**Algorithmic Problem Solving [17ECSE309]**

**Q-Box Assignment Set**

Student Name: P.Shri Aakash

SRN:01FE19BAR005

Branch: A&R

**Question 01**

Title: Permutations

Level: Easy

Concepts Tested: Arrays

**Problem Statement:**

Given an integer N. Find out the PSum where PSum for integer N is defined as the maximum sum of difference of adjacent elements in all arrangement of numbers from 1 to N.

Note: Difference between elements A and B indicates the absolute difference between the elements.

i.e A-B-|A-B|

**Input Format:**

First line of input contains number of test case T. Each test case contains a single integer N.

**Constraints:**

1<=T<=10000  
1<=N<=105

**Output Format:**

For each test case print the maximum value of PSum.

**Solution:**

#include<bits/stdc++.h>

#define pb push\_back

#define ll long long

using namespace std;

ll ar[100001]={0};

void permutationsum(){

ar[0]=0;

ar[1]=1;

ar[2]=1;

for(ll i=3;i<100001;i++){

ar[i]=ar[i-1]-(i%2)+i;

}

}

int main(){

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

cin.tie(NULL);

ll t;

cin>>t;

permutationsum();

while(t--){

ll n;

cin>>n;

cout<<ar[n]<<"\n";

}

}

**Sample Test Cases:**

Sample Input 0:

3

1

2

3

Sample Output 0:

1

1

3

**Test Cases:**

https://github.com/Shri-Aakash/Q-Box/tree/main/permutation%20again

**Question 02**

Title:

Level: Medium

Concepts Tested: Dynamic Programming, Number theory

**Problem Statement:**

Scooby the dog likes sets which are closed. Let us lay down a few definitions first.

**Definition 1:** A set S is said to closed with respect to a prime number P if and only if:

a∗b (modP)∈S∀a,b∈S

Note that a,b can be equal.

As an example, the set S={2,4,1} is closed with respect to prime P=7.

**Definition 2:** Closure(S): Closure of a set S is defined to be the smallest closed set containing the set S.

As an example, for P=5 and set S={3}, Closure(S)={3,4,2,1}

**Definition 3:** Partition(S): Partition of a set S is a set of non-empty subsets of S such that every element of S is in exactly one of these subsets. Moreover, the length of the partition is the number of non-empty subsets required.

As an example, for S={1,3,7,4,6,2}, T={{1,4,6},{3,7,2}} is a partition of set S of length 2 as there are two non-empty subsets and each element of S is included in exactly one subset. Also set S itself is a partition of S of length 1.

Now the task is that you are given a prime number P and an array A of N integers , in which the ith integer is denoted by Ai, 1≤i≤N, 1≤Ai<P.

You have to partition the set of indices {1,2,3,4,...,N} into a partition of **minimum** length such that if Closure({Ai})⊆Closure({Aj}) then i and j must be in different sets of the partition.

Here, i≠j , 1≤i,j≤N.

**Note**: x denotes a set containing a single element x.

**Input Format:**

The first line will consist of the integer T denoting the number of test cases.

For each of the T test cases, the first line will consist of two integers P and N, where P is a prime number and N denotes the number of elements.

Next line will consist of N integers, where the ith integer denotes the element Ai, 1≤i≤N.

**Constraints:**

1<=T<=100

 1<=P<=10^9,P is guaranteed to be prime.

1≤N≤10^3

**Output Format:**

For each of the *T* test cases, output a single integer denoting the minimum length of the partition.

**Solution:**

#include <bits/stdc++.h>

using namespace std;

#define ll long long

int A[1011];

int dp[1011];

ll bpow(ll x,ll n, ll mod) {

ll ans = 1;

while(n>0) {

if(n&1) ans\*=x;

x\*=x;

ans%=mod;

x%=mod;

n/=2;

}

return ans;

}

int main()

{

int T;

cin >> T;

while(T--) {

int P,N;

cin >> P >> N;

vector<int>divs;

int sz = sqrt(P-1);

int num = P-1;

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

if(num%j==0) divs.push\_back(j), divs.push\_back(num/j);

}

sort(divs.begin(),divs.end());

vector<int>periods;

for(int i=0;i<N;i++) {

assert(cin >> A[i]);

assert(A[i]>=1 and A[i]<P);

ll cur = 1;

for(auto d:divs) {

if(bpow(A[i],d,P)==1){

periods.push\_back(d);

break;

}

}

}

int maxLen = 0;

sort(periods.begin(),periods.end());

for(int i=N-1;i>=0;i--) {

dp[i] = 1;

for(int j=i+1;j<N;j++) {

if(periods[j] % periods[i] == 0) dp[i] = max(dp[i], dp[j]+1);

}

maxLen = max(maxLen, dp[i]);

}

cout << maxLen << "\n";

}

}

**Sample Test Cases:**

Sample Input 0:

2

2 1

1

3 2

1 2

Sample Output 1:

1

2

**Test Cases:**

https://github.com/Shri-Aakash/Q-Box