Zadaci sa JBOI

1. Članovi Naučnog-Nastavnog Veća (NNV) fakulteta F zaista uživaju u diskutovanju problema tokom sednica u sali 706. Tada svi NNV članovi sede za stolom (dugačak prvaougaoni sto gde svi članovi NNV-a sede jedan do drugog uz jednu od dužih strana stola). Članovi NNV-a su izbirljivi ljudi – oni su zadovoljni svojom pozicijom za stolom samo kad sede između dvoje ljudi i oba njihova suseda su ili strogo viša od njih ili su strogo niža od njih. Interesantno, nikoja dva člana odbora sa jednakim visinama nisu postavljeni da sede jedan do drugog.

Pomozite Slaviši, osobi koja je zadužena za postavljanje članova NNV-a za stolom, i nađite najveći broj članova NNV-a koji sede jedan do drugog i svi su zadovoljni svojom pozicijom za stolom.

Ulaz

Prva linija ulaza sadrži ceo broj N ($5 \leq N \leq 50000$) – broj članova NNV-a. Druga linija ulaza sadrži N pozitivnih celih brojeva Hi ($100 \leq Hi \leq 1000000$) – visine svih članova NNV-a, u redosledu sedenja za stolom sleva nadesno.

Izlaz

Vaš program treba da ispiše tačno jedan ceo broj – postojeći maksimalni broj zadovoljnih članova NNV-a koji sede jedan do drugog.

Ograničenja

Vremensko ograničenje: 1 sekunda

Memorijsko ograničenja: 64 megabajta

Testiranje

U barem 30% test slučajeva, N će biti manji od 100 ($5 \leq N < 100$).

Primer 1

Ulaz	izlaz
5 170 172 169 173 150	3

Primer 2

Ulaz	izlaz
6 160 165 170 175 180 185	0

2. Strari veleposednik Jagodić poseduje zemljište koje želi da ostavi u nasledstvo svojim sinovima nakon smrti. Naime, stari Jagodić ima *n* sinova, $1 \leq n \leq 8$. Zemljište je ograničeno horizontalnom duži AB, vertikalnim dužima AP_1 , BP_m, i poligonalnom linijom P, $P=[P_1P_2...P_m]$, $1 \leq m \leq 500$. Može da se pretpostavi da je duž AB smeštena na osi Ox i da je poligonalna linija P smeštena iznad ose Ox. Stari Jagodić mora da napravi n-1 vertikalnih ograda tako da svaka povezuje duž AB sa P. Dakle, svaki sin će u nasledstvo dobiti neke od tako kreiranih *n* parcela i to pod sledećim uslovima:

1. Svaki sin će dobiti parcelu čija je površina proporcionalna njegovim godinama.
2. Ukupna dužina svih ograda mora biti što je moguće manja.

Kreirati C program koji iz prve linije standardnog ulaza učitava dva cela broja n, m, a iz druge linije učitava n celih brojeva v1,v2,...,vn koji predstavljaju godine sinova, $1 \leq v_i \leq 50$. Potom se sa standardnog ulaza učitava m linija od kojih svaka sadrži cele brojeve x_i, y_i koji

predstavljaju koordinate tačaka P_i, $0 \le x_1 < x_2 < ... < x_m \le 32000$, $1 \le y_1,y_2,...,y_m \le 32000$. Brojevi u svakoj liniji su razdvojeni jednim blanko karakterom.
Program treba da na standardni izlaz odštampa rezultat u formi dve linije. U prvoj liniji će se nalaziti realan broj koji predstavlja sumu dužina svih ograda, a u drugoj liniji će se nalaziti n-1 celih brojeva u rastućem poretku i razdvojeni sa jednim blankom tako da k-ti broj (k=1,2,…,n-1) predstavlja koordinatu k-te ograde na Ox osi.
Procenite vremensku složenost ukupnog rešenja.

Napomene:
1. Ignorišite širinu svake ogradice.
2. Ukupna vremnska složenost ne bi smela da premaši *O (n !)*

3. Zeka mora da pređe *n* metara skačući i to skokovima dužine 3, 2 ili 1 metar. Na koliko načina to može da uradi zeka, ako dužine uspešnih skokova formiraju neopadajući niz?

Napišite program **jumps**, koji računa broj načina.

Ulaz

Vrednost *n* se unosi sa standardnog ulaza ($1 \le n \le 10^9$).

Izlaz

Program treba da na standardni izlaz ispiše jedan ceo broj jednak ostatku pri deljenju sa 1000000.

Napomena: U 50% test slučajeva biće $n \le 10^5$.

Test primer

Input

6

Output

7

Objašnjenje: Broj različitih načina je 7, i njegov ostatak pri deljenju sa 7 je 7. Nizovi skokova su:

1) 3+3
2) 3+2+1
3) 3+1+1+1
4) 2+2+2
5) 2+2+1+1
6) 2+1+1+1+1
7) 1+1+1+1+1+1

Rešenja:

1.

```
#include <stdio.h>
#include <stdlib.h>

int main()
{
```

```
        int broj_clanova;
        int i;
        /* Promenljiva u koju se ucitava visina tekuceg clana NNV-a sa standardnog ulaza */
        int visina;
        /* Promenljive u kojima se cuva tekuci i maximalni broj clanova NNV-a koji sede jedan do drugog i zadovoljni su svojom pozicijom */
        int maximalni_podniz = 0;
        int trenutni_podniz = 0;

        //printf("Unesite broj clanova NNV-a koji sede za stolom:\n");
        scanf("%d",&broj_clanova);

        //if(broj_clanova < 5)
        //  printf("Zao nam je, minimalan broj clanova NNV-a mora da bude minimum 5.\n");
        //else if(broj_clanova > 50000)
        //  printf("Zao nam je, maksimalan broj clanova NNV-a moze da bude 50000.\n");
        //else
          //{
            int clanovi[broj_clanova];
            //printf("Unesite visine clanova NNV-a koji sede jedan do drugog s leva na desno za //stolom: \n");
            for(i = 0; i < broj_clanova; i++)
       {
        scanf("%d",&visina);
        //if(visina < 100 || visina > 1000000)
         // {
         //   printf("Visina clana NNV-a moze biti od 100 do 1000000. Morate ponovo uneti visinu //za clana sa visinom %d, a zatim visine za clanove koji sede desno od njega.\n", visina);
         //   i--;
         // }
        //else if(i > 0 && visina == clanovi[i-1])
         // {
         //     printf("Dva clana NNV-a koji su iste visine ne mogu sedeti jedan pored drugog. //Morate staviti nekog drugog clana da sedi sa desne strane clana koji ima visinu %d, a zatim i //visine clanova koji sede desno od novounesenog.\n", clanovi[i-1]);
         //   i--;
         // }
        //else
          clanovi[i] = visina;
       }

          for(i = 1; i < broj_clanova - 1; i++)
       {
        if(clanovi[i] < clanovi[i-1] && clanovi[i] < clanovi[i+1])
          {
```

```
      trenutni_podniz++;
    }
  else if(clanovi[i] > clanovi[i-1] && clanovi[i] > clanovi[i+1])
    {
      trenutni_podniz++;
    }
  else
    {
      if(trenutni_podniz > maximalni_podniz)
            {
              maximalni_podniz = trenutni_podniz;
            }

      trenutni_podniz = 0;
    }

  if(i==broj_clanova-2)
    {

      if(trenutni_podniz > maximalni_podniz)
            {
              maximalni_podniz = trenutni_podniz;
            }
    }
}

      printf("%d", maximalni_podniz);

    //}
  return 0;
}
```

2.

```
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
typedef double tacka[2];
int n = 0, m = 0, *povrsineParcela, sumaGodina = 0;
double *rasporedOgrada, *ogradeX, *ogradeY, najmanjaDuzinaOgrada = -1;
tacka *poligon;
double povrsinaTrapeza(tacka tacka1, tacka tacka2)
{
return (tacka1[1] + tacka2[1])/2 * (tacka2[0] - tacka1[0]);
}
double pPoligona(tacka *poligon)
```

```
{
double p = 0.0;
int i;
for (i = 0; i < m - 1; i++)
{
p += povrsinaTrapeza(poligon[i], poligon[i+1]);
}
return p;
}
void postaviJedinicneOgrade(int brojOgradica)
{
int i, j = 0, k;
tacka *pomocniPoligon = (tacka*)malloc(sizeof(tacka) * m);
for (k = 0; k < m; k++)
{
pomocniPoligon[k][0] = poligon[k][0];
pomocniPoligon[k][1] = poligon[k][1];
}
double p = pPoligona(poligon) / (brojOgradica+1);
ogradeX = (double*)malloc(sizeof(double) * brojOgradica);
ogradeY = (double*)malloc(sizeof(double) * brojOgradica);
for (i = 0; i < brojOgradica; i++)
{
double tempP = 0;
while (tempP < p)
{
tempP += povrsinaTrapeza(pomocniPoligon[j], pomocniPoligon[j+1]);
j++;
}
if (tempP == p)
{
ogradeX[i] = pomocniPoligon[j-1][0];
}
else
{
j--;
double deltaP = p - (tempP - povrsinaTrapeza(pomocniPoligon[j], pomocniPoligon[j+1]));
double k = (pomocniPoligon[j+1][1] - pomocniPoligon[j][1]) / (pomocniPoligon[j+1][0] - pomocniPoligon[j][0]);
double n = poligon[j+1][1] - k * pomocniPoligon[j+1][0];
double c = - deltaP - (k * pomocniPoligon[j][0] * pomocniPoligon[j][0] / 2 + n * pomocniPoligon[j][0]);
double x1, x2;
x1 = (- n + sqrt(n * n - 2 * k *c)) / k;
x2 = (- n + sqrt(n * n + 2 * k *c)) / k;
if (x1 > pomocniPoligon[j][0])
{
ogradeX[i] = x1;
```

```
ogradeY[i] = k * x1 + n;
pomocniPoligon[j][0] = x1;
pomocniPoligon[j][1] = k * x1 + n;
}
else
{
ogradeX[i] = x2;
ogradeY[i] = k * x2 + n;
pomocniPoligon[j][0] = x2;
pomocniPoligon[j][1] = k * x2 + n;
}
//printf("Ograde %d x: %f y: %f \n", i, ogradeX[i], ogradeY[i]);
}
}
}
double postaviOgrade(int *sinovi, double *ograde)
{
int i, trenutnaOgrada = 0;
double duzinaOgrada = 0;
for (i = 0; i < n - 1; i++)
{
trenutnaOgrada += povrsineParcela[sinovi[i]];
ograde[i] = ogradeX[trenutnaOgrada - 1];
duzinaOgrada += ogradeY[trenutnaOgrada - 1];
}
return duzinaOgrada;
}
void zameni(int *broj1, int *broj2)
{
int pomocni;
pomocni = *broj1;
*broj1 = *broj2;
*broj2 = pomocni;
}
void permutacijaSinova(int *sinovi, int i, int duzina)
{
if (i == duzina)
{
int j;
double *trenutneOgrade = (double*)malloc(sizeof(double)*(n - 1));
double duzinaOgrada = postaviOgrade(sinovi, trenutneOgrade);
if (duzinaOgrada < najmanjaDuzinaOgrada || najmanjaDuzinaOgrada < 0)
{
najmanjaDuzinaOgrada = duzinaOgrada;
rasporedOgrada = trenutneOgrade;
}
}
else
```

```
{
int j = i;
for (j = i; j < duzina; j++)
{
zameni(sinovi + i, sinovi + j);
permutacijaSinova(sinovi, i+1, duzina);
zameni(sinovi + i, sinovi + j);
}
}
}
int nadjiNajboljeOgrade()
{
int i, *sinovi;
sinovi = (int*)malloc(sizeof(int) * n);
for (i = 0; i < n; i++)
{
sinovi[i] = i;
}
postaviJedinicneOgrade(sumaGodina - 1);
permutacijaSinova(sinovi, 0, n);
}
int main(int argc, char *argv[])
{
int i, j;
//for (i = 1; i <= 20; i++)
//{
//char fileName[255];
//sprintf(fileName, "testPrimeri/%d.in", i);
//FILE *fin = fopen(fileName, "r");
//sprintf(fileName, "testPrimeri/%d.out", i);
//FILE *fout = fopen(fileName, "w");
scanf("%d%d", &n, &m);
povrsineParcela = (int*)malloc(sizeof(int) * n);
poligon = (tacka*)malloc(sizeof(tacka) * m);
sumaGodina = 0;
for (j = 0; j < n - 1; j++)
{
scanf("%d", &povrsineParcela[j]);
sumaGodina += povrsineParcela[j];
}
scanf("%d\n", &povrsineParcela[j]);
sumaGodina += povrsineParcela[j];
for (j = 0; j < m; j++)
{
// char c[255];
int x, y;
//fgets(c, 255, fin);
scanf("%d%d", &x, &y);
```

```
poligon[j][0] = x;
poligon[j][1] = y;
}
najmanjaDuzinaOgrada = -1;
nadjiNajboljeOgrade();
printf("%f\n", najmanjaDuzinaOgrada);
for (j = 0; j < n - 1; j++)
{
printf("%f\n", rasporedOgrada[j]);
}
// fclose(fin);
// fclose(fout);
//}
return 0;
}
```

3.

n	1	7	13	19	25
cnt	1	8 = 1 + 7	21 = 8 + 13	40 = 21 + 19	65 = 40 + 25

$$S = 1 + 7 + 13 + 19 + 25$$
$$S = 25 + 19 + 13 + 7 + 1$$

$$2S = 26 + 26 + 26 + 26 + 26 = 26 * 5$$
$$S = 26 * 5 / 2 = 65.$$

```
long long cnt = ((n%6 + n)/2%M * ((1 + n/6)%M));
if(n%6==0) cnt++;
cout << cnt%M << endl;
```

```
#include <cstdio>
typedef long long ll;
const ll MOD = 1000000;
int main() {
   int n;
   scanf ("%d",&n);
   printf ("%d\n",(int)(((((((n / 6) % MOD) * ((n / 6 + 1) % MOD)) % MOD) * 3LL) % MOD + (ll)(n % 6 == 0) + ((((ll)(n % 6) % MOD) * (ll)((n / 6 + 1) % MOD)) % MOD)) % MOD));

   return 0;
}
```