

- ① Koje asimptotske notacije postoje?
teta(Θ), veliko o (O), malo o (o), veliko omega (Ω), malo omega (ω).
② Kako definišemo ~~teta~~ notaciju; $\Theta(g) = ?$

Teta služi za određivanje vremena izvršenja u najgorem slučaju; uko je opisuje?

* Uslov: $0 \leq c_1 g(n) \leq f(n) \leq c_2 g(n)$ za svako $n \geq n_0$

$\Theta(g(n))$ je skup funkcija $f(n)$

- ③ Kako definišemo veliko o (O -notacija)?

O -notacija se koristi za određivanje ASIMPTOTSKI GORNJE GRANICE zadate funkcije koje zavise samo od strukture programa

- ④ Uslov za veliko O sa gornje strane ograničava f
 $0 \leq f(n) \leq c \cdot g(n)$ za svako $n \geq n_0$

- ⑤ U matematičkom smislu u kom su odnosu veliko teta i veliko O . * Oba su skupovi
* Veliko teta je podskup od velikog O

- ⑥ Kako glasi teorema o asimptotskim notacijama.

Za bilo koje dve funkcije $f(n)$ i $g(n)$, važi da je
 $f(n) = \Theta(g(n))$ ako i samo ako je $f(n) = O(g(n))$ i
 $f(n) = \Omega(g(n))$. \rightarrow veliko omega

- ⑦ Kako glasi rekurentna jednačina za master teoremu?

$$T(n) = aT(n/b) + f(n)$$

$$a \geq 1$$
$$b > 1$$

} konstante $f(n)$ - asimptotski pozitivna f-ja
nenegativan

⑥ Kako smo definisali rad nad grafom računanja?

- To smo definisali kao ukupno vreme računanja na 1 procesoru T_1

⑦ Kako smo definisali raspon? T_∞

- Najveće vreme koje je potrebno da se izvrše linije duž bilo kog puta - Najduži put je kritična putanja.

⑧ Kako glasi zakon rada?

$$T_p \geq \frac{T_1}{p}$$

⑨ Kako glasi zakon raspona?

$$T_p \geq T_\infty$$

* Nije dovoljno meriti samo ubrzanje, nego treba i računati rad i raspon da bismo znali kakvi će rez. sistemi biti za zadatim delom procesora.

⑩ Kako smo definisali ubrzanje paralelnog programa? $\frac{T_1}{T_p}$

Ubrzanje je jednako vremenu serijskog izvršenja kroz vreme na p procesora

Linearno ubrzanje $\Theta(p)$

Savršeno ubrzanje $= p$

⑪ Kako glasi gornja granica za vreme T_p ?

$$\frac{T_1}{T_\infty} - \text{paralelizam}$$

Teorema o gornjoj granici:

$$T_p \leq \frac{T_1}{p} + T_\infty$$

ali nije to odgovor

⑫ Kako definišemo ~~skalarnost~~ paralelizma?

Kad paralelizam podelimo sa brojem procesora p

⑬ Čemu je jednak raspon redne veze 2 podgrafa?

Jednak je zbiru tih raspona pojedinačnih podgratova, a raspon kod paralelne veze je maksimalan raspon od ta 2.

⑭ Čemu je jednak raspon parallel for petlje i ide od 1 do n .

$$\lg n = \log_2 n$$

↳ pretvara se u bin. stablo i visina tog stabla je $\lg n$

Pitajna za prethodna predavanja - PARALELNO

① Algoritam insertion sort, sortirajuća sa umetanjem elemenata
U koju klasu algoritama spada taj algoritam?

To je inkrementalan algoritam jer inkrementalno gradi rešenja.

② Šta su prednosti, a šta mane ovog algoritma? ^{kod malih nizova}

Prednost: efikasnost je dobra na malom broju jer raste sa kvadratom pa je kvadrat mali; * ne koriste dodatnu memoriju već se radi inplace

Mana: vreme izvršavanja se povećava mnogo; kad se nizovi povećavaju

③ Kako glasi invarijanta tog niza? ^{a od 1 do j-1}

j trenutni član niza koga posmatramo, onda je niz $a[1]$ do $a[j-1]$ podniz tog niza i on je uvek sortiran.

④ Kako dokazujemo korektnost algoritama za koji imamo neku invarijantu definisanu?

1 Inicijalizacija - mora da važi pre prve iteracije

2 Održavanje - ako je istinita pre iteracije petlje, ostaje istinita posle iteracije, tj. pre sledeće iteracije

3 Završetak - kada se petlja završi, invarijanta daje korisnu osobinu koja olakšava dokazivanje da je algoritam korektan

17-03-2012.

→ ⑤ Kako glasi završetak za ovaj naš algoritam? 5:00

⑥ Koje je najgore ^{vreme} izvršenje ovog algoritma tj. kako glasi

asimptotski uska notacija ovog algoritma?

teta $\Theta(n^2)$ $An^2 + Bn + c$

Apstraktne klase

bad

↓ moraju da imaju [✓] 1 pure virtual metodu, i ona je prazna

* U izvedenoj klasi mora da se implementira ta pure virtual klasa

* Ne možemo da instanciramo apstraktnu klasu
oza klasu Shape to je `getArea()`;

Npr.

```
Shape *shapes[] =
```

```
{ new Square(5),  
  new Triangle(8,10),  
  new Square(7),  
  new Triangle(3,4)  
};
```

```
for (int i=0; i<4; i++)
```

```
cout << "Shape" << i <<
```

```
" : " << shapes[i] -> area() << endl;
```

! Ako izvedena klasa ne pregrati (override) tu pure virtual fju, onda ona automatski postaje i sama apstraktna klasa.

Inverzija paralelnog algoritma sortiranja
sa spajanjem podnizova

* za serijski algoritam $T(n) = \Theta(n \lg n)$

↳ koristi pristup podeli i zavladaaj

vreme po svakom nivou
stabla; čvorovi kad se
visina stabla = $\lg n$ saberi
* zanemariću
samo +1

$T_1(n) = \Theta(n \lg n)$ $T_1(n) = T(n) \rightarrow$ uklonimo spawn i sync
Δ RAD Δ \uparrow vremenu. sekv. izvršavanja

▷ RASPON imamo 3 podgrafa (1-2, 3-5, linija 6)

$\Theta(1)$ $\Theta(n/2)$ $\Theta(n)$

ser. procedura
rad = raspon

$$T_\infty(n) = \Theta(1) + T_\infty(n/2) + \Theta(n)$$

Merge-Sort Merge

$$T_\infty(n) = T_\infty(n/2) + \Theta(n) = \Theta(n)$$

$$\log_2 1 = 0 \quad f_n = \Theta(n)$$

III slučaj uslov regularnosti
važi jer je lin. fja

Paralelizam $\frac{T_1(n)}{T_\infty(n)} = \Theta(\lg n)$ nije tako veliki

1:15:00

Strategija podeli-i-zavladaaj za paralelno spajanje podnizova
slika:

Binary-search

$$T_1(n) = \Theta(n \lg n) = T_\infty(n)$$

Analiza procedure P-Merge

$$T_\infty(n) = T_\infty(3n/4) + \Theta(\lg n)$$

* osnovni slučaj $\Theta(1) \leftarrow$ raspon

$$T_\infty(n) = \Theta(\lg^2 n)$$

rad P-Merge-Sort
 $T_1(n) = 2T_1(n/2) + TPM_1(n)$
 $= 2T_1(n/2) + \Theta(n)$
 $T_1(n) = \Theta(n \lg n)$
 raspon $\rightarrow T_\infty(n) = T_\infty(n/2) + TPM_\infty(n)$
 $= T_\infty(n/2) + \Theta(\lg^2 n)$
 ne može master
 $\rightarrow T_\infty(n) = \Theta(\lg^3 n)$ metod smene

⑥ Algoritmi za sortiranje, 1 alg. \rightarrow uveli spawn za rekurzivne pozive i koji je rad za to i rešenje?
 \downarrow
i sync iz njih

$$T_1(n) = 2T_1\left(\frac{n}{2}\right) + \theta(n)$$

$$T_1(n) = \theta(n \cdot \lg n) \quad \text{II slučaj po master teoremi}$$

⑦ Koje je rešenje za raspon za 1 rešenje sortiranja?

$$T_\infty(n) = T_\infty\left(\frac{n}{2}\right) + \theta(n)$$

$$T_\infty(n) = \theta(n)$$

serijska procedura

II rešenje - paralelizacija procedure za spajanje

* analiza binary search metode

raspon se malo teže računa; nismo ravnomerno raspoređeni radovi na rekurzivnim pozivima

* Najgori slučaj: kad se spaja $\frac{3}{4}n$ elemenata i računanje rada složeno

- izračunali smo donju granicu, gornju granicu

\hookrightarrow metodom zamene
da bi došli do
 $\theta(n)$

- I onda smo prešli na analizu II rešenja gde smo koristili rezultate paralelizovanog mergea kao f je fb u rekurencijama za konačno rešenje

⑦ U koju klasu algoritama spada merge sort?

To je algoritam koji se zasniva na strategiji podeli i zavlada; a to se rekurzivno realizuje.

⑧ Koja je prednost, a koja je mana ovog algoritma?

Prednost - malo brži u odnosu na prethodni algoritam

Mana - ~~zauzima~~ zauzimamo dodatnu memoriju

⑨ Kako glasi invarijanta za ovaj algoritam? 8:00

* sortirano uvek od $a[p]$ do $a[k-1]$

⑩ Kako dokazujemo korektnost ovog algoritma?

- ? 8:00
- 1 Inicijalizacija - $k-p \rightarrow$ prazan niz \rightarrow sortiran uvek
 - 2 Održavanje \rightarrow $\begin{cases} \text{1 slučaj ubaci element iz levog niza} \\ \text{2 slučaj ubaci element iz desnog niza} \end{cases}$
 - 3 Završetak

⑪ Kako glasi rekurencija za ovaj algoritam?

$$T[n] = \begin{cases} 2T(n/2) + \Theta(n), n > 1 \end{cases}$$

rekurzivno poziva samu sebe da uradi
 \rightarrow poziva se procedura spajanja
kombinuje međurezultate polovinu problema

⑫ Zašto je $f(n)$ -vreme merge procedure $\Theta(n)$?

Zato što moramo da prođemo kroz ceo niz, tj. kroz n elemenata, a osnovni slučaj nosi konstantno vreme

$$\Theta(1) \quad \underline{n \cdot \Theta(1) = \Theta(n)}$$

⑬ Koji je to slučaj u master teoremi?

- To je 2. slučaj master teoreme, a rešenje je

$$T(n) = \Theta(n \cdot \log n)$$

⑮ pa na kraju je poslednji uslov - provera petlje i sam izlazak
iz petlje i onda je $j-1$ tj. taj podniz od 1 do $j-1$ ceo
sortiran

⑧ Šta poredimo u master metodi / poredimo $f(n)$ sa čime?

U sva tri slučaja $f(n)$ se poredi sa $f(n)$ om $n^{\log_b a}$.

⑨ Ako su f i $n^{\log_b a}$ asimptotski jednaki; kako glasi $T(n)$? \hookrightarrow 2-slučaj

$$T(n) = \Theta(n^{\log_b a} \lg n) \quad \boxed{\lg n = \log_2 n}$$

$$f(n) = \Theta(n^{\log_b a})$$

⑩ Čemu služe asimptotske notacije?

Sa njima možemo da prikazemo kako ~~ve~~ vreme i trošenje memorije zavisi od veličine problema (npr. broj elemenata niza)

Platforma za dinamičku paralelizaciju

③ ① Koje dve osnovne apstrakcije podržava ta platforma

- ugnježdeni paralelizam
- paralelne petlje

② Problem računanja fib. brojeva i algoritam rekursivni.
Kako smo paralelizovali taj alg.

Koristili smo ključne reči *spawn* i *sync*.

Vreme izvršavanja tog ^{serijski} sekvencijalnog algoritma:

$$T(n) = \Theta(F_n)$$

* zlatni odnos

③ Kako smo definisali graf računanja?

Čvorovi predstavljaju neke instrukcije, tj. linije izvršavanja
a grane predstavljaju njihove međusobne zavisnosti.

Vrste grana:

- * grana mreščenja
- * grana nastavka
- * grana poziva
- * grana povratka

③ Tako do podataka - postoji između 2 paralelne instrukcije ukoliko bar jedna od njih upisuje u deljenu promenljivu.

④ Kako se definiše ~~redna~~ redna veza 2 linije izvršavanja u i v u grafu računanja?

Ako u tom grafu postoji neka usmerena putanja od linije u do v , onda su te linije u logičkoj rednoj vezi; a ako ta putanja ne postoji onda su u paralelnoj vezi.

⑤ Kako smo definisali sekvencijalno konzistentnu memoriju / deljenu? * linearno se izvršavaju tih više ^{read} _{write} operacija zadržava se redosled u grafu računanja kao da ih je 1 procesor redom izvršavao.

III alg. - Strassenov metod za množenje matrica

* Umesto 8 množenja matrica $n/2 \times n/2$, on obavlja 7

račun S_1, S_2, \dots, S_{10} * u 2 ugnježdene for petlje

(rad) (vreme izvršenja za 1 korak) $\Theta(n^2)$

račun $C_{11}, C_{21}, \dots, C_{22}$ - rad $\Theta(n^2)$
mat. C_{12}, C_{22}
C

raspon - 2 ugnježden for petlje se mogu paralelizovati $\Theta(\lg n)$

račun P_1, \dots, P_7 - izvodimo kroz rekurzivne pozive $7 \cdot T(\frac{n}{2})$

mat. P
n=1 račun nad skalarima
 $T(n) = \begin{cases} \Theta(1), & n=1 \\ 7 \cdot T(\frac{n}{2}) + \Theta(n^2), & n>1 \end{cases}$ rad koji potiče od računa za mat. i matricu C.

7 rekurzivnih poziva
svaki deluje nad
polovinom elemenata

$$\lg 8 = 3$$

$\lg 7 \rightarrow$ malo manje od 3

$$n^{\lg 7} > n^2 \quad | \text{ slučaj}$$

Primenom
master
metode

$$T(n) = \Theta(n^{\lg 7})$$

orig. alg. $\Theta(n^3)$

Asimptotski brže od direktnog množenja matrice

* Strassenov metod dovodi do toga da serijski algoritam je asimpt. brži od običnog alg. zasnovanog na 3 for petlje.

Paralelizovan Strassenov metod - paralelizujemo for petlje za račun mat. S u C; rekurzivne pozive paralel. - prvih 6 mogu da budu izmrešćeni - njima ćemo dodati spawn ispred a 7. će biti pozvan kao obična procedura.

S_1, S_{10} rad $\Theta(n^2)$
raspon $\Theta(\lg n)$

$$\text{Rad } T_1(n) = \Theta(n^{\lg 7})$$

$$\text{Raspon } T_2(n) = \Theta(\lg^2 n)$$

Paralelizam

$$\frac{T_1(n)}{T_2(n)} = \Theta\left(\frac{n^{\lg 7}}{\lg^2 n}\right)$$

$C_{11}, C_{12}, \dots, C_{21}, C_{22}$ rad $\Theta(n^2)$
raspon $\Theta(\lg n)$

[Malo niži od paralelizma P-Matrix-Multiply-Recursive
[Manje razgranat graf računanja (svaki čvor 7 potomaka - potrošnja memorije manja)] 2

4) Pitanja

① koji je rad, odnosno raspon za taj direktan algoritam?
Rad i raspon 1 algoritam

↳ onoliko koliko imamo ugnježenih petlji -

$\Theta(n^2) \rightarrow$ rad da bismo izbegli tik do podataka

$\Theta(n) \rightarrow$ raspon (prethodne 2 paralelizovane; 3. for petlja urađena kao serijski for)

② // algoritam - podeli i zavlada - kako glasi rekurencija za rad za taj algoritam. čemu je jednak $T_1(n)$?

8 slučajeva / 2 ugnježene petlje

$$T_1(n) = 8T_1\left(\frac{n}{2}\right) + \Theta(n^2) \quad \text{master metoda}$$

rešenje $T_1(n) = \Theta(n^3)$

③ kako glasi rekurencija za raspon za 2. algoritam? ^{raspon}
najveći (svi su isti za 8 paralelnih instanci)

$$T_2(n) = T_2\left(\frac{n}{2}\right) + \Theta(\lg n)$$

rešenje $T_2(n) = \Theta(\lg^2 n)$ metoda zamene

2 ugnježene
parallel for petlje
 $2\Theta(\lg n)$

④ kako glasi rekurencija za rad za Strassenov algoritam?

$$T(n) = \begin{cases} \Theta(1) & , n=1 \\ 7 \cdot T\left(\frac{n}{2}\right) + \Theta(n^2) & , n>1 \end{cases}$$

master metoda
↳ $T(n) = \Theta(n^{\lg 7})$

⑤ kako glasi rekurencija za raspon za Strassenov algoritam?
rešenje: $T_2(n) = \Theta(\lg^2 n)$

* imamo 7 rekursivnih poziva - $\Theta\left(\frac{n}{2}\right)$

* imamo 2 parallel petlje - parallel od 1 do n $\Theta(\lg n)$

~ rešavamo pomoću smene

TEST ① Šta predstavlja dekompozicija zadatka u toku pronalazanja paralelizma?

② Napisati formulu za gornju granicu vremena izvršavanja paralelnog algoritma na P procesora. Napisati & posledice te teoreme. ✓

③ Definisati labavost paralelizma paralelnog algoritma koji se izvršava na idealnom paralelnom računaru sa P procesora. Koja je veza labavosti paralelizma i savršenog lin. ubrzanja? Objasni značenje elem. u jednačinama.

* ④ Ra, Raspon i paralelizam metode P-Square-Matrix-Multiply

* ⑤ Koje osobine ima invarijanta petlje da bi se utvrdila korektnost

a) $n^2/2 = W(n)$ DA NE

* ⑥ b) $n^2/2 = W(n^2)$ DA NE

* ⑦ a) $T(n) = 2T(\frac{n}{4}) + \sqrt{n} + 1023$

b) $T(n) = 3T(n/4) + n \lg n$

c) $T(n) = T(2n/3) + 1$