Županijsko natjecanje iz informatike

Srednja škola

Druga podskupina (3. i 4. razred)

15. veljače 2019.

ime zadatka	**TROKUT**	**KVADRAT**	**DIONICE**
vremensko ograničenje	1 sekunda	1 sekunda	2 sekunde
memorijsko ograničenje	512 MiB	512 MiB	512 MiB
broj bodova	40	50	60
	150		

Ministarstvo znanosti i obrazovanja

Agencija za odgoj i obrazovanje

HRVATSKI SAVEZ INFORMATIČARA

Sastavni dio kompleta zadataka su i ove upute te uvodna stranica na kojoj se nalaze važni podatci o zadacima. Molimo vas da i jedno i drugo pažljivo pročitate. Na ostalim stranicama nalaze se tri zadatka. Prilikom rješavanja zadataka preporučuje se korištenje olovke i papira za skiciranje i razradu algoritma.

Nakon završetka natjecanja, članovi županijskih povjerenstava poslat će izvorne kodove vaših rješenja na automatsku evaluaciju. Nakon završetka evaluacije za vašu županiju, rezultati evaluacije bit će dostupni putem sustava za evaluaciju na adresi https://srednje.hsin.hr.

Žalbe na rezultate evaluacije možete uputiti direktno Županijskom povjerenstvu (koje će vas obavijestiti o roku i načinu predaje žalbi) ili Državnom povjerenstvu na adresu dp-informatika@azoo.hr. Rok za predaju žalbi Državnom povjerenstvu istječe *u ponoć na dan natjecanja*. Rok žalbi Državno povjerenstvo može dodatno produžiti u slučaju većih problema sa zadatcima ili sustavom za evaluaciju.

Ulazni i izlazni podatci

Kod svakog pojedinog zadatka obratite pozornost na sekcije *Ulazni podatci* i *Izlazni podatci*. Tu su definirana pravila vezana uz format ulaznih i izlaznih podataka koji mora biti strogo poštovan kako bi vaša rješenja bila ispravno evaluirana. Vaš program sa standardnog ulaza mora očekivati samo zadane ulazne podatke, a na standardni izlaz ispisivati samo tražene izlazne podatke bez ikakvih dodatnih poruka. Ako vaš program bude čekao na unos nečeg drugog osim ulaznih podataka ili ispisivao nešto drugo osim izlaznih podataka (npr. *Unesite brojeve..., Rješenje je...* i slično), nećete dobiti bodove jer evaluator to ne očekuje.

Vaši programi ne smiju pristupati datotekama ili ih kreirati te ne smiju pokretati nove procese ili koristiti više od jedne dretve (threada).

Prilikom rješavanja nekog zadatka i testiranja njegovog rješenja preporučuje se korištenje operatora redirekcije ulaza kako ne biste više puta nepotrebno unosili podatke preko tipkovnice. Na primjer, ulazne podatke za neki od oglednih primjera iz teksta zadatka možete spremiti u tekstualnu datoteku i testirati vaš program tako da ga pokrećete iz komandne linije na sljedeći način (pretpostavimo da se zadatak zove „Neboder“):

```
neboder < primjer.txt
```

Znak < je operator redirekcije ulaza i sve što se nalazi u datoteci `primjer.txt` bit će proslijeđeno vašem programu kao da je uneseno preko tipkovnice.

U slučaju Pythona potrebno je eksplicitno pozvati odgovarajući prevoditelj. Na primjer:

```
C:\Python27\python neboder.py < primjer.txt
```

Bodovanje

Da bi program koji rješava neki zadatak dobio maksimalan broj bodova, primijenjeni algoritam mora biti najprije točan, ali i efikasan tj. brz. Test podatci unaprijed su osmišljeni tako da će programi koji koriste manje efikasne, ali valjane algoritme, također dobiti određeni broj bodova (npr. od ukupno 60 bodova, vrlo spor algoritam dobit će npr. 20 bodova, dok će dobar algoritam, ali ne i najbolji, dobiti npr. 40 bodova). Jako brz program koji ne daje točne rezultate, naravno, ne donosi bodove. Valjanost algoritma je na prvom mjestu, a brzina izvršavanja na drugom.

Različite zadatke možete rješavati u različitim jezicima, imajući na umu da će rješenja u Pythonu, zbog prirode jezika, biti sporija od ekvivalentnih rješenja u drugim dozvoljenim jezicima i da bi zato mogla na nekim test podatcima premašiti vremensko ograničenje.

U (nekim) zadatcima obratite pažnju na sekciju *Bodovanje*. Ona opisuje dio test podataka (te pripadni broj bodova) koji su potencijalno lakše rješivi jer sadrže manje brojeve ili su na neki način specifični.

Evaluacija

Bodove za pojedini test podatak dobit će samo oni programi koji budu generirali točan rezultat unutar navedenog vremenskog i memorijskog ograničenja, te regularno završe svoje izvođenje. Točnije, program se treba izvršiti do kraja. U programskim jezicima C/C++ to znači do `return 0;` na kraju funkcije `main` koja treba biti deklarirana kao `int main()`, ili do naredbe `exit(0);`.

Ne trebate predate izvršnu (exe) datoteku već samo predajete datoteku s izvornim kodom. Imena datoteka moraju odgovarati imenima zadataka, a ekstenzija datoteke mora odgovarati programskom jeziku i standardu u skladu s donjom tablicom (`.c, .cpp, .cxx` ili `.py`). Npr. ako se zadatak zove „Neboder“ i ako koristite C++11, predat ćete datoteku `neboder.cxx`. Na temelju odgovarajuće ekstenzije, za jezike C/C++ sustav za evaluaciju iz vašeg će izvornog kôda kreirati izvršnu datoteku na sljedeći način:

```
C          gcc -DEVAL -O2 -o neboder neboder.c -lm
C++        g++ -DEVAL -O2 -o neboder neboder.cpp
C++11      g++ -DEVAL -std=c++11 -O2 -o neboder neboder.cxx
```

Za rješenja u Pythonu, kako bi evaluator prepoznao koristite li verziju 2 ili verziju 3, prva linija u kodu mora točno odgovarati jednom od sljedećeg:

- `#!/usr/bin/python2`
- `#!/usr/bin/python3`

Za evaluaciju će se koristiti verzije prevoditelja 2.7 odnosno 3.5, a prije izvođenja vaš kod bit će preveden u bytecode naredbom `python -m py_compile neboder.py`.

Računalo na kojem se vrši evaluacija je Linux računalo s Ubuntu 16.04 LTS 64-bitnim operativnim sustavom i sljedećim verzijama prevoditelja: gcc 4.9, g++ 4.9. Preporučujemo da svoja rješenja obavezno isprobate sa `gcc` ili `g++` prevoditeljima s gore navedenim opcijama, osobito ako koristite neki drugi alat (npr. Microsoft Visual Studio) tijekom natjecanja. Da bi vaše rješenje bilo uspješno prevedeno, morate koristiti samo standardne biblioteke, tj. ne smijete koristiti naredbe i funkcije specifične za Windowse.

Primjeri pravilno napisanih programa

Zadatak: Napišite program koji će zbrojiti i oduzeti dva cijela broja.

Ulaz: U prvom retku nalaze se dva cijela broja A i B, međusobno odvojena jednim razmakom.

Izlaz: U prvi redak ispišite zbroj, a u drugi redak razliku brojeva A i B.

C

```
#include <stdio.h>
int main(void) {
  int a, b;
  scanf("%d%d", &a, &b);
  printf("%d\n", a + b);
  printf("%d\n", a - b);
  return 0;
}
```

C++

```
#include <iostream>
using namespace std;
int main(void) {
  int a, b;
  cin >> a >> b;
  cout << a + b << endl;
  cout << a - b << endl;
  return 0;
}
```

Python 2

```
#!/usr/bin/python2
a, b = map(int, raw_input().split())
print a + b
print a - b
```

Python 3

```
#!/usr/bin/python3
a, b = map(int, input().split())
print(a + b)
print(a - b)
```

Česte pogreške

Donosimo listu **čestih i nepotrebnih pogrešaka** na natjecanjima ovog tipa. Sve od sljedećeg može rezultirati time da će evaluator dodijeliti nula bodova vašem rješenju.

• **Manjak inicijalizacije lokalnih varijabli:** U jezicima C i C++ lokalne varijable (što uključuje i varijable deklarirane unutar funkcije `main`) ne inicijaliziraju se automatski te njihova početna vrijednost nije definirana. Obavezno inicijalizirajte vrijednosti svih lokalnih varijabli, inače je moguće da vaš program dobro radi lokalno, a dobije nula bodova na sustavu za evaluaciju.

• **Rješenje alocira premala polja:** Ako ne alocirate dovoljno velika polja, moguće je da vaš program dobro radi za primjere iz teksta zadatka, a dobije nula bodova na evaluatoru. Na primjer, ako s ulaza trebate pročitati niz od najviše 1000 znakova, u jeziku C potrebno je alocirati polje od najmanje 1001 elementa (`char s[1001]`).

• **Rješenje alocira prevelika polja:** Nepotrebno glomazna polja mogu prouzročiti da vaše rješenje prekorači memorijsko ograničenje i dobije nula bodova na evaluatoru. Primjerice, u jeziku C polje deklarirano sa `int x[1024][1024][100]` koristi 400MiB memorije.

• Manjak direktive `from sys import exit` u slučaju korištenja narebe **`exit`** u jeziku Python.

• Korištenje **`conio.h`** u jezicima C i C++. Korištenje sintakse, tipova i funkcija specifičnih za Microsoftove C i C++ prevoditelje, npr. tip podataka **`int64`**. Korištenje naredbi **`itoa`** (koje nema pod Linuxom) ili **`atoi`** (koja radi malo drugačije pod Linuxom). Umjesto njih preporučamo korištenje funkcija `sscanf` i `sprintf`.

• Rješenja čekaju na pritisnutu tipku nakon ispisa rezultata, npr. zbog naredbe **`system("pause")`**.

• Rješenja ispisuju rezultate u **pogrešnom formatu**, npr. ispisuju dva broja svaki u svoj redak umjesto u istom retku.

• Rješenja ispisuju **višak podataka** na standardni izlaz, npr. **debug informacije** ili poruke poput "Upišite broj:", "Rješenje je:" i slično.

• Glavna funkcija u jezicima C i C++ definirana je kao `void main()` ili nema naredbe `return 0;`.

• Manjak potrebnih `include` direktiva u jezicima C i C++.

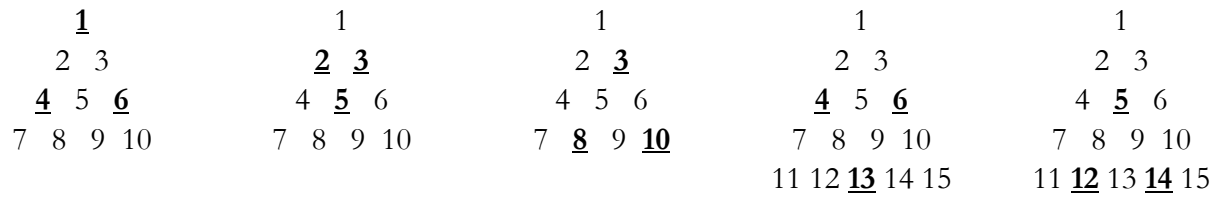
Dodatno, natjecateljima koji koriste C i C++ savjetujemo da prilikom testiranja obavezno uključe optimizacijsku opciju koja se koristi na sustavu na evaluaciju (`-O2`) te opcije koji upozoravaju na moguće probleme u kodu (primjerice `-Wall -Wextra`). Niže je ilustracija takvog prevođenja jednog rješenja sa Županijskog natjecanja 2016. godine. Prvo upozorenje koje je prijavio prevoditelj je lažna uzbuna, dok drugo ukazuje na stvarnu grešku (neinicijalizirana varijabla) zbog koje je natjecatelj nepotrebno izgubio bodove.

```
$ g++ -O2 -o poli.exe poli.cpp -Wall -Wextra
poli.cpp: In function ‘int main()’:
poli.cpp:16:20: warning: comparison between signed and unsigned integer expressions [-Wsign-compare]
  for (int i = 0; i < in.length(); ++i)
                  ^
poli.cpp:36:29: warning: ‘curr’ may be used uninitialized in this function [-Wmaybe-uninitialized]
  num[(curr == 0 ? 1 : curr) - 1] = val;
```

Niz prirodnih brojeva 1, 2, 3, 4, ... zapisali smo u trokut tako da smo u prvi red trokuta upisali jedan broj, u drugi red dva broja, u treći red tri broja, i tako dalje:

1
2 3
4 5 6
7 8 9 ...

Na ovaj način ispunili smo točno N redova trokuta. Potom smo uočili da unutar ovog trokuta možemo pronaći mnogo manjih, jednakostraničnih trokuta čiji su vrhovi neki od napisanih brojeva. Kao primjeri, na sljedećim slikama podebljane su trojke brojeva {1, 4, 6}, {2, 3, 5} {3, 8, 10}, {4, 6, 13} i {5, 12, 14} od kojih svaka tvori jednakostraničan trokut:



Zadana su dva broja A i B koji se nalaze u istom redu ili na istoj „dijagonali" unutar opisanog trokuta. Drugim riječima, pravac koji sadrži A i B paralelan je jednoj od stranica trokuta. Vaš je zadatak pronaći treći broj C unutar istog trokuta koji s brojevima A i B tvori jednakostraničan trokut.

ULAZNI PODATCI

U prvom retku nalazi se prirodan broj N ($2 \leq N \leq 500$), broj redova trokuta.

U drugom retku nalaze se međusobno različiti prirodni brojevi A i B koji zadovoljavaju uvjet iz teksta zadatka.

IZLAZNI PODATCI

Ispišite traženi broj C iz teksta zadatka. Ako broj C možemo odabrati na više načina, ispišite ih svaki u svoj redak, u rastućem poretku.

(Test podatci jamče da postoji barem jedan traženi C unutar opisanog trokuta.)

PRIMJERI TEST PODATAKA

ulaz	**ulaz**	**ulaz**
4 2 3	4 8 3	5 4 13
izlaz	**izlaz**	**izlaz**
1 5	10	6 11

Mirko želi kupiti dio velikog zemljišta dimenzija N × N metara koje predstavljamo matricom od N redaka i N stupaca podijeljenom na N × N polja. Mirko želi da njegov odabrani dio unutar zemljišta bude kvadratnog oblika, tj. da zauzima K × K polja za neki K između 1 i N (uključivo). Odabrani dio može se nalaziti bilo gdje unutar velikog zemljišta; jedini je uvjet da taj dio zemljišta bude što plodniji.

Za svako polje zemljišta (1 × 1) poznata je njegova plodnost izražena kao cijeli broj – što je broj veći, to je polje plodnije, a plodnost može biti i negativna. Mirko će procijeniti ukupnu plodnost dijela zemljišta koji planira kupiti. Na prvi pogled mogao bi samo zbrojiti plodnosti pripadnih polja, ali njegova je formula malo zanimljivija jer mu je za polja koja su bliža središtu kupljenog dijela važnije da budu plodna. Mirko će dobiti svoju procjenu na sljedeći način:

- plodnosti polja na rubu odabranog kvadrata pomnožit će s 1,
- za dio kvadrata bez tog ruba (kvadrat dimenzije smanjene za dva) plodnosti polja na njegovom rubu pomnožit će s 2,
- rub novog ostatka pomnožit će s 3, i tako dalje, množeći sve većim brojevima plodnosti polja koja su bliža središtu, kao na sljedećim skicama za K = 4 i K = 5:[1]

× 1	× 1	× 1	× 1
× 1	× 2	× 2	× 1
× 1	× 2	× 2	× 1
× 1	× 1	× 1	× 1

× 1	× 1	× 1	× 1	× 1
× 1	× 2	× 2	× 2	× 1
× 1	× 2	× 3	× 2	× 1
× 1	× 2	× 2	× 2	× 1
× 1	× 1	× 1	× 1	× 1

Zbroj navedenih umnožaka daje Mirkovu procjenu. Pomozite Mirku i pronađite kvadrat unutar zadanog zemljišta za koji će njegova procjena plodnosti biti najveća.

ULAZNI PODATCI

U prvom retku nalazi se prirodan broj N ($2 \leq N \leq 400$), dimenzija zemljišta.

U sljedećih N redaka nalazi se po N cijelih brojeva iz intervala [-1000, 1000], plodnosti odgovarajućih polja zemljišta.

IZLAZNI PODATCI

U jedini redak ispišite najveću plodnost nekog kvadratnog dijela zemljišta.

(Primjeri se nalaze na sljedećoj stranici.)

[1] Primijetite da se pojmovi ruba i središta ovdje odnose na odabrani kvadratni dio negdje unutar zemljišta, a ne na cijelo N x N zemljište. Tako je i na slici skiciran samo odabrani K × K kvadrat.

PRIMJERI TEST PODATAKA

ulaz	**ulaz**	**ulaz**
2 1 -7 4 5	3 1 2 -3 4 5 6 7 -8 -9	3 -3 4 5 7 9 -2 1 0 -6
izlaz	**izlaz**	**izlaz**
5	12	24

Pojašnjenje prvog primjera: Odabrat ćemo jedinični kvadrat plodnosti 5. Kad bismo odabrali cijeli 2×2 kvadrat, dobivena plodnost bila bi $1 - 7 + 4 + 5 = 3$.

Pojašnjenje drugog primjera: Odabrat ćemo 2 x 2 kvadrat u gornjem lijevom dijelu.

Pojašnjenje trećeg primjera: Odabrat ćemo cijeli 3×3 kvadrat.

Mali Ogi u slobodno vrijeme proučava burzu. Jednoga dana pronašao je godišnje izvješće prodaja i kupovina dionica tvrtke Jadranska Solana. Izvješće se sastoji od N **kronološki** poredanih zapisa od kojih svaki označava zahtjev za kupnjom ili prodajom jedne dionice tvrtke po određenoj cijeni. Točnije, za svaki od N zapisa poznate su dvije informacije: S_i – vrsta zahtjeva (kupnja ili prodaja), te X_i – ponuđena cijena.

Gledajući ove brojeve, Ogi se zapitao koliko bi najviše mogao zaraditi tijekom dane godine na kupnji i prodaji dionica Jadranske Solane. Za svaki zahtjev za prodajom, Ogi može odlučiti kupiti točno jednu dionicu po ponuđenoj cijeni; te za zahtjev za kupnjom, Ogi može odlučiti prodati točno jednu dionicu po ponuđenoj cijeni ako u tom trenutku posjeduje barem jednu dionicu.

Iako već zna da burza ne funkcionira tako jednostavno, pretpostavio je da zahtjevi na burzi ne ovise o njegovim postupcima, to jest da bi se isti niz zahtjeva dogodio čak i da je on trgovao na burzi. Također, kako se u izvješću ne nalaze informacije o tome kada je neki zahtjev povučen, Ogi može kupiti ili prodati dionicu jedino u trenutku kada je postavljen određeni zahtjev. Da ne bi morao brinuti oko detalja, Ogi pretpostavlja da na početku ima **neograničen kapital** te da **ne posjeduje nijednu dionicu**.

Kako je hrvatska burza vrlo prometna te se izvješće sastoji od velikog broja zapisa, Ogi vas je zamolio da napišete program koji će izračunati najveću moguću zaradu na dionicama Jadranske Solane. Zaradu definiramo kao ukupan iznos dobiven prodajom dionica umanjen za ukupan iznos potrošen na kupnju dionica.

ULAZNI PODATCI

U prvom retku nalazi se prirodan broj N ($1 \leq N \leq 500\,000$), broj zapisa na burzi.

U idućih N redaka nalaze se po dva prirodna broja S_i ($1 \leq S_i \leq 2$) i X_i ($1 \leq X_i \leq 10^9$) odvojeni razmakom. Ako je $S_i = 1$, tada *i* -ti zapis opisuje zahtjev za kupnjom jedne dionice. U suprotnom je $S_i = 2$, te *i*-ti zapis opisuje zahtjev za prodajom jedne dionice. X_i označava ponuđenu cijenu kupnje/prodaje.[1]

IZLAZNI PODATCI

U jedini redak ispišite traženu najveću moguću zaradu.

BODOVANJE

U nekim test podatcima cijene zahtjeva za kupnjom bit će padajuće, tj. ako je $S_i = S_j = 1$ za neke $i < j$, vrijedit će $X_i \geq X_j$. Također, u nekim test podatcima postoji manje ograničenje na broj dionica. Ova dodatna ograničenja prikazujemo detaljno u sljedećoj tablici, pri čemu se test podatci prikazani različitim redcima ne preklapaju:

(Tablica i primjeri nalaze se na sljedećoj stranici.)

[1] Da ne bude zabune, Ogi na zahtjev za *kupnjom* ima mogućnost *prodati* dionicu, a na zahtjev za *prodajom* ima mogućnost *kupiti* dionicu. Pogledajte npr. pojašnjenje nakon prvog primjera niže.

Postotak bodova	*Ograničenja*
10%	N ≤ 20, cijene zahtjeva za kupnjom su padajuće
10%	N ≤ 20
10%	N ≤ 2000, cijene zahtjeva za kupnjom su padajuće
10%	N ≤ 2000
20%	N ≤ 500 000, cijene zahtjeva za kupnjom su padajuće
40%	N ≤ 500 000

PRIMJERI TEST PODATAKA

ulaz

```
5
2 2
1 5
2 1
1 4
1 6
```

izlaz

```
8
```

ulaz

```
3
2 200
1 100
2 50
```

izlaz

```
0
```

ulaz

```
6
2 20
2 40
2 30
1 10
1 70
1 50
```

izlaz

```
70
```

Pojašnjenje prvog primjera:
Prvi zapis: Ogi kupuje dionicu po cijeni 2.
Drugi zapis: Ogi prodaje dionicu po cijeni 5.
Treći zapis: Ogi kupuje dionicu po cijeni 1.
Četvrti zapis: Ogi odlučuje ne prodati dionicu po cijeni 4.
Peti zapis: Ogi prodaje dionicu po cijeni 6.
Ogijeva ukupna zarada: -2 + 5 - 1 + 6 = 8.

Pojašnjenje drugog primjera: Jedina je mogućnost zarade prodati dionicu za cijenu 100, no prije toga morao bi ju kupiti po višoj cijeni. U ovom primjeru najisplativija je strategija ne trgovati te je stoga Ogijeva zarada 0.