OBLIKOVANJE I ANALIZA ALGORITAMA — 2. kolokvij

4. 7. 2008.

1. (20) Neka je $A_n = \{a_1, a_2, \ldots, a_n\}$ konačan skup različitih vrsta novčića. Vrijednosti novčića a_i su prirodni brojevi i vrijedi $a_1 > a_2 > \ldots > a_n$. Na raspolaganju imamo neograničenu količinu novčića svake vrste. Problem razmjene novca glasi: treba izabrati **najmanji** ukupni broj novčića tako da zbroj njihovih vrijednosti bude zadani prirodni broj C.

   (a) Ako je $a_n > 1$, pokažite primjerom da postoje A_n i C za koje problem nema rješenja.

   (b) Ako je $a_n = 1$, dokažite da rješenje postoji.

   (c) Za slučaj $a_n = 1$, sastavite pohlepni algoritam razmjene novca, koji prolazi novčiće redoslijedom $a_1, a_2, \ldots, a_n$, uzimajući maksimalni broj novčića određene vrijednosti a_i. Algoritam treba vratiti brojeve x_i novčića i–te vrste u razmjeni, za $i = 1, \ldots, n$. Nađite složenost ovog algoritma. Ovisi li ona o vrijednostima novčića a_i?

   (d) Nalazi li pohlepni algoritam **uvijek** rješenje s najmanjim brojem novčića? Posebno, što vrijedi za slučaj $a_i = k^{n-i}$, $i = 1, \ldots, n$, gdje je $k > 1$ zadani prirodni broj? Dokažite ili nađite kontraprimjer.

2. (20) Neka je p polinom stupnja $n = 2^k - 1$ s cjelobrojnim koeficijentima i vodećim koeficijentom (uz x^n) jednakim 1. Polinom p možemo napisati u obliku

$$p(x) = \left(x^{(n+1)/2} + a\right) q(x) + r(x) \quad ,$$

gdje je a konstanta, a q i r su polinomi stupnja $2^{k-1} - 1$. Isti postupak rekurzivno primijenimo na polinome q i r. Na kraju, dobivamo polinom p izražen u terminima polinoma oblika $x^i + c_i$, gdje je i potencija od 2. Dobiveni oblik polinoma p zovemo **pripremljeni** oblik.

   (a) Izrazite polinom $p(x) = x^7 - 5x^6 + 4x^5 - 13x^4 + 3x^3 - 10x^2 + 5x - 17$ u pripremljenom obliku.

   (b) Neka je p zadan u pripremljenom obliku. Izaberite pogodnu strukturu podataka za ovaj prikaz polinoma. Sastavite algoritam koji računa vrijednost $p(x)$, za zadani $x \in \mathbf{Z}$. Nađite točan broj množenja i točan broj zbrajanja u tom algoritmu. Usporedite rezultate s brojem operacija u Hornerovom algoritmu. Kolika je ušteda?

(a): $(x^4+a)\cdot(x^3+q_2x^2+q_1x+q_0)+(x^3+r_2x^2+r_1x+r_0)=p(x)$

$\Rightarrow q_2=-5,\ q_1=4,\ q_0=-13,\ \underline{\underline{a=2}},\ r_2=0,\ r_1=-3,\ r_0=9$

⋮

OKRENITE!

$p(x) = (x^4+2)\cdot\left[(x^2+3)(x-5)+(x+2)\right] + \left[(x^2-4)\cdot x+(x+9)\right]$

4\. (35) Tvrtka za sigurnost treba kupiti licence za razne dijelove kriptografskih programa. Zbog važećih propisa, tvrtka može kupiti najviše jednu licencu mjesečno. Svaka licenca trenutno (na početku cijele kupovine) košta 1000 kn. Međutim, svaki sljedeći mjesec, svaka licenca postaje sve skuplja: cijena licence j raste za faktor $r_j > 1$. To znači da licenca j, kupljena nakon t mjeseci od početka, košta $1000 \cdot r_j^t$ kn. Tvrtka mora kupiti n licenci i treba naći plan kupovine koji **minimizira** ukupnu cijenu, s tim da su zadani faktori rasta cijene $r_1, \ldots, r_n$ za svaku licencu. Pretpostavimo da su svi faktori rasta međusobno **različiti**, tj. vrijedi $r_i \neq r_j$, za $i \neq j$.

(a) Nađite redoslijed u kojem treba kupovati licence, tako da ukupna cijena bude **minimalna**. Dokažite optimalnost tog redoslijeda. Uputa: analizirajte slučaj $n = 2$.

(b) Sastavite algoritam koji nalazi optimalni redoslijed kupovine i nađite njegovu složenost. Ulazni argumenti su broj n i polje faktora rasta r, a izlaz je permutacija brojeva od 1 do n, koja sadrži optimalni redosljed kupovine, i pripadna minimalna cijena c. Složenost algoritma mora biti $O(n^2)$.

5\. (25) Definirajte što je problem nalaženja minimalnog razapinjućeg stabla (MST) neusmjerenog povezanog grafa.

(a) Opišite ukratko Kruskalov algoritam za nalaženje MST i objasnite što bi se dogodilo da graf nije povezan. Za opis algoritma smijete koristiti osnovne operacije na strukturi disjunktnih skupova.

(b) Nad kojim objektima se koristi struktura disjunktnih skupova u Kruskalovom algoritmu? Koje su osnovne operacije definirane na toj strukturi (**što** rade te operacije, nije bitno **kako**)?

(c) Kako mjerimo složenost tih operacija? Navedite neku implementaciju tih operacija i komentirajte njihovu složenost.

REZULTATI: srijeda, 10. 2. 2010. u 11:30 sati. Saša Singer

K3 – 8.2.2016

Zad 4 – kriptografske licence (opt. lic. buying)
KT, Chap. 4, Ex 2, pp. 185–187

faktori rasta cijene $r_j > 1$. ($r_i \neq r_j$ za $i \neq j$)

$n=2$: prvo kupimo r_1, pa r_2 (u mjesecima $t, t+1$)
cijena $r_1^t + r_2^{t+1}$
– obratni red
cijena $r_1^{t+1} + r_2^t$

Usporedba <: $r_1^t + r_2^{t+1} \overset{?}{<} r_1^{t+1} + r_2^t$

$r_2^t(r_2-1) < r_1^t(r_1-1)$

$\Rightarrow \quad r_2 < r_1$

$\Rightarrow$ Sortirati r_j u padajućem poretku $r_1 > r_2 > \ldots > r_n$.

OBLIKOVANJE I ANALIZA ALGORITAMA — 2. kolokvij

24. 1. 2011.

Interval sched., max. compatible subset, KT, Chap 4, 116-121

1. (25) Astronomski laboratorij treba napraviti raspored zahtjeva za korištenjem novog teleskopa "Velike Okice". U svakom trenutku, teleskop može koristiti samo jedan korisnik. Prikupljeno je ukupno n zahtjeva, a svaki zahtjev ima oblik vremenskog intervala $[p_i, k_i]$, gdje je p_i početak, a k_i kraj korištenja. Pretpostavljamo da su intervali korektno zadani, tj. da je $p_i < k_i$, za $i = 1, \ldots, n$. Naravno, razni intervali se mogu međusobno preklapati, tako da se može dogoditi da nije moguće zadovoljiti sve zahtjeve. Uprava laboratorija želi da u probnom pogonu što više korisnika upozna mogućnosti nove opreme, bez obzira na iskoristivost i ukupno vrijeme korištenja. Zato, od pristiglih n zahtjeva, treba izabrati podskup s najvećim **brojem** zahtjeva koje je moguće zadovoljiti.

Zahtjevi i, j su **kompatibilni** ako se traženi intervali ne preklapaju, tj. ako vrijedi $k_i \leq p_j$ ili $k_j \leq p_i$. Podskup zahtjeva A je **kompatibilan** ako za svaki par zahtjeva $i, j \in A$, uz $i \neq j$, vrijedi da su kompatibilni. Najveći kompatibilni podskup zahtjeva zovemo **optimalnim**. Sastavite **pohlepni** algoritam koji nalazi optimalni podskup zahtjeva, dokažite korektnost algoritma i nađite njegovu složenost.

(a) Pokažite primjerom da pohlepa po kriteriju "izaberi dozvoljeni interval koji najranije počinje", tj. uzlazno po vremenu početka p_i, ne mora biti optimalna.

(b) Pokažite primjerom da pohlepa po kriteriju "izaberi dozvoljeni interval koji najkraće traje", tj. uzlazno po vremenu trajanja $k_i - p_i$, ne mora biti optimalna.

(c) Nađite "pravi" kriterij za pohlepu i dokažite njegovu optimalnost, tj. da ponovljenom primjenom tog kriterija dobivamo kompatibilni podskup s najvećim brojem intervala.

(d) Sastavite algoritam koji nalazi optimalni podskup i nađite njegovu složenost.

(c) izaberi zahtjev koji najranije završava — najmanji k_i
$\Rightarrow$ ostavlja najviše vremena za ostale

P = skup svih pristiglih
$A = \emptyset$ — skup prihvaćenih
sve dok je $P \neq \emptyset$
  izaberi zahtjev i iz P s min k_i
  dodaj ga u A, izbaci iz P
  izbaci sve zahtjeve iz P koji nisu kompatibilni sa zahtjevom i.

OKRENITE!

OBLIKOVANJE I ANALIZA ALGORITAMA — 2. kolokvij

16. 1. 2012.

1. (25) Natjecanje u trijatlonu ima tri dijela: plivanje u malom bazenu na 800 m, zatim vožnju biciklom na 10 km i, na kraju, trčanje na 3 km. Za svakog pojedinog natjecatelja nema odmora između ovih dijelova. Zbog sigurnosnih propisa, u bazenu se u bilo kojem trenutku smije nalaziti samo **jedan** natjecatelj. Na biciklističkoj i trkačoj stazi nema takvog ograničenja, tj. više natjecatelja može istovremeno biti na svakoj od tih staza.

   Umjesto skupnog starta, natjecatelji startaju jedan za drugim — čim prethodni natjecatelj izađe iz bazena, tog trena starta sljedeći natjecatelj, i tako redom, pa se bazen koristi bez prekida.

   Ukupno je prijavljeno n natjecatelja, a prijava i-tog natjecatelja sadrži očekivana vremena za svaki od tri dijela utrke: p_i za plivanje, b_i za vožnju bicikla i t_i za trčanje.

   Za minimizaciju troškova pratnje, organizatori natjecanja žele naći **redosljed** kojim trebaju startati natjecatelji tako da **cijela** utrka — od starta prvog natjecatelja, do trena kad svi natjecatelji završe trčanje, ima **najkraće** očekivano trajanje (izračunato na osnovu očekivanih vremena iz prijava).

   (a) Analizirajte slučaj $n = 2$ i nađite **pohlepni** kriterij koji daje najmanje trajanje cijele utrke. Dokažite optimalnost tog kriterija za bilo koji n.

   (b) Sastavite algoritam koji nalazi optimalni startni redosljed i nađite njegovu složenost. Ulazni argumenti su broj n i polja očekivanih vremena p, b i t. Izlaz algoritma je permutacija brojeva od 1 do n, koja sadrži optimalni startni redosljed, i pripadno najmanje očekivano trajanje cijele utrke T. Složenost algoritma mora biti $O(n^2)$.

2. (20) 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 disjunkntih skupova i pripadne **efikasne** implementacije osnovnih operacija. Komentirajte njihovu složenost.

OKRENITE!

p_i = plivanje i (projected – prijavlj. vrijeme)
b_i = bicikl
t_i = trčanje $\qquad i=1,\dots,n$

– Min. vrijeme završ. uz ogr. jednog plivača u bazenu
predv.

– Gledamo 2 natjecatelja: $p_1, b_1, t_1 \quad p_2, b_2, t_2 \ (>0)$

poredak (1,2): 1. završ. u $p_1+b_1+t_1$
2. završ. u $p_1+p_2+b_2+t_2$

traj. je <u>max</u> od ova 2 broja

$$T_{1,2} = \max\{\underbrace{p_1+b_1+t_1}_{T_1}, \underbrace{p_1+p_2+b_2+t_2}_{p_1+T_2}\}$$

obratno: (2,1):

$$T_{2,1} = \max\{\underbrace{p_2+p_1+b_1+t_1}_{p_2+T_1}, \underbrace{p_2+b_2+t_2}_{T_2}\}$$

(ovome se još može dodati vrijeme čekanja zbog ranijih plivača, no to ne mijenja odluku za ova 2)

Kad je $T_{1,2} \le T_{2,1}$?

ako je to: $\Rightarrow$

ako $T_1 \ge p_1+T_2$:
tj. $T_{1,2}=T_1$ $\quad\Rightarrow\quad T_{1,2}\le T_{2,1} \quad\Rightarrow\quad T_1 \ge p_1+T_2 \ge T_2$

$T_{2,1} = p_2+T_1 > T_2$

$\Rightarrow \quad p_2+T_1 \ge T_1 \ge p_1+T_2 \ge T_2$

<u>$b_1+t_1 \ge b_2+t_2$</u>

ako je $T_1 \le p_1+T_2$ $\qquad T_1 \le p_1+T_2$ &
$T_{1,2}=p_1+T_2$ $\qquad T_{1,2}\le T_{2,1} \Rightarrow (p_1+T_2 > T_2) \Rightarrow \quad T_{2,1}=p_2+T_1$

i $\quad p_1+T_2 \le p_2+T_1 \Rightarrow b_2+t_2 \le b_1+t_1$

(1) $T_1 \geq p_1 + T_2$ tj. $T_{1,2} = T_1$

onda $T_1 \geq p_1 + T_2 > T_2$ i $p_2 + T_1 \geq T_1 \geq p_1 + T_2 > T_2$

$\Rightarrow T_{2,1} = p_2 + T_1 > T_{1,2} \Rightarrow \underline{\min = T_{1,2}}$

Analogno:

(2) $T_2 \geq p_2 + T_1$ tj. $T_{2,1} = T_2$

onda: $p_1 + T_2 > T_2 \geq p_2 + T_1 > T_1$

$\Rightarrow T_{1,2} = p_1 + T_2 > T_{2,1} \Rightarrow \underline{\min = T_{2,1}}$

(3) $T_1 < p_1 + T_2$ tj. $T_{1,2} = p_1 + T_2$

(znamo $T_2 < p_1 + T_2$)

(3a) ako je $p_2 + T_1 \leq T_2$ onda $T_{2,1} = T_2 < p_1 + T_2 = T_{1,2}$

tj. $\underline{\min = T_{2,1}}$

ako je $p_2 + T_1 > T_2$ onda $T_{2,1} = p_2 + T_1$

pa sve ovisi o odnosu $p_2 + T_1$ i $p_1 + T_2$

$p_1 + T_2 \leq p_2 + T_1 \Rightarrow \min = T_{1,2}$

$p_1 + T_2 > p_2 + T_1 \Rightarrow \min = T_{2,1}$

tj. $b_1 + t_1$ i $b_2 + t_2$

npr: $b_1 + t_1 \leq b_2 + t_2 \Rightarrow T_{2,1} \leq T_{1,2}$ (brži $b+t$ kasnije)

$b_2 + t_2 \leq b_1 + t_1 \Rightarrow T_{1,2} \leq T_{2,1}$

OBLIKOVANJE I ANALIZA ALGORITAMA — popravni kolokvij

30. 1. 2012.

1. (10) Neka su $f, g : \mathbf{N} \to \mathbf{N}$ dvije funkcije. Napišite preciznu definiciju asimptotske relacije ponašanja

$$f(n) \in o(g(n)) \quad (n \to \infty).$$

def = 5

svojstva = 5 Koja svojstva ima relacija o (u smislu relacije uređaja)?

2. (10) Između ponuđenih odgovora

$$\Theta(1), \quad \Theta(\lg n), \quad \Theta(n), \quad \Theta(n \lg n), \quad \Theta(n^2), \quad \Theta(n^2 \lg n), \quad \Theta(n^3), \quad \Theta(2^n),$$

nađite točan red veličine za broj koliko puta se izvršava naredba `x = x + 1` u svakom od sljedećih dijelova programa (/ je operator cjelobrojnog dijeljenja, kao u C-u):

odg = 2

arg = 3

(a)
```
i = n;
while (i >= 1) {
  for j = 1 to n
    x = x + 1;
  i = i / 3;
}
```
$\Theta(n \cdot \lg n)$

(b)
```
i = 1;
while (i <= n) {
  for j = 1 to i
    for k = 1 to n
      x = x + 1;
  i = i * 4;
}
```
$\Theta(n^2)$

Ukratko **argumentirajte** odgovore!

3. (35) Investicijska tvrtka planira buduća ulaganja na osnovu analize ponašanja dionica na tržištu u prethodnom razdoblju. U postupku analize često treba riješiti zadaće sljedećeg oblika. Promatra se ponašanje cijene određene dionice u bloku od n uzastopnih dana u prošlosti. Za svaki pojedini dan $i = 1, \ldots, n$, poznata je cijena c_i jedne dionice za taj dan (pretpostavimo da je cijena c_i konstantna kroz cijeli dan i).

Uzmimo da tvrtka u tom periodu želi **kupiti** 1000 dionica u nekom danu k, a zatim **prodati** sve te dionice u nekom **kasnijem** danu p. Na osnovu poznatih podataka treba naći koji dan je trebalo kupiti dionice i kada ih je trebalo prodati da zarada tvrtke bude **najveća** moguća. Ako u tom periodu od n dana **nije** bilo moguće ostvariti ikakvu zaradu, onda, umjesto traženih dana k i p, treba vratiti $k = p = 0$.

Na primjer, neka je $n = 3$, a poznate cijene jedne dionice kroz ta tri dana su $c_1 = 9$, $c_2 = 1$ i $c_3 = 5$ novčanih jedinica. Onda treba vratiti $k = 2$ (kupi na dan 2) i $p = 3$ (prodaj na dan 3), jer to daje zaradu od 4 novčane jedinice po svakoj dionici.

Sastavite algoritam koji, za zadani n i polje cijena c, nalazi optimalne dane k i p, ako oni postoje, odnosno, vraća $k = p = 0$, ako zarada nije moguća.

Red veličine složenosti algoritma mora biti $O(n \log n)$. Analizirajte složenost vašeg algoritma i pokažite da ona zadovoljava ovaj uvjet. Uputa: iskoristite princip "podijeli, pa vladaj" na polovinama zadanog vremenskog perioda (do $n/2$ dana i nakon toga).

najbolje = max $\left\{ \begin{array}{l} \text{najbolje u prvom } (k < p \leq \frac{n}{2}), \\ \text{najbolje u drugom } (\frac{n}{2} < k < p \leq n) \end{array} \right.$

OKRENITE!

i najbolje za $k \leq \frac{n}{2}$, $p > \frac{n}{2}$, a to je $\max c(p)$ za $p > \frac{n}{2}$ $- \min c(k)$ za $k \leq \frac{n}{2}$.

OBLIKOVANJE I ANALIZA ALGORITAMA — 2. kolokvij

9. 1. 2013.

1. (20) Mala kopiraonica ima samo jedan kopirni aparat i želi optimizirati zadovoljstvo svojih dnevnih korisnika. Svako jutro, pri otvaranju, poznat je skup od n poslova koje treba obaviti taj dan. Posao za i-tog korisnika troši t_i vremena kopirnog aparata. Osim toga, svaki korisnik ima poznatu težinu w_i, koja predstavlja važnost tog korisnika za cjelokupni posao.

Zadovoljstvo svakog korisnika ovisi o trenutku **završetka** njegovog posla — "što prije, to bolje". Za bilo koji poredak poslova, neka C_i označava trenutak završetka posla za i-tog korisnika. Na primjer, ako je posao za i-tog korisnika obavljen kao **prvi** po redu, onda je $C_i = t_i$, a ako je posao za i-tog korisnika obavljen **odmah** iza posla za j-tog korisnika, onda je $C_i = C_j + t_i$, tj. aparat radi bez prestanka i vremena se zbrajaju.

Kopiraonica želi **poredati** poslove tako da **minimizira** težinsku sumu vremena završetka poslova

$$S = \sum_{i=1}^{n} w_i C_i.$$

Na primjer, neka je $n = 2$. Posao za prvog korisnika traje $t_1 = 1$, s težinom $w_1 = 10$, a posao za drugog korisnika traje $t_2 = 3$, s težinom $w_2 = 2$. Ako posao za prvog korisnika obavimo ranije, onda je pripadna težinska suma vremena završetka jednaka $10 \cdot 1 + 2 \cdot 4 = 18$, dok u obratnom poretku dobivamo veću težinsku sumu vremena završetka $10 \cdot 4 + 2 \cdot 3 = 46$. $\boxed{\text{UZLAZNO po } t_i/w_i}$ ili SILAZNO po $\frac{w_i}{t_i}$ = w_i. po jedinici vremena!

Kriterij BEZ DOKAZA = 3

Dz. za n=2 (5)

Dz. opt (5)

Alg = (8)

(a) Analizirajte slučaj $n = 2$ i nađite **pohlepni** kriterij koji daje najmanju težinsku sumu S. Dokažite **optimalnost** tog kriterija za bilo koji n.

(b) Sastavite algoritam koji nalazi optimalni poredak poslova i nađite njegovu složenost. Ulazni argumenti su broj n i polja t, w. Izlaz algoritma su permutacija p brojeva od 1 do n, koja sadrži optimalni redoslijed poslova ($p[j] = i$ znači da posao za i-tog korisnika treba obaviti kao j-ti po redu), i pripadna najmanja težinska suma S. Složenost algoritma mora biti $O(n^2)$.

Alg. - korektan, za pogrešni kriterij (5)

OKRENITE!

2\. Svaka od n osoba zna točno **jednu** glasinu, različitu od svih ostalih glasina. Cijela
(30) skupina želi što brže podijeliti "novosti" sa svim ostalim osobama, međusobnim slanjem poruka. Svaka pojedina poruka sadrži **sve** glasine koje pošiljatelj zna u trenutku slanja poruke i ima samo **jednog** primatelja. Slanje jedne poruke osobe i osobi j zapisujemo kao poziv funkcije $poruka(i,\ j)$. Zbog proizvoljne numeracije osoba, uzmite da prvu poruku šalje osoba 1, a zadnju šalje osoba n. Najmanji broj $= 2n-2$

rješ. $2n-2$ (3)
dokaz (7)
alg (može i ispod) (5)

(a) Nađite **najmanji** ukupni broj poruka koje ove osobe trebaju poslati, tako da bude sigurno da je **svaka** osoba saznala **sve** glasine. Dokažite da je to najmanji mogući broj poruka koji garantira "potpuno širenje" glasina. Napišite neki algoritam — redoslijed slanja poruka, koji to realizira u najmanjem broju poruka.

U nastavku zadatka tražimo posebne algoritme za "potpuno širenje" glasina, koji imaju **najmanji** ukupni broj poruka i **još** zadovoljavaju dodatna ograničenja.

dokaz (4-5)
alg. 5-6

(b) Pretpostavimo da svaka glasina ima istu "duljinu" u poruci — na primjer, jednaku 1. Napišite algoritam koji ima **najmanji** ukupni **zbroj** duljina svih poruka i dokažite njegovu optimalnost.

dokaz (4-5)
alg. 5-6

(c) Svakom poslanom porukom želimo **maksimalno povećati** ukupan **zbroj** poznatih glasina po svim osobama. Na početku, ovaj zbroj je n — svaka osoba zna jednu glasinu, a na kraju je n^2 — svaka osoba zna svih n glasina. Napišite "pohlepni" algoritam koji to realizira i dokažite njegovu optimalnost.

jedan alg. $\equiv$ (5-6)
dva ili tri alg = (10-12)

Napomena: Ako ste pod (b) ili (c) napisali odgovarajući algoritam, nije potrebno posebno pisati neki algoritam pod (a). (dokazi ukupno 7 + 4 + 4 = 15
rješ $2n-2$ = 3, alg = 6 + 6)

3\. Zadano je polje od n objekata. Te objekte možemo uspređivati **binarnom** uređaj-
(20) nom relacijom $\leq$. Definirajte što je problem sortiranja takvog polja i kako mjerimo složenost.

prob. (2)
slož. (2)

(4) (a) Opišite ukratko Quicksort algoritam za sortiranje polja.

(4) (b) Napišite **rekurziju** za složenost (ili neku mjeru složenosti) Quicksort algoritma. Što je rješenje te rekurzije u **najgorem** slučaju i kad se to događa, tj. kako tad izgleda polazno polje?

(4) (c) Ukratko opišite uz koje pretpostavke se izvodi **prosječna** složenost i koji rezultat se dobiva tom analizom.

(4) (d) Kolika je **donja** ograda složenosti za **bilo koji** algoritam sortiranja polja koji koristi samo binarnu operaciju uspoređivanja elemenata u polju? Ukratko argumentirajte kako dolazimo do te ograde.

REZULTATI: petak, 11. 1. 2013. u 12 sati. Saša Singer

1 Permutacija p za $\sum_{i=1}^{n} w_i C_i \to \min$

(a) $n=2$.

Ako je $p[1]=1$, $p[2]=2$, tj. prvo obavimo prvi, pa onda drugi posao, onda je $C_1=t_1$, $C_2=C_1+t_2=t_1+t_2$, pa je

$$S_{1,2}=w_1\cdot t_1+w_2\cdot(t_1+t_2)$$

Ako je $p[1]=2$, $p[2]=1$, tj. prvo obavimo drugi, pa onda prvi posao, onda je $C_1=C_2+t_1$, $C_2=t_2$ tj. $C_1=t_1+t_2$, pa je

$$S_{2,1}=w_1\cdot(t_1+t_2)+w_2\cdot t_2$$

Ako želimo da je prvi poredak bolji, što je ekvivalentno s $S_{1,2}\le S_{2,1}$ ili

$$w_1 t_1+w_2\cdot(t_1+t_2)\le w_1(t_1+t_2)+w_2 t_2$$
$$\cancel{w_1 t_1}+w_2\cdot t_1+\cancel{w_2\cdot t_2}\le \cancel{w_1 t_1}+w_1\cdot t_2+\cancel{w_2\cdot t_2}$$
$$w_2\cdot t_1\le w_1\cdot t_2$$
$$\frac{t_1}{w_1}\le\frac{t_2}{w_2}$$

Dakle, poslove treba poredati UZLAZNO po $\frac{t_i}{w_i}$

2 - Duljina poruka stalno raste ⇒ ako svi žele saznati svih preostalih $n-1$ glasina, moraju primiti poruku barem duljine $n-1$ (kad su različite). Stvarno, zadnja osoba koja šalje – prima $n-1$ i dodaje svoju glasinu, a šalje poruku duljine n svim ostalima ("broadcast" na $n-1$ adresa) ⇒ zadnjih $n-1$ poruka (svaka je duljine n).

- Svaka osoba od preostalih $n-1$ (tj. od 1 do $n-1$) mora poslati svoju glasinu nekoj od preostalih, tako da "dođu" do osobe n (broadcastera). To je minimalno $n-1$ poruka u početnoj fazi (mogu biti raznih duljina – ovisi o poretku).

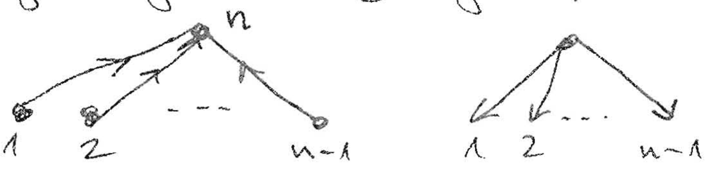
⇒ Min. broj $\geq 2n-2 = 2\cdot(n-1) = mbp(n)$

Dz. za jednako: $n=1$, $mbp(1)=0$. Korak indukcije: dodamo jednu osobu. Ona nekome mora poslati svoju (1 poruka) i od nekog primiti sve (1 poruka), tj.

$$mbp(n)\geq mbp(n-1)+2.$$

Alg 1: Svi šalju n-tom svoju (dulj $=1$) : for $i=1$ to $n-1$ do poruka(i,n)

(b) Zadnji šalje svima (dulj $=n$) : for $i=1$ to $n-1$ do poruka(n,i)



Broj poruka $=2n-2$

$\Rightarrow mbp(n)=2n-2$.

Broadcaster = prva osoba koja zna svih n glasina mora poslati $n-1$ poruka duljine n ostalima.

Dakle, duljina drugog dijela je $n \cdot (n-1)$.

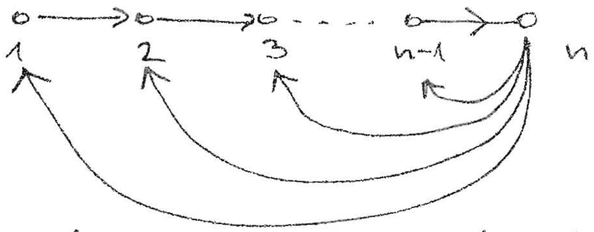
Najkraći prvi dio: svi šalju samo svoju (dulj. 1) zadnjem.

tj. dulj. prvog dijela je $1 \cdot (n-1)$.

Zbroj je $n^2 - 1 = (n+1)(n-1)$.

(c) Alg. je: for $i=1$ to $n-1$ do poruka$(i, i+1)$

for $i=1$ to $n-1$ do poruka(n, i) ← baš tim redom (prvi sazna $n-1$ novu!)



U prvom dijelu – dulj. poruka kreće od 1 i može narasti najviše za 1.

Greedy – svatko prima u danom trenu poruku najveće moguće duljine i sve što prima je <u>novo</u> za njega $\Rightarrow$ duljina mora rasti točno za 1 $\Rightarrow$

$\Rightarrow$ Prva faza je <u>put</u> od 1 do n, bez ciklusa (ne može biti ciklusa, ako je $n-1$ poruka, ali mora doći SVE osobe).

4. (30) Poznata web-tražilica El Goog troši veliku količinu vremena svaki puta kad slaže svoj indeks stranica. Tvrtka ima na raspolaganju samo **jedno** super-računalo i gotovo neograničen broj osobnih računala.

Cjelokupno slaganje indeksa sastoji se od n poslova koji se mogu obaviti potpuno nezavisno jedan od drugog. Svaki pojedini posao P_i ima dva dijela: prvo treba obaviti **pripremu** na super-računalu i ta faza traje s_i jedinica vremena, a zatim se posao **završava** na nekom osobnom računalu i ta faza traje t_i jedinica vremena.

Pripremna faza je toliko složena da super-računalo može obavljati samo **jedan** posao u danom trenutku. Čim je taj dio posla gotov, posao se trenutno prebacuje na neko osobno računalo. Istog trena, bez prekida, počinje izvršavanje sljedećeg posla na super-računalu. Osobnih računala ima barem n, tako da se završna faza svih poslova može obaviti potpuno paralelno — svi poslovi se mogu istovremeno završavati na raznim osobnim računalima.

El Goog treba naći **redosljed** kojim treba izvršiti poslove na super-računalu, tako da **ukupno** trajanje slaganja cijelog indeksa — od početka prvog posla na super-računalu, do završetka zadnjeg posla na nekom osobnom računalu, bude **najkraće** moguće. SORT SILAZNO po t_i.

kriterij = 5
dz$_1$ = 5
dz$_2$ = 5
alg = 15

(a) Analizirajte slučaj $n = 2$ i nađite **pohlepni** kriterij koji daje najmanje trajanje slaganja cijelog indeksa. Dokažite optimalnost tog kriterija za bilo koji n.

(b) Sastavite algoritam koji nalazi optimalni redosljed poslova na super-računalu i nađite njegovu složenost. Ulazni argumenti su broj poslova n i polja vremena s, t. Izlaz algoritma su permutacija p brojeva od 1 do n, koja sadrži optimalni redosljed poslova ($p[j] = i$ znači da posao s indeksom i treba obaviti kao j-ti po redu), i pripadno najmanje trajanje slaganja indeksa. Složenost algoritma mora biti $O(n^2)$.

5. (35) Zadana su dva polinoma A i B s kompleksnim koeficijentima.

(10) samo slož = (2)

(a) Opišite osnovne korake **brzog** algoritma za računanje produkta $C = A \cdot B$ zadanih polinoma, korištenjem brze diskretne Fourierove transformacije (FFT) vektora čija duljina je potencija broja 2. Kolika je složenost tog algoritma?

(20)

(b) Ukratko opišite što radi diskretna Fourierova transformacija vektora duljine $n = 2^k$. Skicirajte rekurzivni algoritam za **brzu** diskretnu Fourierovu transformaciju (FFT) i izvedite njegovu složenost, uz pretpostavku sekvencijalnog izvršavanja operacija.

(5)

(c) Ukratko opišite kako se dobiva algoritam za inverznu Fourierovu transformaciju.

Napomena: odgovori na pojedine dijelove zadatka vrednuju se nezavisno, tj. ako ne znate odgovoriti na (a) dio, smijete odgovoriti na ostale dijelove.

REZULTATI: petak, 25. 1. 2013. u 12 sati. Saša Singer

$s_1 + t_1$

$\quad s_2 + t_2$

$s_1 + \max\{t_1, s_2 + t_2\}$

$s_1 + s_2 + \max\{t_1 - s_2, t_2\}$

Ako je $t_1 \geqslant t_2$ onda

$\max\{t_1 - s_2, t_2\} \leq t_1$

a može i $<$

$s_2 + t_2$

$\quad s_1 + t_1$

$s_2 + \max\{t_2, s_1 + t_1\}$

$s_2 + s_1 + \max\{t_2 - s_1, t_1\}$

$\max\{\underbrace{t_2 - s_1}_{\leq t_2 \leq t_1}, t_1\} = t_1$