20. veljače 2013.

Infokup 2013

Županijsko natjecanje / Osnovna škola (5. i 6. i 7. i 8. razred)

Algoritmi (Basic/Pascal/C/C++)

OBJAŠNJENJA ZADATAKA

Agencija za odgoj i obrazovanje
Education and Teacher Training Agency

MINISTARSTVO ZNANOSTI, OBRAZOVANJA
I ŠPORTA REPUBLIKE HRVATSKE

5.1. Zadatak: 200 Autor: Nikola Dmitrović[1]

Za točno i potpuno rješenje ovog zadatka treba provjeriti da li se među tri zadana broja nalazi barem jedna vrijednost 200. To je moguće postići upotrebom višestruke naredbe odlučivanja pri čemu treba paziti na situaciju kada među brojevima nema spomenute vrijednosti. Pogreška koja se može dogoditi pri ovakvom načinu rješavanja je da se prilikom ispisa traženi odgovor ispiši više puta.

Pseudokod rješenja:

```
učitaj(p,d,g);
ako je p=200 tada
      ispiši('DA')
inače
      ako je d=200 tada
            ispiši ('DA')
      inače
            ako je t=200 tada
                  ispiši ('DA')
            inače
                  ispiši ('NE');
```

Drugi način uključuje naredbu ponavljanja i prebrojavanje broja pojavljivanja vrijednosti 200 među zadanim brojevim. Ako je ta vrijednost jednaka nula, tada je odgovor 'NE', inače je 'DA'.

Pseudokod rješenja:

```
koliko:=0;
za i:=1 do 3 radi
{
      učitaj(broj);
      ako je broj=200 tada
            koliko=koliko+1;
}
ako je koliko=0 tada
      ispiši('NE')
inače
      ispiši('DA')
```

Potrebno znanje: naredba učitavanja i ispisivanja, višestruka naredba odlučivanja

Kategorija: ad hoc

[1] član Državnog povjerenstva, voditelj natjecanja u kategoriji primjena algoritama (programski jezik Basic/Pascal/C/C++), profesor informatike u XV. gimnaziji, Zagreb

5.2. Zadatak: SMS — Autor: Nikola Dmitrović

Zadatak se može rješiti promatranjem svakog od 26 slučajeva koji se mogu pojaviti u ulaznim podacima. To znači da bi trebalo otkriti koliko je puta potrebno pritisnuti tipku da bi se upisalo slovo 'A', koliko je puta potrebno pritisnuti tipku da bi se upisalo slovo 'B' i tako sve do broja pritiskanja tipki za slovo 'Z'. Naravno, kako će ukupno biti N takvih slova, tada je potrebno pratiti i zbrajati vrijednosti pritisaka za svako od N slova.

Pseudokod rješenja:

```
učitaj(n);
ukupno:=0;
za i:=1 do n radi
{
      učitaj(c);
      ako je (c='A') tada ukupno:=ukupno+1;
      ako je (c='B') tada ukupno:=ukupno+2;
      ako je (c='C') tada ukupno:=ukupno+3;
      ...
      ako je (c='W') tada ukupno:=ukupno+1;
      ako je (c='X') tada ukupno:=ukupno+2;
      ako je (c='Y') tada ukupno:=ukupno+3;
      ako je (c='Z') tada ukupno:=ukupno+4;
};
ispiši(ukupno);
```

Uočimo da se svih 26 znakova koji mogu doći na ulazu može podijeliti u 4 grupe, ovisno o tome koliko je puta potrebno pritisnuti odgovarajuću tipku za taj znak. Tako nam npr. treba jedan pritisak tipke da bi upisali slova 'A', 'D', 'G', 'J', 'M', 'P', 'T', 'W'. Ovim načinom možemo smanjiti veličinu našeg koda.

Pseudokod rješenja:

```
učitaj(n);
ukupno:=0;
za i:=1 do n radi
{
      učitaj(c);
      ako je (c='A') ili (c='D') ili (c='G') ili (c='J') ili (c='M') ili
      (c='P') ili (c='T') ili (c='W') tada ukupno:=ukupno+1;
      ako je (c='B') ili (c='E') ili (c='H') ili (c='K') ili (c='N') ili
      (c='Q') ili (c='U') ili (c='X') tada ukupno:=ukupno+2;
      ako je (c='C') ili (c='F') ili (c='I') ili (c='L') ili (c='O') ili
      (c='R') ili (c='V') ili (c='Y') tada ukupno:=ukupno+3;
      ako je (c='S') ili (c='Z') tada ukupno:=ukupno+4;
};
ispiši(ukupno);
```

Jedno alternativno i malo neobično rješenje uključuje nizove. Naime, kreiramo niz od 26 komponenti, te na prvoj poziciji u tom nizu upišemo vrijednost koja odgovara broju pritisaka tipki za slovo 'A' (1),

na drugoj poziciji upišemo vrijednost za slovo 'B' (2) i tako sve do posljednje komponente na koju upišemo vrijednost za slovo 'Z' (4). Kada učitamo znak s ulaza, odgovarajućom naredbom možemo saznati njegov odgovarajući ASCII kod od koga trebamo oduzeti 64 i tako dobiti poziciju u nizu koja odgovara tom znaku. Npr. asc('A')-64=65-64=1. Direktnim pristupom vrijednosti na odgovarajućoj poziciji dobijamo odgovarajući broj pritisaka.

Potrebno znanje: naredba učitavanja i ispisivanja, naredba odlučivanja, naredba ponavljanja

Kategorija: ad hoc

5.3. Zadatak: APP — Autor: Nikola Dmitrović

Ovo je tipičan zadatak koji u svom rješenju traži veliku sistematičnost i pažljivo promatranje svih slučajeva. Konkretno, imamo 8 slučajeva koje trebamo pažljivo proučiti i precizno napisati u odabranom programskom jeziku. Test primjeri u zadatku su tako složeni da netočno rješenje nekog od 8 slučajeva nije značilo gubitak svih bodova.

Proučimo rješenje koje pojedinačno promatra svaki slučaj i spaja ih u jednu cjelinu. Ima više rješenja ovog zadatka, posebno ako se poznaju tipovi podataka niz i string. Međutim, mi ćemo se ovdje fokusirati na rješenje koje uključuje rad sa znamenkama prirodnog broja.

Pseudokod rješenja:

```
učitaj(x);
učitaj (n);
// %-ostatak pri djeljenju, div-operator cjelobrojnig djeljenja
za i:=1 do n radi
     {
     učitaj (izbor);

     ako je izbor=-1 tada // prvi slučaj: komet '-1'
     {
          x:=x div 10; // iz trenutnog broja uklonimo jedinicu
     };
     ako je izbor=-2 tada // prvi slučaj: komet '-2'
     {
          x:=(x div 100)*10+x % 10; // izbacimo deseticu iz broja
     };
     ako je izbor=-3 tada
     {
          x:=(x div 1000)*100+x % 100;
     };
     ako je izbor=-4 tada
     {
          x:=x % 1000;
     };
```

```
        ako je izbor=1 tada
        {
             ako je x % 10<>9 tada
                x:=x+1
             inače
                 x:=x-x % 10;
        };
        ako je izbor=2 tada
        {
             ld:=x div 10;
             dd:=x % 10;
             ako je ld % 10<>9 tada
                ld:=ld+1
             inače
                 ld:=ld-ld % 10;
             x:=ld*10+dd;
        };
        ako je izbor=3 tada
        {
             ld:=x div 100;
             dd:=x % 100;
             ako je ld % 10<>9 tada
                ld:=ld+1
             inače
                 ld:=ld-ld % 10;
             x:=ld*100+dd;
        };
        ako je izbor=4 tada
        {
             ld:=x div 1000;
             dd:=x % 1000;
             ako je ld % 10<>9 tada
                ld:=ld+1
             inače
                 ld:=ld-ld % 10;
             x:=ld*1000+dd;
        };
};
ispiši(x);
```

Potrebno znanje: složena primjena naredbe odlučivanja

Kategorija: ad hoc

6.1. Zadatak: Kup — Autor: Nikola Dmitrović

Prvi dio zadatka je jednostavan i svodi se na zbrajanje učitanih brojeva. Za potpuno rješenje zadatka, dovoljno je samo pažljivo implementirati navedene uvjete.

Pseudokod rješenja:

```
učitaj(jd,sg); učitaj (sd,jg);

junak:=jd+jg;
segesta:=sg+sd;
ispiši(junak,' ', segesta);

ako je junak>segesta tada
   ispiši ('Junak')
inače
    ako je segesta>junak tada
       ispiši ('Segesta')
    inače
        ako je jg>sg tada
           ispiši ('Junak')
        inače
            ako je sg>jg tada
               ispiši ('Segesta')
            inače
                ispiši ('Jedanaesterci');
```

Potrebno znanje: naredba učitavanja i ispisivanja, naredba odlučivanja

Kategorija: ad hoc

6.2. Zadatak: Poruka — Autor: Nikola Dmitrović

Rješenje zadatka SMS je rješenje dijela u kome ne postoje dodatna čekanja između pritisaka tipki. Za cjelokupno rješenje treba pronaći način kako prepoznati da se trenutni znak koji tipkamo i znak koji smo utipkali netom prije nalaze na istoj tipki. To možemo postići definiranjem svih takvih situacija koje su nam zanimljive (npr. ako tipkamo B iza A, tada dodaj jednu sekundu).

Postoje i jednostavnija rješenja koja uključuje nizove, a ovdje ćemo pokazati jedno koje koristi stringove. Definirajmo string „tipkovnica“ koji će pored svih velikih slova engleske abecede imati i po tri zvjezdice između blokova slova koji su napisani na istoj tipki. Sada će se jedna sekunda dodavati ako je pozicija u definiranom stringu trenutnog slova iz poruke od prethodnog slova iz poruka udaljena za manje ili jednako od 3.

Pseudokod rješenja:

```
tipkovnica:='***ABC***DEF***GHI***JKL***MNO***PQRS***TUV***WXYZ';
učitaj(n);
učitaj(poruka); poruka:=poruka+' '; // dodam prazninu kako ne bih morao
                                    // posebno gledati prvo slovo

ukupno:=0;
za i:=1 do n radi
{
     c:=poruka[i];

      ako je (c='A') ili (c='D') ili (c='G') ili (c='J') ili (c='M') ili
      (c='P') ili (c='T') ili (c='W') tada ukupno:=ukupno+1;
      ako je (c='B') ili (c='E') ili (c='H') ili (c='K') ili (c='N') ili
      (c='Q') ili (c='U') ili (c='X') tada ukupno:=ukupno+2;
      ako je (c='C') ili (c='F') ili (c='I') ili (c='L') ili (c='O') ili
      (c='R') ili (c='V') ili (c='Y') tada ukupno:=ukupno+3;
      ako je (c='S') ili (c='Z') tada ukupno:=ukupno+4;

     ako je abs(pos(poruka[i+1],tipkovnica)-pos(c,tipkovnica))<=3 tada
             ukupno:=ukupno+1;
// pos(c,s) je naredba koja vraća prvo pojavljivanje znaka c u stringu s
};
ispiši(ukupno);
```

Potrebno znanje: naredba ponavljanja, string

Kategorija: ad hoc

6.3. Zadatak: Fish — Autor: Nikola Dmitrović

Tipičan primjer simulacije u kome je potrebno precizno nakodirati uvjete navedene u zadatku. Za određivanje koliko je igrač s oznakom jedan dobio karata u prvom krugu igre potrebno je dobro osmisliti i posložiti podatke koji su zadani. Nakon toga navedeno određivanje je jednostavno. Za sve bodove treba paziti na nekoliko detalja.

Prvo što trebamo napraviti je dobro posložiti zadane podatke. Za lakši rad, odredimo koliko koji igrač ima kojih karti. Npr. igrač jedan ukupno ima nula jedinica, tri dvojke, itd. To ćemo postići korištenjem varijable „stanje“ koja je tablica s tri reda i deset stupaca. Može i s tri niza, ali je ovako lakše. Prilikom učitavanja podataka, za svakog igrača učitavamo njegovih deset karata i svaki put za jedan povećamo `stanje[i,j]`, pri čemu je i oznaka igrača, a j oznaka karte.

Pseudokod rješenja:

```
// varijabla stanje je tablica s 3 retka i 10 stupaca.
// varijabla spil je niz duljine 10. U njega zapisujemo stanje preostalih karti.
// varijabla poker je niz duljine 3. U njemu pratimo koliko je puta koji igrač sakupio poker.
            // učitaj podatke i kreiraj varijablu stanje
            učitaj(n);
            potez:=1; // tko je na potezu
            vrh:=1; // koja karta iz špila se uzima
            poker[1]:=0; poker[2]:=0; poker[3]:=0;
            za i:=1 do n radi
            {
                 učitaj(koga, koja);
                 ako je stanje[koga,koja]>0 tada
                 {
                      povećaj(stanje[potez,koja],stanje[koga,koja]);
                      ako je i=1 tada
                              prvi_ispis:=stanje[koga, koja];
                      stanje[koga,koja]:=0;
                 }
                 inače
                 {
                      povećaj(stanje[potez,spil[vrh]]);
                      ako je spil[vrh]<>koja tada
                           potez:=koga;
                      vrh:=vrh+1;
                 };
                  // prebroji koliko ima ostvarenih pokera
                 za ii:=1 do 3 radi
                     za jj:=1 do 10 radi
                         ako je stanje[ii,jj]=4 tada
                         {
                              povećaj(poker[ii]);
                              stanje[ii,jj]:=0;
                         };
            };

            ispiši(prvi_ispis,' ');
            ispiši(poker[1],' ',poker[2],' ',poker[3]);
```

Potrebno znanje: naredba ponavljanja, nizovi (tablica)

Kategorija: simulacija

7.1. Zadatak: Alka — Autor: Nikola Dmitrović

Očito je da prvo trebamo pronaći trenutno vodećeg alkara, tj. onog s najviše bodova i zatim tražiti koliko drugih alkara može izjednačiti ili poboljšati taj rezultat. Ako uočimo da neki alkar može osvojiti nula punata, tada je rješenje jednostavno. Naime, u tom slučaju najoptimalnija mogućnost je da svi alkari (osim jednog) dobiju nula, a jedan alkar tri punta. To ponovimo za svakog od N alkara i prebrojimo koliko njih može izjednačiti ili poboljšati trenutno najbolji rezultata.

Pseudokod rješenja:

```
učitaj(n);
za i:=1 do n radi
učitaj(alkar[i]); // alkar je niz u koji spremamo trenutni broj
                  //bodova za svakog od N alkara
max:=alkar[1]; // algoritam traženja maksimuma
za i:=2 do n radi
   ako je alkar[i]>max tada
      max:=alkar[i];

koliko:=0;
za i:=1 do n radi
   ako je abs(alkar[i]-max)<=3 tada
      koliko:=koliko+1;

ispiši(koliko);
```

Potrebno znanje: naredba ponavljanja, algoritam traženja maksimuma, niz

Kategorija: ad hoc

7.2. Zadatak: Gandalf — Autor: Nikola Dmitrović

Ovo je zadatak koji malo toga ostavlja nejasnim. Ako se zna algoritam za kreiranje broja, tada je rješenje vrlo jednostavno. Postoje i jako jednostavna rješenja koja uključuju string, ali ih ovdje nećemo spominjati.

Pseudokod rješenja:

```
učitaj(n);
ispiši(n);
dok je n>0 radi
{
    tmp:=n; // kreiraj kopiju broja
    novi:=0; pot:=1;
    dok je tmp>0 radi
    {
        ako je tmp % 10>0 tada
        {
```

```
            novi:=(tmp % 10-1)*pot+novi;
            pot:=pot*10;
        };
        tmp:=tmp div 10;
    };

    ispiši(novi);
    n:=novi;
};
```

Potrebno znanje: uvjetna naredba ponavljanja, algoritam kreiranja broja

Kategorija: ad hoc

7.3. Zadatak: Snake

Autor: Matija Milišić[2]

Ovaj zadatak riješit ćemo pomicanjem zmije po tablici znakova upravo onako kako je to zadano stringom poteza. Za to ćemo morati u svakom trenutku znati na kojoj se poziciji nalaze zmijina glava i rep te u kojem smjeru gleda njezina glava. Sav kod za kretanje po tablici nalazi se unutar for petlje koja prolazi po stringu poteza. Na početku petlje izračunavamo novi smjer zmijine glave. On ovisi o dotadašnjem smjeru i trenutnom znaku u stringu. Odmah nakon toga izračunavamo sljedeću poziciju S na koju bi se glava trebala pomaknuti. Ta pozicija ovisi o smjeru i trenutnoj poziciji glave. Ako se pozicija S nalazi izvan tablice ili se na njoj nalazi prepreka prekidamo izvođenje petlje te time zanemarujemo preostale poteze. Isto radimo u slučaju kad bi pomakom na tu poziciju zmija udarila u samu sebe. To se događa kada je na toj poziciji prije pomicanja zmijino tijelo. Iznimka je kada se radi o repu jer se tada zmija može pomaknuti bez sudara. Nakon toga slijedi pomicanje zmije. To radimo tako da na poziciju S postavimo zmijino tijelo (glavu). Ako je na toj poziciji do tada bila hrana, rep ostavimo na istom mjestu i prelazimo na sljedeći potez. U suprotnom, na poziciju repa postavimo slobodno mjesto. Tada nam još jedino preostaje izračunati novu poziciju repa, a to možemo napraviti na više načina.

Jedan od načina je da u novom nizu pamtimo sve pozicije po kojima se zmija kretala i ukupnu duljinu D njezina tijela. Prilikom svakog poteza u taj niz moramo dodati novu poziciju glave te povećati duljinu D ukoliko zmija pojede hranu. Rep se uvijek nalazi na mjestu na kojem je bila glava prije D-1 poteza.

Još jedan način je da u novoj tablici zapisujemo redni broj poteza na svako mjesto na koje dođe zmijina glava. Ako znamo broj prethodnog repa u tablici, broj sljedećeg repa je za jedan veći.

Na kraju, izvan petlje, ispišemo sadržaj tablice znakova, što je rješenje zadatka.

Potrebno znanje: naredba odlučivanja, naredba ponavljanja, nizovi znakova (string), tablica znakova

Kategorija: simulacija

[2] student Fakulteta elektrotehnike i računarstva, osvajač brončane medalje na IOI 2011., srebrne medalje na CEOI 2011. te dviju brončanih medalja na IMO 2011. i IMO 2012, dvostruki državni prvak iz informatike (2. i 4. razred)

8.1. Zadatak: Imendan Autor: Nikola Dmitrović

Toliko jednostavan zadatak da se lako ne uoče dvije stvari na koje treba paziti. Naime, zadatak kaže „Ako postoji više takvih datuma, tada se za imendan uzima onaj koji je **vremenski najbliže našem rođendanu**. Ako i nakon toga nije moguće donijeti odluku, tada se uzima onaj koji **prije dolazi u godini“.** Ako se ovo pažljivije pročita, postaje jasno da se gleda općenito vremensko uspoređivanje. To se najbolje ogleda u primjeru u kome je rođendan npr. 2.1., a kandidati za imendan su 5.1. i 31.12. Imendan će se slaviti 31.12. jer je općenito vremenski bliži rođendanu.

Ali, ako promotrimo primjer u kome je rođendan npr. 30.12., a kandidati za imendan su 1.1. i 29.12., tada će se imendan slaviti 1.1. jer je prije u godini.

Potrebno znanje: naredba odlučivanja, naredba ponavljanja

Kategorija: ad hoc

8.2. Zadatak: Vatromet Autor: Nikola Dmitrović

Rješenje je slično kao u zadatku „Gandalf“, samo što sada trebamo dodatno paziti da li znamenku povećavamo ili smanjujemo.

Pseudokod rješenja:

```
učitaj(n);
učitaj(carolija); // carolija je string
ispiši(n);
dok je n>0 radi
{
    tmp:=n; // kreiraj kopiju broja
    novi:=0; pot:=1; koja:=5; // krećemo od kraja carolije
    dok je tmp>0 radi
    {
        ako je (tmp % 10>0) i (carolija[koja]='-') tada
        {
           novi:=(tmp % 10-1)*pot+novi;
           pot:=pot*10;
        };
        ako je (tmp % 10<9) i (carolija[koja]='+') tada
        {
           novi:=(tmp % 10+1)*pot+novi;
           pot:=pot*10;
        };
        tmp:=tmp div 10;
        koja:=koja-1; // pomaknemo se u lijevo za jedno mjesto
    };
    ispiši(novi);
    n:=novi;
};
```

Potrebno znanje: naredba ponavljanja, algoritam kreiranja broja, string

Kategorija: ad hoc

8.3. Zadatak: Kolona

Autor: Antun Razum[3]

Kako bi pojednostavili rješavanje zadatka zamisliti ćemo da se cijeli sustav kreće brzinom kamiona u smjeru njihovog kretanja. Kada ovako promatramo sustav kamioni stoje na mjestu, Krešin auto se kreće brzinom od jednog kilometra u minuti u smjeru gibanja kamiona, a prekidi u traci za pretjecanje (u daljnjem tekstu „prekidi na cesti" ili samo „prekidi") kreću se također brzinom od jednog kilometra u minuti, ali u suprotnom smjeru.

Sada je prilično očito što je potrebno ispisati u prvom retku ispisa. Budući da kamioni stoje na mjestu, ako nema radova na cesti, Krešo jednostavno mora proći vodeći kilometar kamiona u koloni vozeći se maksimalnom brzinom. Kako bi to postigao mora preći KN + 1 kilometara (gdje je KN posljednji kilometar na kojemu se nalaze kamioni), a kako se Krešin auto kreće brzinom od jednog kilometra u minuti za to će mu biti potrebno točno KN + 1 minuta.

Kako bi riješili drugi dio zadatka za početak ćemo susjedne kilometre kamiona (susjedni su oni između kojih nema razmaka) grupirati u skupine kojima ćemo znati pozicije i duljine. Sada kretanje Krešinog auta možemo podijeliti na zaobilaske pojedinih skupina kamiona. Ostatak programa ćemo stoga smjestit u petlju koja će proći po svim tim skupinama kamiona. Prije same petlje stvorit ćemo dvije pomoćne varijable od kojih će nam prva služiti za brojanje ukupnog vremena koje je proteklo (to je ujedno i broj koji ćemo na posljetku ispisati), a pomoću druge ćemo pamtiti koje smo prekide na cesti do sada prošli, točnije koji je prvi prekid kojega još nismo prošli. Na početku svakog koraka petlje možemo zamisliti da se auto nalazi točno ispred (ispred znači da je Krešo pretekao tu skupinu) prethodne skupine kamiona. Za početak ćemo zato na ukupno vrijeme dodati vrijeme potrebno autu da dođe do začelja iduće skupine (za to će mu biti potrebno vremena koliki je razmak između kolona). Sada moramo provjeriti je li Krešo, dok je putovao od prethodne skupine do sljedeće, prošao neke od prekida, i ako je, povećati varijablu koja pamti koje smo prekide prošli sukladno tome. U ovom trenutku Krešo čeka iza skupine kamiona prvi prekid nakon kojeg može početi pretjecati. Sada treba proći sve prekide prije kojih Krešo ne smije pretjecati ovu skupinu kamiona (jer bi naletio na njih) te za svaki pojedini prekid ukupnom vremenu dodati vrijeme koje mu je bilo potrebno za čekanje. Kako bi znali ima li Krešo dovoljno vremena da prestigne neku skupinu kamiona prije nekog prekida moramo usporediti duljinu te skupine i razliku udaljenosti Krešinog auta i prekida (primijetite da je pozicija prekida jednaka razlici njegove početne pozicije i ukupnog proteklog vremena jer se prekidi pomiču). Ukoliko je spomenuta udaljenost veća od duljine skupine koju Krešo mora preteći uvećanoj za jedan, Krešo smije početi pretjecati. Kada smo prošli sve prekide koje je Krešo morao proći prije nego počne pretjecanje jedino nam još preostaje ukupnom vremenu dodati vrijeme potrebno Kreši da pretekne tu skupinu (ono je jednako duljini te skupine uvećanoj za jedan). Nakon petlje, kao što smo i rekli, jednostavno ispišemo ukupno vrijeme budući da je to traženo vrijeme iz zadatka.

Potrebno znanje: naredba odlučivanja, naredba ponavljanja, nizovi

[3] student Fakulteta elektrotehnike i računarstva, osvajač srebrnih medalja na IOI 2012. i CEOI 2012., dvostruki državni prvak iz informatike (1. i 3. razred)

Kategorija: ad hoc

Kod u C++:

```
#include <cstdio>
const int MAXN = 20000;
int n, m, d[MAXN], p[MAXN], x[MAXN], s[MAXN];
int main() {
    scanf("%d%d", &n, &m);
    for(int i = 0; i < n; ++i)
        scanf("%d", d + i);
    for(int i = 0; i < m; ++i)
        scanf("%d", p + i);
    int br = 1;
    for(int i = 0; i < n; ) {
        for(x[br] = d[i]; i < n && d[i] == x[br] + s[br]; ++i)
            ++s[br];
        ++br;
    }
    int t = 0, ps = 0;
    for(int i = 1; i < br; ++i) {
        t += x[i] - x[i - 1] - s[i - 1] - 1;
        for(; ps < m && p[ps] - t < x[i] - 1; ++ps);
        for(; ps < m && s[i] + 1 >= p[ps] - t - (x[i] + s[i]); ++ps)
            t = p[ps] - x[i] + 2;
        t += s[i] + 1;
    }
    printf("%d\n%d", x[br - 1] + s[br - 1], t);
    return 0;
}
```