

19. siječnja 2024.

# **Natjecanje**<sup>2024</sup> *iz informatike*

Školska razina 2024./ Osnovna škola (5. i 6. i 7. i 8. razred)

Primjena algoritama OŠ

## **OPISI ALGORITAMA**

Zadatke pripremili: Nikola Dmitrović, Fran Babić, Vito Anić, Stjepan Požgaj



Agencija za odgoj i obrazovanje  
Education and Teacher Training Agency



HRVATSKI SAVEZ  
INFORMATIČARA



Ministarstvo znanosti,  
obrazovanja i sporta



## 5.1. Zadatak: Tie

Iz teksta zadatka možemo zaključiti da će za sve ulazne podatke manje ili jednake od 5 ispis biti 7. Za sve ostale vrijednosti ispis će morati biti za dva veći od ulazne vrijednosti.

*Programski kod (Python):*

```
N = int(input())
if N <= 5:
    print(7)
else:
    print(N + 2)
```

**Potrebno znanje:** naredba odlučivanja

**Kategorija:** ad hoc

## 5.2. Zadatak: IKEA

Za rješenje trebamo odrediti proizvod čija je cijena druga po vrijednosti od cijena triju odabranih proizvoda (ako je to moguće odrediti). Uočimo da su vrijednosti proizvoda složene po veličini. Nekoliko je slučajeva:

- ako su sve tri cijene jednake, tada nije moguće odrediti traženi proizvod te je rješenje zbroj vrijednosti  $C_1$ ,  $C_2$  i  $C_3$ ;
- ako su prve dvije vrijednosti jednake i zbog zapažanja da su onda najveće po vrijednosti, tada je vrijednost  $C_3$  druga po vrijednosti te je rješenje zbroj vrijednost  $C_1$  i  $C_2$  (jednake su);
- ako su druga i treća vrijednosti jednake i zbog zapažanja da je prva vrijednost onda najveća, tada su vrijednost  $C_2$  i  $C_3$  druge po vrijednosti te je rješenje zbroj vrijednost  $C_1$  i jedne od vrijednosti  $C_2$  ili  $C_3$ ;
- ako su sve tri cijene različite, tada je druga po vrijednosti vrijednost  $C_2$  te je rješenje zbroj vrijednosti  $C_1$  i  $C_3$ ;

*Programski kod:*

```
C1 = int(input())
C2 = int(input())
C3 = int(input())
if C1 == C2 == C3:
    print(C1 + C2 + C3)
elif C1 == C2:
    print(C1 + C2)
elif C2 == C3:
    print(C1 + C2)
else:
    print(C1 + C3)
```



**Potrebno znanje:** naredba odlučivanja

**Kategorija:** ad hoc

### 5.3. Zadatak: EURO

Iz ulaznih podataka znamo ishod svi šest odigranih utakmica te možemo odrediti kako je koja reprezentacija odigrala svoje utakmice. Pomaže nam činjenica da je utakmica mogla završiti samo pobjedom/porazom reprezentacije.

Četiri boda po testnom primjeru jednostavno se moglo dobiti zbrajanjem prva tri ulazna podatka i množenjem zbroja s tri. Savjet, ako ne znate rješenje cijelog zadatka, pokušajte riješiti dio koji će vam donijeti bodove jer kako bi to rekli u stilu Alan Forda - bolje je imati 40 bodova nego nula.

*Programski kod:*

```
H1 = int(input())
H2 = int(input())
H3 = int(input())
I1 = int(input())
S2 = int(input())
A3 = int(input())

S1 = 1 - H1 (ako je Hrvatska pobijedila (1) u prvoj utakmici tada je Španjolska
izgubila (1-1=0) te obrnuto)
S3 = 1 - A3
I2 = 1 - S2
I3 = 1 - H3
A1 = 1 - I1
A2 = 1 - H2

# Hrvatska
H = (H1 + H2 + H3) * 3 (pobjeda (1) vrijedi tri boda, poraz (0) nula bodova)

# Španjolska/Italija/Albanija
S = (S1 + S2 + S3) * 3
I = (I1 + I2 + I3) * 3
A = (A1 + A2 + A3) * 3

# Ispis
print(H)
print(S, I, A)
```



**Potrebno znanje:** naredba učitavanja i ispisa

**Kategorija:** ad hoc

## 6.1. Zadatak: IKEA

Vidi 5.2

## 6.2. Zadatak: EURO24

Iz ulaznih podataka znamo ishod svi šest odigranih utakmica te možemo odrediti kako je koja reprezentacija odigrala svoje utakmice. Prilikom određivanja ishoda, možemo odmah odrediti i broj bodova koje je reprezentacija osvojila za tu utakmicu.

U primjerima vrijednima 35 bodova utakmica nije završila neriješenim ishodom i rješenje tog dijela zadatka možete pogledati pod 5.3 EURO.

*Programski kod:*

```
H1, H2, H3 = map(int, input().split()) #učitavanje tri podatka u jednom retku
I1 = int(input())
S2 = int(input())
A3 = int(input())

I2 = I3 = S1 = S3 = A1 = A2 = 0

#S1 - odredimo kako je Španjolska odigrala prvu svoju utakmicu
if H1 == 0:
    S1 = 3 #ako je H izgubila, Š je pobijedila i dobila 3 boda
elif H1 == 1:
    S1 = 1 #ako je H igrala neriješeno, Š je igrala neriješeno i dobila 1 bod
elif H1 == 2:
    S1 = 0 #ako je H pobijedila, Š je izgubila i nije dobila bodove
#S3
if A3 == 0:
    S3 = 3
elif A3 == 1:
    S3 = 1
elif A3 == 2:
    S3 = 0
#I2
if S2 == 0:
    I2 = 3
```



```
elif S2 == 1:
    I2 = 1
elif S2 == 2:
    I2 = 0

#I3
if H3 == 0:
    I3 = 3
elif H3 == 1:
    I3 = 1
elif H3 == 2:
    I3 = 0

#A1
if I1 == 0:
    A1 = 3
elif I1 == 1:
    A1 = 1
elif I1 == 2:
    A1 = 0

#A2
if H2 == 0:
    A2 = 3
elif H2 == 1:
    A2 = 1
elif H2 == 2:
    A2 = 0

# pretvori u bodove ishode utakmica iz ulaznih podataka
#S2, I1, A3
if S2 == 2: S2 = 3
if I1 == 2: I1 = 3
if A3 == 2: A3 = 3

#Hi
if H1 == 2: H1 = 3
if H2 == 2: H2 = 3
if H3 == 2: H3 = 3
```



```
# ukupno bodova Hrvatska
```

```
H = H1 + H2 + H3
```

```
# ukupno bodova Španjolska/Italija/Albanija
```

```
S = S1 + S2 + S3
```

```
I = I1 + I2 + I3
```

```
A = A1 + A2 + A3
```

```
# Ispis
```

```
print(H)
```

```
print(S, I, A)
```

**Potrebno znanje:** naredba odlučivanja

**Kategorija:** ad hoc

### 6.3. Zadatak: Alone

Redom, za svaku zadanu vrijednost, odredimo znamenku jedinica (j), desetica (d) i stotica (s). Ako je j jednako 1, znači da se film d prvi put prikazuje na televiziji s. Je li to prva premijera ili samo premijera ovisi o tome je li film već prikazan na nekoj drugoj televiziji. Podatak o tome je li film već nekad prikazan, pratiti ćemo u posebnoj listi.

*Programski kod:*

```
N = int(input())
```

```
filmovi = 10 * [0]
```

```
print(filmovi)
```

```
repriza = premijera = nova_premijera = 0
```

```
for i in range(N):
```

```
    Z = int(input())
```

```
    s = Z // 100
```

```
    d = Z // 10 % 10
```

```
    j = Z % 10
```

```
    filmovi[d] += 1
```

```
    if j > 1:
```

```
        repriza += 1
```

```
    else:
```

```
        if filmovi[d] != 1:
```



```
        premijera += 1
    else:
        nova_premijera += 1

print(nova_premijera)
print(premijera)
print(repriza)
```

**Potrebno znanje:** naredba ponavljanja, odlučivanja i jednostavna uporaba lista

**Kategorija:** ad hoc

## 7.1. Zadatak: EURO

Vidi 5.3

## 7.2. Zadatak: Skijanje

Kada učitamo pojedino vrijeme vožnje, pretvorit ćemo ga iz danog formata u jednostavno vrijeme u stotinkama ( $cc+100*ss+60*100*mm$ ). Također, može se pretvoriti u vrijeme izraženo u minutama ili sekundama, ali izabrali smo stotinke da izbjegnemo decimalne brojeve.

Jednostavno je pridružiti svako vrijeme prve vožnje odgovarajućoj natjecateljici. Međutim, potrebno je pravilno poredati natjecateljice po redu kako će voziti u drugoj vožnji. Za to je potrebno upotrijebiti jedan od algoritama za sortiranje, bila to postojeća implementacija kao npr. `std::sort` u C++ ili funkcija `sort()` u Pythonu, ili da sami implementiramo neki algoritam za sortiranje. Primjer implementacije navedenoga možete vidjeti u službenom rješenju.

*Programski kod (C++):*

```
#include <bits/stdc++.h>

#define X first
#define Y second

using namespace std;
typedef long long llint;

int n;
int t[maxn];
int v[maxn];

bool cmp(int x, int y) {
    return t[x] < t[y];
}
```



```
int main() {
    scanf("%d", &n);
    for (int i = 0; i < n; i++) {
        int a, b, c;
        scanf("%d:%d.%d", &a, &b, &c);
        t[i] = c + 100 * b + 60 * 100 * a;
    }
    for (int i = 0; i < n; i++) v[i] = i;
    sort(v, v+n, cmp);
    reverse(v, v+n);
    for (int i = 0; i < n; i++) {
        int a, b, c;
        scanf("%d:%d.%d", &a, &b, &c);
        t[v[i]] += c + 100 * b + 60 * 100 * a;
    }
    sort(v, v+n, cmp);
    printf("%d\n", v[0] + 1);

    int k;
    scanf("%d", &k);
    printf("%d\n", v[k - 1] + 1);
    return 0;
}
```

**Potrebno znanje:** liste/nizovi

**Kategorija:** ad hoc

### 7.3. Zadatak: 2P

U testnim primjerima gdje je duljina riječi manja ili jednaka 3, rastavljamo na slučajeve. Jednostavnom razradom svih slučajeva dolazimo do zaključka da će odgovor biti 0 ako riječ zadovoljava sve uvjete navedene u zadatku, a 1 inače.

Jedna od očitih strategija je isprobavanje svih mogućih izmjena slova i provjeravanje zadovoljava li izmijenjena riječ (uključujući i originalnu riječ) sve uvjete navedene u zadatku. Od svih takvih riječi, treba ispisati najmanji broj zamjena neke riječi u tom skupu.

Međutim, postoji efikasnije i elegantnije rješenje ako uočimo sljedeće: prva dva slova u riječi jednoznačno određuju ostatak riječi. Treće slovo mora biti jednako prvom, četvrto slovo mora biti jednako drugom, peto slovo mora biti jednako trećem tj. prvom... Sada možemo doći do sljedeće strategije pretraživanja: fiksirajmo prva dva (različita) slova izmijenjene riječi, jednostavno generiramo





ostatak te riječi i usporedimo koliko je izmjena bilo potrebno da dođemo od originalne riječi do trenutne izmijenjene riječi. Detalje u implementaciji možete vidjeti u službenom rješenju.

*Programski kod (C++):*

```
#include <bits/stdc++.h>

#define X first
#define Y second

using namespace std;
typedef long long llint;

int n;
string s;
int main() {
    cin >> s; n = s.size();
    int sol = n;
    for (char a = 'a'; a <= 'z'; a++) {
        for (char b = 'a'; b <= 'z'; b++) {
            if (a == b) continue;
            string tr = " ";
            tr[0] = a, tr[1] = b;

            int tren = 0;
            for (int i = 0; i < n; i++)
                tren += (s[i] != tr[i % 2]);
            sol = min(sol, tren);
        }
    }
    printf("%d\n", sol);
    return 0;
}
```

**Potrebno znanje:** string

**Kategorija:** ad hoc



## 8.1. Zadatak: Alone

Vidi 6.3

## 8.2. Zadatak: 3P

U test primjerima gdje je duljina riječi manja ili jednaka 3, rastavljamo na slučajeve. Jednostavnom razradom svih slučajeva dolazimo do zaključka da će odgovor biti 0 ako su sva slova u riječi međusobno različita, a 1 inače.

Jedna od očitih strategija je isprobavanje svih mogućih izmjena slova i provjeravanje zadovoljava li izmijenjena riječ (uključujući i originalnu riječ) sve uvjete navedene u zadatku. Od svih takvih riječi, treba ispisati najmanji broj zamjena neke riječi u tom skupu.

Međutim, postoji efikasnije i elegantnije rješenje ako uočimo sljedeće: prva tri slova u riječi jednoznačno određuju ostatak riječi. Čvrto slovo mora biti jednako prvom, peto slovo mora biti jednako drugom, šesto slovo mora biti jednako trećem, sedmo slovo mora biti jednako četvrtom tj. prvom... Sada možemo doći do sljedeće strategije pretraživanja: fiksirajmo prva tri (različita) slova izmijenjene riječi, jednostavno generiramo ostatak te riječi i usporedimo koliko je izmjena bilo potrebno da dođemo od originalne riječi do trenutne izmijenjene riječi. Potrebno je dodatno paziti na slučaj kada je duljina riječi manja ili jednaka 3, ali već smo opisali kako riješiti taj slučaj. Detalje u implementaciji možete vidjeti u službenom rješenju.

*Programski kod (C++):*

```
#include <bits/stdc++.h>

#define X first
#define Y second

using namespace std;
typedef long long llint;

int n;
string s;

int main() {
    cin >> s; n = s.size();
    if (n <= 3) {
        bool flag = false;
        for (int i = 1; i < n; i++)
            if (s[i] == s[i - 1]) flag = true;
        printf("%d\n", flag);
        return 0;
    }
```



```
int sol = n;
for (int a = 'a'; a <= 'z'; a++) {
    for (int b = 'a'; b <= 'z'; b++) {
        if (a == b) continue;
        for (int c = 'a'; c <= 'z'; c++) {
            if (a == c || b == c) continue;
            string tr = " ";
            tr[0] = a, tr[1] = b, tr[2] = c;

            int cnt = 0;
            for (int i = 0; i < n; i++)
                cnt += (s[i] != tr[i % 3]);
            sol = min(sol, cnt);
        }
    }
}

printf("%d\n", sol);
return 0;
}
```

**Potrebno znanje:** string

**Kategorija:** ad hoc

### 8.3. Zadatak: Slagalica

Rješenje ovog zadatka sastoji se od dva dijela. U prvom dijelu opisujemo kako generirati sve moguće kombinacije riječi koje možemo dobiti slaganjem nekoliko riječi u nekom poretku, ali ne moraju nužno biti različite. U drugom dijelu opisujemo kako izbaciti duplikate tj. odrediti broj različitih riječi od svih dobivenih iz prvog dijela.

Da bi generirali sve moguće kombinacije, najjednostavnije je pronaći sve permutacije niza riječi i za svaku permutaciju, u moguće spojene riječi uvrstavamo prvu riječ u permutaciji, zatim spojene prve dvije riječi u permutaciji, spojene prve tri riječi u permutaciji. Permutacije se mogu generirati rekursivno ili s pomoću standardnih funkcija poput `std::next_permutation()` u C++-u.

Iz prethodno generiranih riječi, potrebno je izdvojiti sve različite. Može se za svaku riječ u nizu mogućnosti provjeravati nalazi li se njoj jednaka riječ negdje prije u nizu. Također, postoje i strukture koje nam osiguravaju da su svi elementi unutar strukture međusobno različiti, poput strukture `set` u Pythonu i C++-u. Detalje implementacije možete vidjeti u službenom rješenju.

*Programski kod (C++):*

```
#include <bits/stdc++.h>

#define X first
```



```
#define Y second

using namespace std;
typedef long long llint;

int n;
string s[maxn];

int main() {
    cin >> n;
    for (int i = 0; i < n; i++)
        cin >> s[i];
    sort(s, s+n);

    set< string > m;
    do {
        string t;
        for (int i = 0; i < n; i++) {
            t += s[i];
            m.insert(t);
        }
    } while (next_permutation(s, s+n));
    cout << m.size() << endl;
    return 0;
}
```

**Potrebno znanje:** string, permutacije, sortiranje

**Kategorija:** simulacija