Logo

Description automatically generated with medium confidence

## ANARC05B - The Double HeLiX

[#dynamic-programming](https://www.spoj.com/problems/tag/dynamic-programming) [#binary-search](https://www.spoj.com/problems/tag/binary-search)

Two ﬁnite, strictly increasing, integer sequences are given. Any common integer between the two sequences constitute an intersection point. Take for example the following two sequences where intersection points are  
printed in bold:

* First= 3 5 **7** 9 20 **25** 30 40 **55** 56 **57** 60 62
* Second= 1 4 **7** 11 14 **25** 44 47 **55** **57** 100

You can ‘walk” over these two sequences in the following way:

1. You may start at the beginning of any of the two sequences. Now start moving forward.
2. At each intersection point, you have the choice of either continuing with the same sequence you’re currently on, or switching to the other sequence.

The objective is ﬁnding a path that produces the maximum sum of data you walked over. In the above example, the largest possible sum is 450, which is the result of adding 3, 5, 7, 9, 20, 25, 44, 47, 55, 56, 57, 60, and 62

### **Input**

Your program will be tested on a number of test cases. Each test case will be speciﬁed on two separate lines. Each line denotes a sequence and is speciﬁed using the following format:

n v1 v2 ... vn

Where n is the length of the sequence and vi is the ith element in that sequence. Each sequence will have at least one element but no more than 10,000. All elements are between -10,000 and 10,000 (inclusive).  
The last line of the input includes a single zero, which is not part of the test cases.

### **Output**

For each test case, write on a separate line, the largest possible sum that can be produced.

### **Sample**

**Input:**

13 3 5 7 9 20 25 30 40 55 56 57 60 62

11 1 4 7 11 14 25 44 47 55 57 100

4 -5 100 1000 1005

3 -12 1000 1001

0

**Output:**

450

#include<bits/stdc++.h>

#define ll long long int

#define maxm 1e5+2

#define f(i, in, n) for(long long int i=in; i<n; i++)

#define fl(i, in, n) for(long long int i=n-1; i>=in; i--)

#define lim(i, a, b) for (long long int i=a; i<b; i++)

using namespace std;

}

int main() {

    ios\_base::sync\_with\_stdio(false);

    cin.tie(NULL);

    /\*int q;

    cin>>q;

    while (q--) {

    }\*/

    while (1) {

        ll n1, n2;

        cin>>n1;

        if (n1==0) break;

        vector<ll> a(n1);

        f(i, 0, n1) cin>>a[i];

        cin>>n2;

        vector<ll> b(n2);

        f(i, 0, n2) cin>>b[i];

        ll i=0, j=0, sum1=0, sum2=0, ans=0;

        while (i<n1 and j<n2) {

            if (a[i]==b[j]) {

                ans+=max(sum1, sum2);

                sum1=a[i]; sum2=b[j];

                i++; j++;

            }

            else if (a[i]<b[j]) {

                sum1+=a[i];

                i++;

            }

            else {

                sum2+=b[j];

                j++;

            }

        }

        while (i<n1) {

            sum1+=a[i];

            i++;

        }

        while (j<n2) {

            sum2+=b[j];

            j++;

        }

        ans+=max(sum1, sum2);

        cout<<ans<<endl;

    }

    return 0;

}

## PRATA - Roti Prata

*no tags*

IEEE is having its AGM next week and the president wants to serve cheese prata after the meeting. The subcommittee members are asked to go to food connection and get P(P<=1000) pratas packed for the function. The stall has L cooks(L<=50) and each cook has a rank R(1<=R<=8). A cook with a rank R can cook 1 prata in the first R minutes 1 more prata in the next 2R minutes, 1 more prata in 3R minutes and so on(he can only cook a complete prata) ( For example if a cook is ranked 2.. he will cook one prata in 2 minutes one more prata in the next 4 mins an one more in the next 6 minutes hence in total 12 minutes he cooks 3 pratas in 13 minutes also he can cook only 3 pratas as he does not have enough time for the 4th prata). The webmaster wants to know the minimum time to get the order done. Please write a program to help him out.

### **Input**

The first line tells the number of test cases. Each test case consist of 2 lines. In the first line of the test case we have P the number of prata ordered. In the next line the first integer denotes the number of cooks L and L integers follow in the same line each denoting the rank of a cook.

### **Output**

Print an integer which tells the number of minutes needed to get the order done.

### **Example**

**Input:**  
  
3  
10  
4 1 2 3 4  
8  
1 1  
8  
8 1 1 1 1 1 1 1 1  
**Output:**  
12  
36  
1

#include<bits/stdc++.h>

#define ll long long int

#define maxm 1e5+2

#define f(i, in, n) for(long long int i=in; i<n; i++)

#define fl(i, in, n) for(long long int i=n-1; i>=in; i--)

#define lim(i, a, b) for (long long int i=a; i<b; i++)

using namespace std;

vector<ll> xor\_of(300001);

vector<ll> seg\_tree(maxm);

// Building a Segment Tree

ll buildSegmentTree(vector<ll> v, ll i, ll l, ll r) {

    if (l==r) {

        seg\_tree[i]=v[l];

        return v[l];

    }

    ll mid=(l+r)/2;

    seg\_tree[i]=buildSegmentTree(v, 2\*i+1, l, mid)+buildSegmentTree(v, 2\*i+2, mid+1, r);

    return seg\_tree[i];

}

// To get sum from Range Query

ll getSum(ll index, ll st, ll en, ll l, ll r) {

    if (st>r or en<l) return 0;

    if (l<=st and en<=r) return seg\_tree[index];

    ll mid=(st+en)/2;

    return getSum(2\*index+1, st, mid, l, r)+getSum(2\*index+2, mid+1, en, l, r);

}

// Update Queries

void updateRange(ll index, ll pos, ll diff, ll st, ll en) {

    if (st>pos or en<pos) return;

    seg\_tree[index]+=diff;

    if (st!=en) {

        ll mid=(st+en)/2;

        updateRange(2\*index+1, pos, diff, st, mid);

        updateRange(2\*index+2, pos, diff, mid+1, en);

    }

}

bool check(vector<ll> v, ll p, ll t) {

    ll count=0;

    f(i, 0, v.size()) {

        ll val=(2\*t)/v[i];

        ll x=sqrt(val);

        if (x\*(x+1)<=val) count+=x;

        else count+=(x-1);

    }

    return count>=p;

}

int main() {

    ios\_base::sync\_with\_stdio(false);

    cin.tie(NULL);

    int q;

    cin>>q;

    while (q--) {

        ll p;

        cin>>p;

        ll n; cin>>n;

        vector<ll> v(n);

        f(i, 0, n) cin>>v[i];

        ll maximum=\*max\_element(v.begin(), v.end());

        ll l=0, r=1e7;

        while (r-l>1) {

            ll mid=(l+r)/2;

            if (check(v, p, mid)) r=mid;

            else l=mid+1;

        }

        if (check(v, p, l)) cout<<l<<endl;

        else cout<<r<<endl;

    }

    return 0;

}