Øving 3, teori: Splitt og hersk

Your answer passed the tests! Your score is 100.0% ×

Question 1: Sorteringsalgoritmer

Vi har en usortert liste med n elementer, og vi ønsker å finne ut hvor mange unike tall det er i listen. Etter algoritmene vi har lært til og med forelesningen om splitt og hersk, hva er den raskeste kjøretiden vi kan garantere for dette problemet (den beste worst-case kjøretiden)?

- ( ) Ingen av alternativene stemmer
- ( ) $\Theta(n)$
- ( ) $\Theta(n^2)$
- ( ) $\Theta(n\sqrt{n})$
- ( ) $\Theta(\log \log n)$
- ( ) $\Theta(\log n)$
- ( ) $\Theta(\sqrt{n})$
- (•) $\Theta(n \log n)$

Her går løsningen ut på å først sortere listen. Da kan vi enkelt finne unike elemeneter ved å iterere gjennom den (et element er unikt dersom det er ulikt elementet til venstre og til høyre). Det beste vi kan garantere for sortering uten å gjøre antagelser er theta(n log n). Det blir derfor kjøretiden til fremgangsmåten.

Her er det fristende å svare lineærtid med en hashtabell-løsning. Da er vi nødt til å huske på at innsetting i en hash-tabell har worst-case O(n), og innsetting av n tall har dermed worst-case O(n^2).

Question 2: Sorteringsalgoritmer

Denne oppgaven handler om quicksort og mergesort. Med "sorteringsarbeid" menes her den faktiske flyttingen av tall som en sorteringsalgoritme gjør. Hvilket av følgende alternativ er sant?

- (•) Quicksort gjør sorteringsarbeidet før det rekursive kallet, mens mergesort gjør det etterpå
- ( ) Begge algoritmene gjør sorteringsarbeidet etter det rekursive kallet
- ( ) Mergesort gjør sorteringsarbeidet før det rekursive kallet, mens quicksort gjør det etterpå
- ( ) Begge algoritmene gjør sorteringsarbeidet før det rekursive kallet

Quicksort partisjonerer først, og så kaller den seg selv rekursivt.

Mergesort kaller seg selv rekursivt først, og så fletter den sammen de to sorterte delene.

Question 3: Sorteringsalgoritmer

I begge tilfeller får vi den dårligst-mulige splitten, der kun ett element blir sortert per nivå av Quicksort.

La liste A være en liste sortert i stigende rekkefølge, og B en liste sortert i synkende rekkefølge. Hvilken påstand stemmer da om kjøretiden til Quicksort?

- (x) Begge har kjøretid $\Theta(n^2)$
- ( ) Kjøretid for liste B er $\Theta(n^2)$ og for liste A er kjøretiden $\Theta(n \log n)$
- ( ) Kjøretid for liste A er $\Theta(n^2)$ og for liste B er kjøretiden $\Theta(n \log n)$
- ( ) Begge har kjøretid $\Theta(n \log n)$

Question 4: Sorteringsalgoritmer

Mergesort bruker O(n) ekstra minne i average case. For å gjøre flettingen trenger den ekstra minne som har plass til alle tallene som skal flettes.

Quicksort allokerer ingen nye lister, men den må ta vare på tre indekser per rekursjonsnivå. I average case går rekursjonen log(n) dypt, og da bruker algoritmen log(n) ekstra minne. I worst case bruker også quicksort O(n) ekstra minne - da er rekursjonen O(n) dyp.

Hvilken av algoritmene forbruker mest ekstra minne i average case?

- (x) Merge sort
- ( ) Quicksort

Question 5: Sorteringsalgoritmer

Siden algoritmen velger et tilfeldig pivot-element på hvert rekursjonsnivå, så kan vi alltid være uheldige og velge det største eller minste elementet som pivot, hver gang. Når den er randomisert vet vi ikke på forhånd hvilken liste som kommer til å gi worst-case.

All input kan gi worst-case kjøretid for randomized-quicksort

- (x) Sant
- ( ) Usant

Type your text

Question 6: Substitusjonsmetoden

Du ønsker å teste om kjøretiden til fire ulike, rekursive algoritmer er $O(n^2)$ ved hjelp av substitusjonsmetoden. Først setter du opp rekurrenser for algoritmene, og så forutsetter du for hver av dem den induktive hypotesen at $T(n) \leq c \cdot n^2$. Etter å ha gjennomført substitusjonsmetoden for hver av rekurrensene får du resultatene

$$T_1(n) \leq c \cdot n^2 - 5n$$

$$T_2(n) \leq c \cdot n^2 + 5n$$

$$T_3(n) \leq c \cdot n^2 + 1$$

$$T_4(n) \leq c \cdot n^2 - 1$$

Hvilke(n) av algoritmene har du greid å bevise at har kjøretid $\mathrm{O}(n^2)$? Anta at grunntilfellene i den matematiske induksjonen også stemmer.

- [x] T_1
- [x] T_4
- [ ] T_2
- [ ] T_3

For T1 og T4 kan vi fullføre beviset med å stegene

cn² - 5n <= cn²
og
cn² - 1 <= cn²

Da har vi kommet frem til, og dermed bevist, den induktive hypotesen vår. Det greier vi ikke med T2 og T4 som her gir et mindre strengt krav enn vår induktive hypotese.

Merk at med substitusjonsmetoden må vi bevise nøyaktig, og ikke asymptotisk. Her kan vi ikke argumentere med at + 1 for T3 er asymptotisk ubetydelig.

Question 7: Master-teoremet

La $T(n) = 27 \cdot T(n/3) + n^3$. Hvilket tilfelle tilhører rekurrensen når du benytter master-teoremet?

- ( ) Case 1
- (•) Case 2
- ( ) Case 3
- ( ) Ingen av dem

Vi får: log3(27) = 3

og dermed n³ = theta(f(n)) = theta(n³)

Da har vi case 2.

Question 8: Master-teoremet

La $T(n) = 27 \cdot T(n/3) + n^3$. Hva blir kjøretiden?

- (•) $\Theta(n^3 \log n)$
- ( ) $\Theta(n^4)$

Case 2 gir at løsningen er theta(n³ log(n))

- ○ $\Theta(n^4 \log n)$
- ○ $\Theta(n^3)$

Question 9: Master-teoremet

La $T(n) = 4 \cdot T(n/2) + n^3$. Hva blir kjøretiden?

- ○ $\Theta(n^3 \log n)$
- ○ $\Theta(n^2)$
- ○ $\Theta(n^4)$
- ● $\Theta(n^3)$

log2(4) = 2

Dermed er n² = O(f(n)) = O(n³)

Vi får løsningen theta(n³)

Question 10: Rekursjonstrær

La $T(n) = T(n/3) + T(n/2) + T(n-1) + 1$der $T(1) = 1$. Hva blir høyden til rekursjonstreet?

- ○ $\Theta(n \log n)$
- ○ $\Theta(\log n)$
- ○ $\Theta(n^2)$
- ● $\Theta(n)$

Høyden blir gitt av leddet T(n-1), som dominerer i forhold til høyden på treet. Denne minker problem-størrelsen med én per nivå, og gir dermed en høyde lik n-1. Den asymptotiske høyden er da theta(n).

Question 11: Variabelskifte

Løs rekurensen gitt ved $T(n) = T(\sqrt{n}) + n$ved hjelp av variabelskifte. Hva blir kjøretiden?

Hint: $\sqrt{n}$ er det samme som $n^{\frac{1}{2}}$.

- ○ $\Theta(\log n)$
- ● $\Theta(n)$
- ○ $\Theta(\log \log n)$

Vi bruker substitusjonsmetoden og setter

m = lg(n) <=> 2^m = n

T(2^m) = T(2^(m/2)) + 2^m

Innfør S(m) = T(2^m)

S(m) = T(m/2) + 2^m

Med masterteoremet har denne løsning S(m) = theta(2^m)

Substituerer tilbake:

T(2^m) = theta(2^m)

T(n) = theta(n)

- $\Theta(n \log n)$

Question 12: Kjøretidsanalyse

Funksjonen `gjoerNoe(n)` under har kjøretid $\Theta(n)$. Hva blir kjøretiden til funksjonen `f1(n)` ?

Hint: Sett opp to rekurrenser $T_1(n)$ og $T_2(n)$ og finn først en løsning på lukket form for $T_1(n)$.

```
function f1(n)
    gjoerNoe(n)
    if n > 1
        f1(n / 2)
        f2(n - 2)
    end
end

function f2(n)
    gjoerNoe(n)
    if n > 1
        f1(n / 2)
    end
end
```

- (selected) $\Theta(n \log n)$
- $\Theta(n^2)$
- $\Theta(\log n)$
- $\Theta(n)$

Setter opp:

T1(n) = T1(n/2) + T2(n-2) + theta(n)
T2(n) = T1(n/2) + theta(n)

Vi setter uttrykket for T2(n) inn i T1(n), og eliminerer da T2(n) fra uttyrkket til T1(n).

T1(n) = T1(n/2) + [T1((n-2) / 2) + theta(n-2)] + theta(n)

Asymptotisk har ikke "-2"-leddet noe å si, så vi fjerner det.

T1(n) = T1(n/2) + T1(n/2) + theta(n) + theta(n)
= 2T1(n/2) + theta(n)

Dette kan vi løse med masterteoremet, og case 2 gir oss løsningen

T1(n) = theta(n log n)

Question 13: Kjøretidsanalyse

Hva blir kjøretiden til funksjonen `f3(n)` ?

```
function f3(n)
    if n > 42
        f3(n - 42)
        f3(42)
    end
end
```

- ( ) $O(n \log n)$
- ( ) $O(\log n)$
- ( ) $O(1)$
- ( ) Den vil aldri stoppe
- (•) $O(n)$

Vi ser at kallet f3(42) avslutter på konstant tid. Vi får

T3(n) = T3(n - 42) + 1

Dette egner seg godt til å løse med iterasjonsmetoden.

T3(n) = T3(n - 42) + 1

T3(n) = [T3(n - 42 - 42) + 1] + 1
  = T3(n - 2*42) + 2
  ...
  = T3(n - i * 42) + i

Grunntilfellet er T(42) og
  n - i * 42 = 42
  i = (n - 42) / 42 = n/42 + 1

T3(n) = T(42) + n/42 + 1
  = 1 + n/42 + 1
  = n/42 + 2 = theta(n)

Question 14: Kjøretidsanalyse

Hva blir kjøretiden til funksjonen `f4(n)`? `println` tar konstant tid.

```
function f4(n)
    for i in 1:n
        println("Algdat ruler!")
    end

    if n > 1
        f4(3/4* n)
        f4(1/4* n)
    end
end
```

- ( ) $O(1)$
- ( ) $O(n)$
- (•) $O(n \log n)$
- ( ) $O(n^2)$
- ( ) $O(\log n)$

Setter opp:

T(4) = T4(3/4 n) + T4(1/4 n) + n

Denne egner seg godt for å løse med rekursjonstrær. Da ser vi at vi får en kostnad per nivå lik n, og antall nivå er øvre begrenset av T(3/4 n) som gir en høyde på

log_4/3(n) = theta(log(n))

Løsningen får vi da av å gange sammen antall nivå med kostnad per nivå:

T(n) = O(n * theta(log(n))) = O(n log(n))

Question 15: Kjøretidsanalyse

Funksjonen `gjoerNoeAnnet(n)` under har kjøretid $\Theta(n^2)$. Hva blir kjøretiden til funksjonen `f5(n)`?

```
function f5(n)
    gjoerNoeAnnet(n/6)

    if n > 1
        f5(n/2 - 2)
        f5(n/2 - 3)
    end
end
```

- ○ $O(\log n)$
- ○ $O(n \log n)$
- ○ $O(n)$
- ● $O(n^2)$
- ○ $O(n^2 \log n)$

I denne oppgaven er det ment at du skal forstå at små ledd kan oversees når vi regner asymptotisk.

Da kan vi sette opp en relativt enkel rekurrens

T5(n) = 2T5(n/2) + theta(n²)

Løst med masterteoremet får vi case 3 og løsning

T5(n) = O(n²)

Submit

>_