OBLIKOVANJE I ANALIZA ALGORITAMA — popravni kolokvij

14. 2. 2018.

1. (10) Napišite precizne definicije sljedećih pojmova:

   (a) za funkcije f, $g : \mathbb{N}_0 \to \mathbb{N}_0$ vrijedi $f(n) \in O(g(n))$, kad $n \to \infty$,

   (b) funkcija f **eksponencijalno raste**.

   Neka su f, $g : \mathbb{N}_0 \to \mathbb{N}_0$ rastuće funkcije. Dana je sljedeća tvrdnja:

$$f(n) \in O(g(n)) \implies 2^{f(n)} \in O(2^{g(n)}).$$

   Ako je ova tvrdnja istinita, dokažite ju. U protivnom, nađite kontraprimjer i detaljno ga opravdajte.

2. (10) Zadana je rekurzivna funkcija za ispis stringova `":)"` (smiješak, engl. smiley):

```
void Smileys(int n) {
  if (n == 0)
    printf(":)");
  else
    for (int t = 1; t <= 2^n; t = t + 1)
      Smileys(n - 1);
  return;
}
```

   Nađite točan broj ispisanih smiješaka, ili nađite točan red veličine (relacijom Θ) za broj ispisanih smiješaka, u funkciji od n, uz pretpostavku da je $n \geq 0$.

3. (30) Zadana je **padajuća** funkcija $f : \mathbb{N} \to \mathbb{Z}$, koju možete pozvati kao funkciju f, s jednim argumentom $i \in \mathbb{N}$, a funkcija vraća vrijednost $f(i)$. Treba naći **najmanji** argument $n \in \mathbb{N}$, za kojeg je $f(n) < 0$, tj. $n = \min\{i \in \mathbb{N} \mid f(i) < 0\}$, ako takav argument n postoji. Složenost algoritma mjerimo **brojem** poziva funkcije f.

   (a) Pretpostavljamo da je funkcija f **strogo** padajuća, tj. da je $f(i) > f(i+1)$, za svaki $i \in \mathbb{N}$. Pokažite da traženi n sigurno postoji. Sastavite što efikasniji algoritam za nalaženje n.

   (b) Pretpostavljamo da je funkcija f padajuća, ali ne mora biti strogo padajuća, već samo znamo da **postoji** $n_0 \in \mathbb{N}$ (kojeg **ne znamo**), takav da je $f(n_0) < 0$. Sastavite što efikasniji algoritam za nalaženje traženog n.

   U oba slučaja, složenost algoritma **mora** biti u $O(\log n)$. Dajte kratku (ali preciznu) argumentaciju da je složenost algoritma zaista u $O(\log n)$.

OKRENITE!

4\. (35) Student u semestru ima n domaćih zadaća i sve su objavljene na kraju dana s rednim brojem 0 (početak brojanja dana). Rješenje bilo koje zadaće troši cijeli dan, tako da student može riješiti najviše **jednu** zadaću po danu. Zadaća i ima zadani rok od T_i dana i broj bodova B_i (prirodni brojevi), za $i = 1, \ldots, n$. Student dobiva B_i bodova ako preda i-tu zadaću do kraja roka T_i (tj. najkasnije na kraju tog dana), a u protivnom ne dobiva nikakve bodove, jer je zakasnio. Student treba napraviti plan rješavanja zadaća, tako da dobije **najveći** mogući broj bodova. Može se dogoditi da postoje planovi u kojima nije moguće sve zadaće predati na vrijeme.

Plan rješavanja prikazujemo poljem dan s n elemenata, s tim da $dan[i] = j$ znači da i-tu zadaću treba riješiti (i predati) na dan j. Ako se i-tu zadaću **ne isplati** rješavati, onda stavljamo $dan[i] = -1$.

Pokažite primjerom (s najviše 3 zadaće) da sljedeće dvije pohlepne strategije (a) i (b) **ne moraju** dati optimalni plan.

(a) Među neraspoređenim zadaćama koje još mogu biti napravljene na vrijeme, uzmi onu s najranijim rokom. Ako takvih ima više, uzmi onu s najviše bodova (ako i takvih ima više, uzmi bilo koju od njih). Izabranu zadaću rasporedi na najraniji slobodni dan. Ponavljaj ovaj postupak.

(b) Među neraspoređenim zadaćama koje još mogu biti napravljene na vrijeme, uzmi onu s najvećim bodovima. Ako takvih ima više, uzmi onu s najranijim rokom (ako i takvih ima više, uzmi bilo koju od njih). Izabranu zadaću rasporedi na najraniji slobodni dan. Ponavljaj ovaj postupak.

(c) Nađite **optimalnu** pohlepnu strategiju za raspoređivanje zadaća i opišite ju **riječima**, kao u (a) i (b). Dokažite optimalnost te strategije.

(d) Sastavite algoritam koji nalazi optimalni plan rješavanja zadaća. Treba vratiti polje dan, i broj k domaćih zadaća koje mogu biti predane na vrijeme (tako da donose bodove). Analizirajte složenost tog algoritma u ovisnosti o n.

5\. (35) Ukratko opišite što je struktura **disjunktnih skupova** i koje osnovne **operacije** su definirane na toj strukturi (**što** rade te operacije, nije bitno **kako**).

(a) Što je polazno stanje strukture i kako mjerimo složenost osnovnih operacija?

(b) Opišite neku reprezentaciju strukture disjunktnih skupova i pripadne **efikasne** implementacije osnovnih operacija. Komentirajte njihovu složenost.

(c) Skicirajte Kruskalov algoritam za nalaženje minimalnog razapinjućeg stabla (MST) zadanog (povezanog) neusmjerenog grafa $G = (V, E)$. Što se događa u tom algoritmu ako graf nije povezan? Kako se algoritam zaustavlja i što dobijemo kao rezultat?

(d) Opišite kako se koristi struktura disjunktnih skupova u Kruskalovom algoritmu. Uz pretpostavku da je graf povezan, koliko osnovnih operacija svake vrste je potrebno u takvoj implementaciji Kruskalovog algoritma?

REZULTATI: četvrtak, 15. 2. 2018., navečer (na webu),
petak, 16. 2. 2018. u 12 sati.

Saša Singer