

Konstrukcija i analiza algoritama 2 - pismeni ispit

JANUAR1

25.02.2026.

Ispit traje 3h. Svaki zadatak nosi po 10 poena. Ukupan broj poena na pismenom delu ispita je 60 poena. Prag za prolaz je 17 poena.

Nema ništa posebno u vezi sa brojem 3

- **(5 poena)** Problem k -obojivosti je definisan na sledeći način: Dat je neusmeren graf G . Potrebno je odrediti da li je moguće obojiti svaki čvor grafa jednom od k boja, tako da svaka grana povezuje čvorove različitih boja. Drugim rečima, traži se pravilno bojenje grafa sa najviše k boja. Dokazati da je problem 4-obojivosti NP-težak svodjenjem problema 3-obojivosti na njega. (Hint: upotrebiti kompletan graf (graf gde je svaki čvor spojen sa svakim) sa 4 čvora kao dodatak. Pravilno 4-bojenje će svakom čvoru iz ovog dodatka dodeliti različitu boju.)
- **(5 poena)** Problem k -SAT je definisan na sledeći način: Data je Bulova formula Φ u konjunktivnoj normalnoj formi (CNF - konjukcija disjunkcija), pri čemu svaka klauza sadrži tačno k literala (promenljiva ili negacija promenljive). Potrebno je utvrditi da li postoji dodela istinitosnih vrednosti promenljivama tako da formula Φ bude zadovoljena. Dokazati da je problem 4SAT NP-težak svodjenjem 3SAT problema na njega.

Max na hiperkocki

Na raspolaganju je 2^n procesora povezanih u n -dimenzionu hiperkocku. Svaki procesor poseduje po jedan broj. Potrebno je konstruisati paralelni algoritam vremenske složenosti $O(n)$ za nalaženje najvećeg od svih zadatih elemenata.

Odgovor strukturirati kroz sledeće korake.

- **(3 poena) Definicija modela.**
Precizno definisati adresiranje procesora u hiperkocki i pojam susedstva.

- **(3 poena) Opis komunikacije.**
Objasniti kako se izvršavanje organizuje po rundama i na koji način procesori u svakoj rundi biraju partnera za komunikaciju.
- **(3 poena) Pseudokod algoritma.**
Napisati jasan pseudokod koji opisuje ponašanje jednog procesora.
- **(1 poen) Složenost.**
Objasнити vremensku složenost predloženog algoritma.

Šahovska tabla i domine

Posmatrajmo pravougaonu šahovsku tablu dimenzija $m \times n$ i domine dimenzija 2×1 . Popločavanje (tiling) je raspored domina koji u potpunosti prekriva sva polja table (tj. bez preklapanja, bez dijagonalnog postavljanja, bez izlaska van granica table i slično).

- **(4 poena)** Modelovati problem postojanja popločavanja kao problem uparivanja u bipartitivnom grafu.
- **(3 poena)** Za koje m i n popločavanje postoji?
- **(3 poena)** Neka je $m = 2$. Dati rekurentnu vezu za $T(2, n)$ gde je T broj različitih popločavanja.

Ponavljanje je majka znanja

Data je implementacija naivnog kompresovanog sufiksnog stabla za zadatu string s . Svaka ivica čuva interval $[l, r)$ u originalnom stringu, umesto da eksplicitno čuva podstring.

Podsetimo se važne osobine:

Unutrašnji čvor sufiksnog stabla (čvor koji nije list i nije koren) predstavlja podstring koji se pojavljuje najmanje dva puta u stringu.

Zadatak je da se pronadje **najduži ponovljeni segment** u stringu, odnosno podstring maksimalne dužine koji se pojavljuje bar dva puta. **Ponovljeni podstringovi mogu da se preklapaju.** Funkcija

```
std::pair<int,int> longest_repeated_substring_pos_len() const
```

već je implementirana i koristi DFS obilazak stabla. Medjutim, funkcija

```
void longest_repeat_dfs(int v,
                        int depth,
                        int& best_len,
                        int& best_node) const;
```

nije implementirana.

Vaš zadatak:

Dopuniti funkciju `longest_repeat_dfs` tako da:

- **(3 poena)** obilazi stablo u dubinu (DFS),
- **(3 poena)** za svaki unutrašnji čvor proverava da li je dubina (dužina putanje od korena do tog čvora) veća od trenutno najbolje pronadjene,
- **(2 poen)** ažurira promenljive `best_len` i `best_node` kada pronadje duži ponovljeni segment,
- **(2 poena)** korektno prosledjuje dubinu pri prelasku preko ivice (dužina ivice je $r - l$).

Napomena:

- Koren (čvor 0) se ne smatra ponovljenim segmentom.
- Listovi ne predstavljaju ponovljene segmente.
- Dubina čvora jednaka je zbiru dužina svih ivica na putanji od korena do tog čvora.

Implementirati samo telo funkcije:

```
void longest_repeat_dfs(int v,
                       int depth,
                       int& best_len,
                       int& best_node) const {
    // VAS KOD
}
```

Primer ulaza: banananananannanana

Odgovarajući izlaz: ananananan

Randomize

Na raspolaganju vam je funkcija `rand()` iz standardne biblioteke jezika C++.

Funkcija `rand()` vraća pseudo-slučajan ceo broj iz intervala

$$0, 1, 2, \dots, \text{RAND_MAX}.$$

Pretpostavite da su vrednosti koje vraća `rand()` uniformno raspodeljene u navedenom opsegu.

Potrebno je implementirati sledeće tri funkcije:

1. (2 poena) Funkciju

```
int rand2();
```

koja vraća 0 ili 1 sa istom verovatnoćom.

2. (4 poena) Funkciju

```
int rand5();
```

koja vraća broj iz skupa $\{0, 1, 2, 3, 4\}$ sa istim verovatnoćama. Funkcija `rand5()` sme da koristi samo funkciju `rand2()` (ali ne direktno `rand()`).

3. (4 poena) Funkciju

```
int randM(int M);
```

koja vraća broj iz skupa $\{0, 1, 2, \dots, M - 1\}$ sa istim verovatnoćama. Funkcija `randM()` sme da koristi isključivo funkciju `rand2()` (ne sme direktno koristiti `rand()`).

Provera uniformnosti generatora

U funkciji `main()` vrši se empirijska provera uniformnosti implementiranih funkcija `rand5()` i `randM()`.

Najpre se testira funkcija `rand5()`. Formira se vektor

$$c = (c_0, c_1, c_2, c_3, c_4),$$

gde element c_i predstavlja broj pojavljivanja vrednosti i u 10000 uzastopnih poziva funkcije `rand5()`.

Ako je implementacija korektna i distribucija uniformna, očekivani broj pojavljivanja svake vrednosti jednak je

$$\frac{10000}{5} = 2000.$$

Zbog slučajne prirode generisanja, vrednosti $c_0, \dots, c_4$ neće biti potpuno jednake, ali bi trebalo da budu približno iste veličine, bez značajnih odstupanja.

Analogno, za funkciju `randM(x)` formira se vektor

$$d = (d_0, d_1, \dots, d_{x-1}),$$

gde d_i označava broj pojavljivanja vrednosti i u 10000 poziva funkcije `randM(x)`.

U slučaju uniformne distribucije očekivana vrednost svakog brojača je

$$\frac{10000}{x}.$$

Ponovo, zbog slučajnosti su dozvoljena manja odstupanja, ali sve vrednosti bi trebalo da budu približno jednake. Znatno veće razlike između pojedinih brojača ukazivale bi na neuniformnu raspodelu i grešku u implementaciji.

Napomena o funkciji `rand2()`

Funkcija `rand2()` predstavlja osnovni izvor slučajnosti (fer novčić), koji vraća 0 ili 1 sa verovatnoćom $\frac{1}{2}$.

Ukoliko student ne uspe da implementira `rand2()`, može je posmatrati kao već zadatu funkciju (crnu kutiju) sa navedenom osobinom. Suština zadatka je konstrukcija funkcija `rand5()` i `randM()` isključivo korišćenjem takvog uniformnog binarnog generatora.

Provera ispravnosti crveno-crnog stabla

Data je struktura čvora crveno-crnog stabla:

```
enum Color { RED, BLACK };
```

```
struct Node {  
    int key;  
    Color color;  
    Node* left;  
    Node* right;  
};
```

Implementirana je funkcija

```
static bool validateRB(Node* u,  
                        int blackCount,  
                        int& targetBlack);
```

koja treba da proveriti da li podstablo sa korenom u čvoru u zadovoljava svojstva crveno-crnog stabla. Parametri funkcije imaju sledeće značenje:

- **blackCount** — broj crnih čvorova na putanji od korena do trenutnog čvora,
- **targetBlack** — referentna (očekivana) crna visina, koju moraju imati sve putanje od korena do NIL listova.

Funkcija treba da proveriti sledeća svojstva:

1. Svi NIL listovi (nullptr) smatraju se crnim.
2. Nijedan crveni čvor ne sme imati crveno dete.
3. Sve putanje od korena do NIL listova moraju imati isti broj crnih čvorova (jednaku crnu visinu).

Dopuniti označene delove funkcije:

```
static bool validateRB(Node* u,  
                        int blackCount,  
                        int& targetBlack)
```

Raspodela poena:

- **(2 poena)** Ispravna obrada NIL listova i postavljanje referentne crne visine.
- **(2 poena)** Korektno ažuriranje broja crnih čvorova.
- **(3 poena)** Provera da crveni čvor nema crveno dete.
- **(3 poena)** Ispravna rekurzivna provera levog i desnog podstabla.

Napomena: Funkcija treba da radi u vremenskoj složenosti $O(n)$, gde je n broj čvorova u stablu.