OBLIKOVANJE I ANALIZA ALGORITAMA — 2. kolokvij

29. 1. 2020.

1. (25) Zadano je n poslova koje treba izvršiti. Posao P_i, za $i = 1, \ldots, n$, ima trajanje $t_i > 0$ i rok d_i do kojeg bi ga trebalo završiti (t_i i d_i su realni brojevi). Međutim, dozvoljeno je da posao završi nakon roka d_i. U tom slučaju, kažemo da posao **kasni**.

Sve poslove izvršava jedan izvršitelj. Radi jednostavnosti, pretpostavljamo da izvršitelj počinje raditi u trenutku $s = 0$. U danom trenutku, izvršitelj može raditi samo jedan posao. Kad jednom počne raditi neki posao, onda ga radi sve dok ga ne završi, tj. ako je posao P_i počeo u trenutku s_i, onda završava u trenutku $k_i = s_i + t_i$. U trenutku kad završi neki posao, izvršitelj može početi neki drugi posao. Ako posao kasni ($k_i > d_i$), onda je njegovo **kašnjenje** $\ell_i = k_i - d_i$. Ako posao završi na vrijeme ($k_i \leq d_i$), onda definiramo $\ell_i = 0$.

Treba naći raspored ili poredak izvršavanja svih poslova, tako da **maksimalno** kašnjenje $L = \max\{\ell_1, \ldots, \ell_n\}$ bude **najmanje** moguće. Algoritam treba vratiti permutaciju p koja opisuje pripadni poredak izvršavanja poslova (prvo $P_{p(1)}$, pa $P_{p(2)}$, i tako redom, do $P_{p(n)}$) i polje koje sadrži početke izvršavanja pojedinih poslova (u uzlaznom poretku). Ako optimalnih permutacija ima više, smijete uzeti bilo koju od njih.

Pokažite primjerom (s najviše 3 posla) da sljedeće dvije pohlepne strategije **ne** moraju dati najmanji L.

(a) Poredaj poslove uzlazno po trajanju t_i, tako da se poslovi koji kraće traju obave ranije.

(b) Poredaj poslove uzlazno po "labavosti" roka obzirom na trajanje, tj. po razlici $d_i - t_i$, tako da se poslovi s manjim "slobodnim" vremenom obave ranije.

(c) Nađite **optimalnu** pohlepnu strategiju za raspored poslova i dokažite optimalnost te strategije.

(d) Sastavite algoritam koji nalazi optimalni raspored poslova i analizirajte njegovu vremensku složenost u ovisnosti o n.

Napomena (i uputa): Uočite da vrijeme početka izvršavanja s može biti bilo koje i nema bitni utjecaj na optimalnost poretka. Zato nema pretpostavke $d_i \geq t_i$.

OKRENITE!

2. (20) Zadan je prirodni broj n i kvadratna matrica A, reda n, koja sadrži samo elemente iz skupa $\{0,1\}$. Kvadratnu podmatricu od A zovemo kvadratom u A. Treba naći najveći kvadrat u A koji sadrži samo jedinice. Algoritam treba vratiti red ℓ te podmatrice i indekse i, j elementa u njezinom desnom donjem kutu, tj. tražena podmatrica je $A[i-\ell+1:i,\ j-\ell+1:j]$. Ako takvih podmatrica s istim (najvećim) ℓ ima više, dovoljno je vratiti indekse i, j za neku od njih (svejedno koju). Indekse u matrici brojimo od 1. Primjerice, u matrici A reda 5,

$$A = \begin{bmatrix} 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 & 0 \\ 0 & 1 & 1 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 \end{bmatrix},$$

algoritam treba vratiti podatke za podmatricu (kvadrat) reda 3, s desnim donjim vrhom na poziciji $(3,4)$, tj. $\ell = 3$, $i = 3$, $j = 4$.

Neka je (x,y) pozicija nekog elementa u matrici A, gdje je $1 \leq x,y \leq n$. Definiramo funkciju $F(x,y)$ kao red najvećeg kvadrata u A, koji sadrži samo jedinice i desni donji vrh tog kvadrata je na poziciji (x,y). Ako je $A(x,y) = 0$, onda definiramo $F(x,y) = 0$. U gornjem primjeru je $F(3,2) = 1$, $F(3,3) = 2$ i $F(3,4) = 3$.

(a) Ako je $A(x,y) = 1$, nađite kako se $F(x,y)$ može izračunati iz susjednih vrijednosti (lijevo i gore) $F(x-1,y-1)$, $F(x-1,y)$ i $F(x,y-1)$. Argumentirajte dobiveni rezultat.

(b) Sastavite algoritam koji nalazi traženi kvadrat u zadanoj matrici A i vraća pripadne podatke ℓ, i, j. U algoritmu smijete koristiti jednu dodatnu matricu reda n. Složenost algoritma mora biti u $O(n^2)$. Ukratko argumentirajte da vaš algoritam zadovoljava taj uvjet.

3. (25) 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 operacija na strukturi?

(b) Opišite prikaz strukture disjunktnih skupova pomoću šume naopakih korijenskih stabala, spremanje tog prikaza u jednom polju i neke **efikasne** implementacije osnovnih operacija za taj prikaz. Komentirajte njihovu složenost. Složenost, u smislu mjere iz (a), **mora** biti u $O(n \log n)$.

(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: utorak, 4. 2. 2020., navečer (na webu),
srijeda, 5. 2. 2020. u 12 sati.

Saša Singer