 1] * base) % MOD;
}
long long get_hashval(string str)
{
    int len=str.length();
    long long hash_val=0;
    for(int i = 0; i < len; i++)
    {
        hash_val=((hash_val*base)+str[i])%MOD;
        HASH[i+1]=hash_val;
    }
    return hash_val;
}
long long SubstringHash(int l, int r)
{
    return (HASH[r]-(HASH[l-1]*pw[r-l+1]) %
           MOD + MOD) % MOD;
}
```

12.2 KMP

```
vector<int> lps(string pattern)
{
    int n = pattern.size();
    vector<int> v(n);
    int index = 0;

    for (int i = 1; i < n;)
    {
        if (pattern[i] == pattern[index])
        {
            v[i] = index + 1;
            i++;
            index++;
        }
        else
        {
            if (index)
                index = v[index - 1];
            else
            {
                v[i] = 0;
                i++;
            }
        }
    }
    return v;
}

int kmp(string s, string pattern)
{
    int n = s.size(), m = pattern.size();
    int i = 0, j = 0;
    int ans = 0;

    vector<int> v = lps(pattern);

    while (i < n)
    {
        if (s[i] == pattern[j])
        {
            i++;
            j++;
        }
        else
        {
            if (j)
                j = v[j - 1];
            else
                i++;
        }

        if (j == m) //to count how many pattern match
        {
            ans++;
            j = v[j - 1];
        }
    }

    return ans;
}
```

12.3 lcs

```
int lcs(string X, string Y, int m, int n, vector<vector<int
    >> &dp)
{
    if (m == 0 || n == 0)
        return 0;
    if (X[m - 1] == Y[n - 1])
        return dp[m][n] = 1 + lcs(X, Y, m - 1, n - 1, dp);

    if (dp[m][n] != -1)
    {
        return dp[m][n];
    }
    return dp[m][n] = max(lcs(X, Y, m, n - 1, dp), lcs(X, Y,
         m - 1, n, dp));
}
```

13 Trie

```
#include <bits/stdc++.h>
using namespace std;
#define endl "\n"
#define con (f ? "YES" : "NO")
#define loj(i, j) "Case " << i << ": " << j

struct Trie
{
    bool lastLetter;
    Trie *children[10];

    Trie()
    {
        for (int i = 0; i < 10; i++)
        {
            lastLetter = false;
            children[i] = nullptr;
        }
    }
};

void insert(string &s, Trie *root)
{
    int n = s.size();

    for (char c : s)
    {
        int index = c - '0';
        if (root->children[index] == nullptr)
            root->children[index] = new Trie();
        root = root->children[index];
    }
    root->lastLetter = true;
```

```
}

bool isPrefix(Trie *node)
{
    for (int i = 0; i < 10; i++)
    {
        if (node->children[i] != nullptr)
        {
            if (node->lastLetter)
                return true;
            if (isPrefix(node->children[i]))
                return true;
        }
    }
    return false;
}

void clear(Trie *node)
{
    for (int i = 0; i < 10; i++)
    {
        if (node->children[i] != nullptr)
        {
            clear(node->children[i]);
            node->children[i] = nullptr;
        }
    }
    delete (node);
}

int main()
{
    ios_base::sync_with_stdio(false);
    cin.tie(0), cout.tie(0);

    long long t, k = 0;
    cin >> t;
    while (t--)
    {
        long long n;
        cin >> n;

        Trie *root = new Trie();

        while (n--)
        {
            string s;
            cin >> s;
            insert(s, root);
        }

        bool f = !isPrefix(root);
        cout << loj(++k, con) << endl;
        clear(root);
    }
}
```

14 Z Function

```
#include <bits/stdc++.h>
using namespace std;
#define endl  \ n
vector<int> z_function(string s)
{
    int n = (int)s.size();
    vector<int> z(n);
    z[0] = n;
    for (int i = 1, l = 0, r = 0; i < n; ++i)
    {
        if (i <= r)
            z[i] = min(r - i + 1, z[i - l]);
        while (i + z[i] < n && s[z[i]] == s[i + z[i]])
            ++z[i];
        if (i + z[i] - 1 > r)
            l = i, r = i + z[i] - 1;
    }
    return z;
}
int main()
{
    // Simple is BEAST
    int n;
    cin >> n;
    cout << n * n << endl;
    return 0;
}
```

15 templates

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

struct Point
{
    double x, y;
    Point() {}
    Point(double x, double y) : x(x), y(y) {}
    Point(const Point &p) : x(p.x), y(p.y) {}
    void input()
    {
        scanf("%lf%lf", &x, &y);
    }
    Point operator+(const Point &p)
        const
    {
        return Point(x + p.x, y + p.y);
    }
    Point operator-(const Point &p)
        const
    {
        return Point(x - p.x, y - p.y);
    }
    Point operator*(double c) const
    {
        return Point(x * c, y * c);
    }
    Point operator/(double c) const
    {
        return Point(x / c, y / c);
    }
};
vector<Point> polygon;
double getClockwiseAngle(Point p)
{
    return -1 * atan2(p.x, -1 * p.y);
}
// compare function to compare clockwise
bool comparePoints(Point p1, Point p2)
{
    return getClockwiseAngle(p1) < getClockwiseAngle(p2);
}
// rotate 90 degree counter clockwise
Point RotateCCW90(Point p)
{
    return Point(-p.y, p.x);
}
// rotate 90 degree clockwise
Point RotateCW90(Point p)
{
    return Point(p.y, -p.x);
}

Point RotateCCW(Point p, double t)
{
    return Point(p.x * cos(t) - p.y * sin(t),
```

```
            p.x * sin(t) + p.y * cos(t));
}
Point RotateCW(Point p, double t)
{
    return Point(p.x * cos(t) + p.y * sin(t),
                -p.x * sin(t) + p.y * cos(t));
}
double dot(Point A, Point B)
{
    return A.x * B.x + A.y * B.y;
}
double cross(Point A, Point B)
{
    return A.x * B.y - A.y * B.x;
}
double dist2(Point A, Point B)
{
    return dot(A - B, A - B);
}
// returns distance between two point
double dist(Point A, Point B)
{
    return sqrt(dot(A - B, A - B));
}
// Distance between point A and B
double distBetweenPoint(Point A, Point B)
{
    return sqrt(dot(A - B, A - B));
}
// project point c onto line AB (A != B)
Point ProjectPointLine(Point A, Point B, Point C)
{
    return A + (B - A) *
                dot(C - A, B - A) / dot(B - A, B - A);
}
// Determine if Line AB and CD are parallel or collinear
bool LinesParallel(Point A, Point B, Point C, Point D)
{
    return fabs(cross(B - A, C - D)) < EPS;
}
// Determine if Line AB and CD are collinear
bool LinesCollinear(Point A, Point B, Point C, Point D)
{
    return LinesParallel(A, B, C, D) && fabs(cross(A - B, A -
         C)) < EPS && fabs(cross(C - D, C - A)) < EPS;
}

// checks if AB intersect with CD
bool SegmentIntersect(Point A, Point B, Point C, Point D)
{
    if (LinesCollinear(A, B, C, D))
    {
        if (dist2(A, C) < EPS || dist2(A, D) < EPS || dist2(B
             , C) < EPS || dist2(B, D) < EPS)
            return true;
        if (dot(C - A, C - B) > 0 && dot(D - A, D - B) > 0 &&
              dot(C - B, D - B) > 0)
            return false;
        return true;
    }
    if (cross(D - A, B - A) * cross(C - A, B - A) > 0)
        return false;
    if (cross(A - C, D - C) * cross(B - C, D - C) > 0)
        return false;
    return true;
}
// Compute the coordinates where AB and CD intersect
Point ComputeLineIntersection(Point A, Point B, Point C,
     Point D)
{
    double a1, b1, c1, a2, b2, c2;
    a1 = A.y - B.y;
    b1 = B.x - A.x;
    c1 = cross(A, B);
    a2 = C.y - D.y;
    b2 = D.x - C.x;
    c2 = cross(C, D);
    double Dist = a1 * b2 - a2 * b1;
    return Point((b1 * c2 - b2 * c1) / Dist, (c1 * a2 - c2 *
         a1) / Dist);
}
// Project point C onto line segment AB -- return the Point
     from AB which is the closest to C --
Point ProjectPointSegment(Point A, Point B, Point C)
{
    double r = dot(B - A, B - A);
    if (fabs(r) < EPS)
        return A;
    r = dot(C - A, B - A) / r;
    if (r < 0)
        return A;
    if (r > 1)
        return B;
    return A + (B - A) * r;
}
// return the minimum distance from a point C to a line AB
double DistancePointSegment(Point A, Point B, Point C)
{
    return distBetweenPoint(C, ProjectPointSegment(A, B, C));
}


// return distance between P and a
// point where p is perpendicular on
// AB.AB er upore p jei point e lombo
// shei point theke p er distance

double distToLine(Point p, Point a, Point b)
{
    pair<double, double> c;
    double scale = (double)(dot(p - a, b - a)) / (dot(b - a,
         b - a));
    c.first = a.x + scale * (b.x - a.x);
    c.second = a.y + scale * (b.y - a.y);
    double dx = (double)p.x - c.first;
    double dy = (double)p.y - c.second;
    return sqrt(dx * dx + dy * dy);
}
long long orientation(Point p, Point q, Point r)
{
    long long val = (q.y - p.y) * (r.x - q.x) - (q.x - p.x) *
          (r.y - q.y);
    if (val > 0)
        return 1;
    if (val < 0)
        return 2;
    else
        return val;
}

// Given three colinear points p,
// q, r, the function checks if
// point q lies on line segment
// 'pr'

bool onSegment(Point p, Point q, Point r)
{
    if (q.x <= max(p.x, r.x) && q.x >= min(p.x, r.x) && q.y
         <= max(p.y, r.y) && q.y >= min(p.y, r.y))
        return true;
    return false;
}
// checks if Point P is inside of
// polygon or not
bool isInside(int n, Point p)
{
    if (n < 3)
        return false;
    Point extreme = Point(INF, p.y); // here INF=1e4
    int count = 0, i = 0;
    do
```

```
    {
        int next = (i + 1) % n;
        if (SegmentIntersect(polygon[i], polygon[next], p,
             extreme))
        {
            if (orientation(polygon[i], p, polygon[next]) ==
                 0)
                return onSegment(polygon[i], p, polygon[next])
                     ;
            count++;
        }
        i = next;
    } while (i != 0);
    return count & 1;
}
// returns the perimeter of a
// polygon
double polygonPerimeter(int n)
{
    double perimeter = 0.0;
    for (int i = 0; i < n - 1; i++) // polygon vector holds
         the corner points of the given polygon
        perimeter += dist(polygon[i], polygon[i + 1]);
    perimeter += dist(polygon[0], polygon[n - 1]);
    return perimeter;
}
// returns the area of a polygon
double polygonArea(int n)
{
    double area = 0.0;
    int j = n - 1;
    for (int i = 0; i < n; i++)
    {
        area += (polygon[j].x + polygon[i].x) * (polygon[j].y
              - polygon[i].y);
        j = i;
    }
    return fabs(area) * 0.5;
}
double getTriangleArea(Point a, Point b, Point c)
{
    return fabs(cross(b - a, c - a));
}
bool compareConvex(Point X, Point Y)
{
    long long ret =
        orientation(points[0], X, Y);
    if (ret == 0)
    {
        ll dist11 = dist2(points[0], X);
        ll dist22 = dist2(points[0], Y);
        return dist11 < dist22;
    }
    else if (ret == 2)
        return true;
    else
        return false;
}
Point nextToTop(stack<Point> &S)
{
    Point p = S.top();
    S.pop();
    Point res = S.top();
    S.push(p);
    return res;
}

// make a minimum area polygon
stack<Point> convexHull(int N)
{
    int ymin = points[0].y, index = 0;
    for (int i = 1; i < N; i++)
    {
        if (points[i].y < ymin || (points[i].y == ymin &&
             points[i].x < points[index].x))
        {
            ymin = points[i].y;
            index = i;
        }
    }
    stack<Point> S;
    swap(points[0], points[index]);
    sort(&points[1], &points[N], compareConvex);
    S.push(points[0]);
    for (int i = 1; i < N; i++)
    {
        while (S.size() > 1 && orientation(nextToTop(S), S.
             top(), points[i]) != 2)
        {
            S.pop();
        }
        S.push(points[i]);
    }
    return S;
}
// Angle between Line AB and AC in degree
double angle(Point B, Point A, Point C)
{
    double c = dist(A, B);
    double a = dist(B, C);
    double b = dist(A, C);
    double ans = acos((b * b + c * c - a * a) / (2 * b * c));
    return (ans * 180) / acos(-1);
}
// returns number of vertices on boundary of a polygon
ll picks_theorem_boundary_count()
{
    int sz = polygon.size(), i;
    long long res = __gcd((long long)abs(polygon[0].x -
         polygon[sz - 1].x), (long long)abs(polygon[0].y -
         polygon[sz - 1].y));
    for (i = 0; i < sz - 1; i++)
    {
        res += __gcd((long long)abs(polygon[i].x - polygon[i
             + 1].x), (long long)abs(polygon[i].y - polygon[i
              + 1].y));
    }
    return res;
}

// picks theorem
// Polygon area= inside points + boundary points/2 -1
// return inside points counts
ll lattice_points_inside_polygon()
{
    ll ar = polygonArea(n);
    ll b =
        picks_theorem_boundary_count();
    long long tot = ar + 1 - b / 2;
    return tot;
}

// Physics Formuas
// motion
v = u + at
s = ut +(1/2) at*t ,
v*v    u*u = 2*a*s

// Projectile motion
x = utcos
y = utsin    (1/2) gt*t
y = x tan     g*x*x/( 2u*u*cos* cos )
T = 2u sin/g
R = u*u*sin2/g
H = u*u*sin*sin/2g

// others:
p=mv
v*v/r*g = tan(Banking angle)
W = F S cos
```

```
K = ( )mv*v = p*p/ 2m
T = 2* *sqrt(l/g)

// Trigonometry
 sin2 =2  sincos
 cos2 = cos * cos - sin * sin
 sin3 =3 sin -4* sin * sin * sin
 cos3 =4* cos * cos * cos -3 cos
For triangle:
a=bcosC+ccosB
b=acosC+ccosA
c=bcosA+acosB
```

```
sin(A+ B )=sinAcosB +   cosAsinB
cos(A+ B )=cosAcosB   + sinAsinB

// Circle Line intersection
double r, a, b, c; // given as input //ax+by+c=0//EPS=1e-9
double x0 = -a*c/(a*a+b*b), y0 = -b*c/(a*a+b*b);
if (c*c > r*r*(a*a+b*b)+EPS)
    puts ("no points");
else if (abs (c*c - r*r*(a*a+b*b)) < EPS){
puts ("1 point");
cout << x0 << ' ' << y0 << '\n';}
else {
```

```
double d = r*r - c*c/(a*a+b*b);
double mult = sqrt (d / (a*a+b*b));
double ax, ay, bx, by;
ax = x0 + b * mult;
bx = x0 - b * mult;
ay = y0 - a * mult;
by = y0 + a * mult;
puts ("2 points");
cout << ax << ' ' << ay << '\n' << bx << ' ' << by << '\n';}
```