

Konstrukcija i analiza algoritama 2 - pismeni ispit SEP1

05.09.2026.

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.

Pažljivo 5-bojenje

Pravilno **5-bojenje** prostog neusmerenog grafa $G = (V, E)$ predstavlja dodelu boja iz skupa $\{0, 1, 2, 3, 4\}$ čvorovima grafa, tako da svaka dva susedna čvora imaju različite boje.

Pravilno 5-bojenje nazivamo **pažljivim** ako boje susednih čvorova nisu ni uzastopne po modulu 5. Preciznije, za svaku granu $\{u, v\} \in E$ mora da važi

$$(c(u) - c(v)) \bmod 5 \in \{2, 3\}.$$

Na primer, susedni čvorovi smeju imati boje 0 i 2, ali ne smeju imati boje 0 i 1, niti boje 0 i 4. Nije neophodno upotrebiti svih pet boja.

Posmatramo sledeći problem odlučivanja:

Dat je prost neusmeren graf G . Da li postoji pažljivo 5-bojenje grafa G ?

1. **(1 poen)** Navesti primer grafa koji ima pravilno 5-bojenje, ali nema pažljivo 5-bojenje. Obrazložiti odgovor.
2. **(2 poena)** Dokazati da problem pažljivog 5-bojenja pripada klasi NP. Navesti sertifikat, opisati njegovu proveru i odrediti vremensku složenost provere.
3. **(5 poena)** Dokazati da je problem pažljivog 5-bojenja NP-težak, redukcijom standardnog problema 5COLOR na njega.

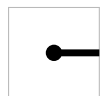
Problem 5COLOR glasi: za dati graf G odrediti da li postoji pravilno 5-bojenje grafa G . Može se koristiti poznata činjenica da je ovaj problem NP-kompletni.

Polazeći od proizvoljnog grafa G , konstruisati graf H tako da G ima pravilno 5-bojenje ako i samo ako H ima pažljivo 5-bojenje. Dokazati oba smera ekvivalencije i obrazložiti da se konstrukcija izvršava u polinomijalnom vremenu.

4. **(2 poena)** Navesti jednu klasu grafova za koju se problem pažljivog 5-bojenja može rešiti u polinomijalnom vremenu. Opisati odgovarajući algoritam i navesti njegovu vremensku složenost. Rad algoritma ilustrovati na jednom grafu iz izabrane klase.

Puzla sa cevima

Data je pravougaona tabla čije je svako polje prazno ili sadrži jednu od dve vrste pločica prikazanih na slici.



Završetak (E)



Koleni (L)



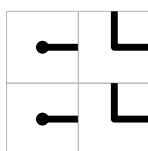
Prazno polje (.)

Crna tačka označava zatvoren kraj cevi, a otvori se nalaze na stranama pločice do kojih cev dopire.

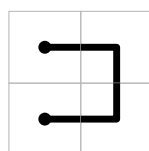
Pločice se mogu proizvoljno rotirati za umnožak od 90° , ali se ne mogu pomerati na druga polja. Oznake E i L predstavljaju vrstu pločice, a ne njenu orijentaciju.

Potrebno je okrenuti pločice tako da **svaki otvor bude spojen sa otvorom susedne pločice preko zajedničke stranice**. Otvor ne sme biti okrenut van table, prema praznom polju ili prema zatvorenoj strani susedne pločice. Nije potrebno da sve cevi pripadaju istoj povezanoj celini.

Na slici je prikazana jedna tabla i jedno njeno rešenje.



Tabla



Jedno rešenje

Zadatak. Opisati algoritam koji za proizvoljnu zadatu tablu određuje **da li rešenje postoji**, svodenjem na problem uparivanja sa najvećim brojem grana u bipartitnom grafu. Nije potrebno ispisivati orijentacije pločica.

Bodovanje:

- **(3 poena)** Objasniti šta predstavljaju čvorovi grafa i kako se raspoređuju u dva dela bipartitnog grafa.
- **(3 poena)** Objasniti između kojih čvorova se dodaju grane.
- **(4 poena)** Navesti pod kojim uslovom pronađeno uparivanje daje potvrđan odgovor i obrazložiti ispravnost redukcije u oba smera.

Napomena: U svakoj orijentaciji koleno ima tačno jedan horizontalni i tačno jedan vertikalni otvor.

Kuglice

Neka je data crna kutija puna kuglica. Polovina kuglica je plave boje, a polovina kuglica je crvene boje. Konstruisati deterministički **(5 poena)** i probabilistički algoritam **(5 poena)** koji određuje jednu plavu kuglicu.

Paralelno obrtanje povezane liste

Data je jednostruko povezana lista čiji su čvorovi smešteni u vektoru `a`, proizvoljnim redosledom. Svaki čvor sadrži vrednost `value`, indeks sledećeg čvora `next` i pomoćno polje `reversedNext`. Vrednost `-1` označava da sledeći čvor ne postoji. Parametar `head` predstavlja indeks početnog čvora liste.

Potrebno je dopuniti C++ funkciju

```
int reverseList(std::vector<Node>& a, int head, int p);
```

tako da paralelno obrne redosled čvorova u listi i vrati indeks njenog novog početka.

Nije dozvoljeno premeštati čvorove u vektoru, menjati njihove vrednosti niti menjati veličinu vektora. Menjaju se samo veze između čvorova, uz korišćenje pomoćnog polja `reversedNext`.

Pretpostaviti da svi čvorovi vektora pripadaju jednoj ispravnoj listi bez ciklusa, da je $p \geq 1$ i da broj čvorova staje u tip `int`. Za praznu listu važi `head = -1`, a funkcija treba da vrati `-1`.

Pomoćno polje može sadržati podatke iz prethodnog poziva funkcije. Za potrebe zadatka pretpostaviti da pokretanje niti uvek uspeva i da ne dolazi do izuzetaka.

Organizacija algoritma. U prvoj paralelnoj fazi formiraju se obrnute veze u pomoćnim poljima, a određuje se i novi početak liste. U ovoj fazi polja `next` **ne smeju da se menjaju**. Tek kada sve niti prve faze završe rad, u drugoj paralelnoj fazi pomoćne veze prepisuju se u polja `next`.

Data pomoćna funkcija `parallelFor(n, p, action)` deli indekse $0, \dots, n-1$ na p disjunktnih intervala i pokreće po jednu nit za svaki interval. Svaka nit poziva `action(i)` za indekse iz svog intervala.

Parametar `action` predstavlja prosleđenu funkciju koja opisuje obradu jednog indeksa.

Dopuniti označene delove koda:

1. **[A] Priprema — 3 poena.** Stari početak postaje kraj novoformirane liste.
2. **[B] Prva paralelna faza — 4 poena.** Za čvor sa indeksom i formirati odgovarajuću obrnutu vezu u pomoćnom polju. U istoj fazi odrediti novi početak liste i upisati njegov indeks u `newHead`.
3. **[C] Druga paralelna faza — 3 poena.** Zameniti staru vezu svakog čvora odgovarajućom pomoćnom vezom.

Nije potrebno implementirati glavni program. Ne menjati neoznačene delove koda niti zamenjivati paralelne faze sekvencijalnim obrtanjem liste.

Primer. Za vektor

```
//                value, next
std::vector<Node> a = {{30, 4}, {10, 3}, {50, -1},
                      {20, 0}, {40, 2}};

int head = 1;
```

redosled indeksa u početnoj listi je

$$1 \rightarrow 3 \rightarrow 0 \rightarrow 4 \rightarrow 2.$$

Posle obrtanja redosled indeksa treba da bude

$$2 \rightarrow 4 \rightarrow 0 \rightarrow 3 \rightarrow 1,$$

a funkcija treba da vrati 2. Vrednosti **value** i položaji čvorova u vektoru ostaju nepromenjeni.

Označavanje S-AVL čvorova

Dato je binarno stablo. Čvor je **S-AVL čvor** ako **nije list** i ako su visine njegovog levog i desnog podstabla jednake. Visina praznog stabla je 0, a visina lista je 1. Ne pretpostavlja se da je dato stablo AVL stablo.

Potrebno je označiti sve S-AVL čvorove **čiji nijedan pravi potomak nije S-AVL čvor**. Pravi potomak je bilo koji čvor ispod posmatranog čvora, ne samo njegovo neposredno dete.

Za svaki čvor polje **oznaka** treba postaviti na **true** ako ispunjava traženi uslov, a na **false** u suprotnom, bez obzira na prethodnu vrednost tog polja. Nije dozvoljeno menjati ključeve ni strukturu stabla.

Dopuniti datu rekurzivnu funkciju

```
int oznaciSAVL(Node* x);
```

koja, pored označavanja čvorova u podstablu sa korenom **x**, vraća **visinu sa predznakom**:

- 0, ako je podstablo prazno;
- $+h$, ako je neprazno podstablo visine h i ne sadrži nijedan S-AVL čvor;
- $-h$, ako je podstablo visine h i sadrži bar jedan S-AVL čvor, uključujući i svoj koren.

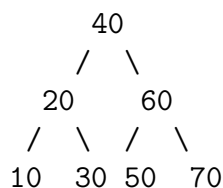
Negativan rezultat ne znači da je koren tog podstabla označen: S-AVL čvor se može nalaziti i dublje u podstablu.

Ukupno vreme izvršavanja mora biti $O(n)$, gde je n broj čvorova stabla. Visine se ne smeju iznova računati posebnim obilascima podstabala. Nije potrebno pisati glavni program.

Dopune i bodovanje:

1. **[A] — 2 poena.** Obraditi prazno stablo. Za neprazno stablo poništiti oznaku tekućeg čvora.
2. **[B] — 2 poena.** Rekurzivno obraditi oba podstabla i njihove rezultate sačuvati u promenljivama **l** i **d**.
3. **[C] — 3 poena.** Kada su oba rezultata pozitivna, proveriti da li treba označiti tekući čvor i vratiti odgovarajuću visinu sa predznakom.
4. **[D] — 2 poena.** Kada je bar jedan rezultat negativan, izračunati i vratiti odgovarajuću visinu sa predznakom.
5. **[E] — 1 poen.** Vratiti odgovarajući rezultat u preostalom slučaju.

Primer. Za stablo



označavaju se čvorovi 20 i 60. Čvor 40 jeste S-AVL čvor, ali ima S-AVL potomke, pa se ne označava. Listovi nisu S-AVL čvorovi. Poziv za koren vraća -3 .

Randomize all around

Data je funkcija `rand100()` koja generiše slučajan ceo broj iz intervala $[1, 100]$ (uključivo), pri čemu su svi brojevi iz tog intervala jednako verovatni.

- (a) **(5 poena)** Potrebno je konstruisati funkciju koja generiše jednu od tri vrednosti x , y ili z tako da važe zadate verovatnoće $P(x)$, $P(y)$ i $P(z)$, gde je

$$P(x) + P(y) + P(z) = 1.$$

- (b) **(5 poena)** Uopštiti prethodni problem na slučaj kada je dato n različitih vrednosti $x_1, x_2, \dots, x_n$ sa odgovarajućim verovatnoćama $p_1, p_2, \dots, p_n$, gde važi

$$\sum_{i=1}^n p_i = 1.$$

Konstruisati algoritam koji koristeći funkciju `rand100()` generiše vrednost x_i sa verovatnoćom p_i .