Øvingsforelesning 5

TDT4120 - Algoritmer og datastrukturer

Øving 4

Stabil sortering

Oppgave 2: Hva betyr det at en sorteringsalgoritme er stabil?

Stabil sortering

Oppgave 2: Hva betyr det at en sorteringsalgoritme er stabil?

Like elementer opptrer i samme rekkefølge som i den usorterte listen.

Stabil sortering

Oppgave 2: Hva betyr det at en sorteringsalgoritme er stabil?

Oppgave 3: $\mathrm{A} = \langle(8, 3), (3, 4), (3, 2), (5, 1), (1, 5), (3, 3)\rangle$, hvordan ser A ut etter å ha blitt sortert basert på *x*-verdiene med en stabil sorteringsalgoritme?

Stabil sortering

Oppgave 2: Hva betyr det at en sorteringsalgoritme er stabil?

Oppgave 3: $\mathrm{A} = \langle (8,3), (3,4), (3,2), (5,1), (1,5), (3,3) \rangle$, hvordan ser A ut etter å ha blitt sortert basert på *x*-verdiene med en stabil sorteringsalgoritme?

$\mathrm{A} = \langle (1,5), (3,4), (3,2), (3,3), (5,1), (8,3) \rangle$

Stabil sortering

Oppgave 2: Hva betyr det at en sorteringsalgoritme er stabil?

Oppgave 3: $\mathrm{A} = \langle(8,3),(3,4),(3,2),(5,1),(1,5),(3,3)\rangle$, hvordan ser A ut etter å ha blitt sortert basert på *x*-verdiene med en stabil sorteringsalgoritme?

Oppgave 4: Hvilke av disse er stabile sorteringsalgoritmer?

- Insertion-Sort
- Merge-Sort
- Quicksort
- Counting-Sort
- Radix-Sort med Insertion-Sort som subrutine
- Radix-Sort med Quicksort som subrutine

Stabil sortering

Oppgave 2: Hva betyr det at en sorteringsalgoritme er stabil?

Oppgave 3: $\mathrm{A} = \langle(8,3),(3,4),(3,2),(5,1),(1,5),(3,3)\rangle$, hvordan ser A ut etter å ha blitt sortert basert på *x*-verdiene med en stabil sorteringsalgoritme?

Oppgave 4: Hvilke av disse er stabile sorteringsalgoritmer?

- INSERTION-SORT
- MERGE-SORT
- ~~QUICKSORT~~
- COUNTING-SORT
- RADIX-SORT med INSERTION-SORT som subrutine
- RADIX-SORT med QUICKSORT som subrutine

Stabil sortering

Oppgave 2: Hva betyr det at en sorteringsalgoritme er stabil?

Oppgave 3: $\mathrm{A} = \langle(8,3),(3,4),(3,2),(5,1),(1,5),(3,3)\rangle$, hvordan ser A ut etter å ha blitt sortert basert på *x*-verdiene med en stabil sorteringsalgoritme?

Oppgave 4: Hvilke av disse er stabile sorteringsalgoritmer?

- Insertion-Sort
- Merge-Sort
- ~~Quicksort~~
- Counting-Sort
- Radix-Sort med Insertion-Sort som subrutine
- ~~Radix-Sort med Quicksort som subrutine~~

Oppgave 5: For en vilkårlig sammenligningsbasert sorteringsalgoritme, hva kan vi si om kjøretiden til algoritmen i verste tilfelle?

Sammenligningsbasert sortering

Oppgave 5: For en vilkårlig sammenligningsbasert sorteringsalgoritme, hva kan vi si om kjøretiden til algoritmen i verste tilfelle?

Kjøretiden må være $\Omega(n \lg n)$.

Sammenligningsbasert sortering

Oppgave 5: For en vilkårlig sammenligningsbasert sorteringsalgoritme, hva kan vi si om kjøretiden til algoritmen i verste tilfelle?

Oppgave 6: Hva stemmer om en sammenligningsbasert sorteringsalgoritme, S?

- Kjøretiden til S i beste tilfelle kan være $\Theta(n)$.
- S er en stabil sorteringsalgoritme.
- Kjøretiden til S i verste tilfelle kan være $\Theta(n^3)$.
- Kjøretiden til S i verste tilfelle kan være $\Theta(n)$.
- S kan være QUICKSORT
- S kan være BUCKET-SORT

Sammenligningsbasert sortering

Oppgave 5: For en vilkårlig sammenligningsbasert sorteringsalgoritme, hva kan vi si om kjøretiden til algoritmen i verste tilfelle?

Oppgave 6: Hva stemmer om en sammenligningsbasert sorteringsalgoritme, S?

- Kjøretiden til S i beste tilfelle kan være $\Theta(n)$.
- ~~S er en stabil sorteringsalgoritme.~~
- Kjøretiden til S i verste tilfelle kan være $\Theta(n^3)$.
- Kjøretiden til S i verste tilfelle kan være $\Theta(n)$.
- S kan være Quicksort
- S kan være Bucket-Sort

Sammenligningsbasert sortering

Oppgave 5: For en vilkårlig sammenligningsbasert sorteringsalgoritme, hva kan vi si om kjøretiden til algoritmen i verste tilfelle?

Oppgave 6: Hva stemmer om en sammenligningsbasert sorteringsalgoritme, S?

- Kjøretiden til S i beste tilfelle kan være $\Theta(n)$.
- ~~S er en stabil sorteringsalgoritme.~~
- Kjøretiden til S i verste tilfelle kan være $\Theta(n^3)$.
- ~~Kjøretiden til S i verste tilfelle kan være $\Theta(n)$.~~
- S kan være QUICKSORT
- S kan være BUCKET-SORT

Sammenligningsbasert sortering

Oppgave 5: For en vilkårlig sammenligningsbasert sorteringsalgoritme, hva kan vi si om kjøretiden til algoritmen i verste tilfelle?

Oppgave 6: Hva stemmer om en sammenligningsbasert sorteringsalgoritme, S?

- Kjøretiden til S i beste tilfelle kan være $\Theta(n)$.
- ~~S er en stabil sorteringsalgoritme.~~
- Kjøretiden til S i verste tilfelle kan være $\Theta(n^3)$.
- ~~Kjøretiden til S i verste tilfelle kan være $\Theta(n)$.~~
- S kan være QUICKSORT
- ~~S kan være BUCKET-SORT~~

Kjøretid ikke-sammenligningsbaserte sorteringsalgoritmer

Oppgave 7: Gitt en tabell med n heltall i intervallet $[0, k]$, $k \gg n$, hvilket uttrykk ville man brukt for å beskrive kjøretiden til COUNTING-SORT på tabellen?

Kjøretid ikke-sammenligningsbaserte sorteringsalgoritmer

Oppgave 7: Gitt en tabell med n heltall i intervallet $[0, k]$, $k \gg n$, hvilket uttrykk ville man brukt for å beskrive kjøretiden til COUNTING-SORT på tabellen?

Vanlig kjøretid: $\Theta(n + k)$

Kjøretid ikke-sammenligningsbaserte sorteringsalgoritmer

Oppgave 7: Gitt en tabell med n heltall i intervallet $[0, k]$, $k \gg n$, hvilket uttrykk ville man brukt for å beskrive kjøretiden til COUNTING-SORT på tabellen?

Vanlig kjøretid: $\Theta(n + k)$
Med $k \gg n$: $\Theta(k)$

Kjøretid ikke-sammenligningsbaserte sorteringsalgoritmer

Oppgave 7: Gitt en tabell med n heltall i intervallet $[0, k]$, $k \gg n$, hvilket uttrykk ville man brukt for å beskrive kjøretiden til Counting-Sort på tabellen?

Oppgave 8: Hva er den gjennomsnittlige kjøretiden til Bucket-Sort med Insertion-Sort, hvis man anvender 512 bøtter?

Kjøretid ikke-sammenligningsbaserte sorteringsalgoritmer

Oppgave 7: Gitt en tabell med n heltall i intervallet $[0, k]$, $k \gg n$, hvilket uttrykk ville man brukt for å beskrive kjøretiden til COUNTING-SORT på tabellen?

Oppgave 8: Hva er den gjennomsnittlige kjøretiden til BUCKET-SORT med INSERTION-SORT, hvis man anvender 512 bøtter?

Kjøretiden blir $\mathrm{O}(n^2)$.

Tellesortering

Oppgave 9: Når COUNTING-SORT legger til elementer i B itereres det over A fra slutt til start. Hva er grunnen til dette?

Tellesortering

Oppgave 9: Når COUNTING-SORT legger til elementer i B itereres det over A fra slutt til start. Hva er grunnen til dette?

COUNTING-SORT$(\mathrm{A}, \mathrm{B}, k)$

let $\mathrm{C}[0..k]$ be a new array
for $i = 0$ **to** k
    $\mathrm{C}[i] = 0$
for $j = 1$ **to** $\mathrm{A}.\mathit{length}$
    $\mathrm{C}[\mathrm{A}[j]] = \mathrm{C}[\mathrm{A}[j]] + 1$
for $i = 1$ **to** k
    $\mathrm{C}[i] = \mathrm{C}[i] + \mathrm{C}[i-1]$
for $j = \mathrm{A}.\mathit{length}$ **downto** 1
    $\mathrm{B}[\mathrm{C}[\mathrm{A}[j]]] = \mathrm{A}[j]$
    $\mathrm{C}[\mathrm{A}[j]] = \mathrm{C}[\mathrm{A}[j]] - 1$

Tellesortering

Oppgave 9: Når COUNTING-SORT legger til elementer i B itereres det over A fra slutt til start. Hva er grunnen til dette?

```
COUNTING-SORT(A, B, k)
let C[0..k] be a new array
for i = 0 to k
        C[i] = 0
for j = 1 to A.length
        C[A[j]] = C[A[j]] + 1
for i = 1 to k
        C[i] = C[i] + C[i − 1]
for j = A.length downto 1
        B[C[A[j]]] = A[j]
        C[A[j]] = C[A[j]] − 1
```

Nødvendig for at COUNTING-SORT skal være en stabil.

Tellesortering

Oppgave 9: Når COUNTING-SORT legger til elementer i B itereres det over A fra slutt til start. Hva er grunnen til dette?

Oppgave 10: Skriv et program hvor du sorterer heltall i intervallet $[0, k]$, $k < 2048$ med COUNTING-SORT.

Tellesortering

Oppgave 9: Når COUNTING-SORT legger til elementer i B itereres det over A fra slutt til start. Hva er grunnen til dette?

Oppgave 10: Skriv et program hvor du sorterer heltall i intervallet $[0, k]$, $k < 2048$ med COUNTING-SORT.

Oversett pseudokoden i boken, hvor du bytter ut k med 2047.

Finne median

Oppgave 11: Hvilket av følgende utsagn om kjøretiden i verste tilfelle til algoritmer som finner medianen til en liste stemmer?

Finne median

Oppgave 11: Hvilket av følgende utsagn om kjøretiden i verste tilfelle til algoritmer som finner medianen til en liste stemmer?

Select finner det k-største elementet i en listen i $O(n)$ tid.

Finne median

Oppgave 11: Hvilket av følgende utsagn om kjøretiden i verste tilfelle til algoritmer som finner medianen til en liste stemmer?

SELECT finner det k-største elementet i en listen i $O(n)$ tid.

Det er mulig å oppnå en kjøretid på $O(n)$ i verste tilfelle.

Valg av sorteringsalgoritme

Oppgave 12: Ønsker å sortere en liste bestående av 100 000 resultater av kasting av en sekssidet terning. Hvilken av følgende sorteringsalgoritmer er mest effektive i denne situasjonen?

- Radix-Sort med Counting-Sort
- Insertion-Sort
- Merge-Sort
- Counting-Sort
- Bucket-Sort

Valg av sorteringsalgoritme

Oppgave 13: Ønsker å sortere en liste som er nesten sortert, med unntak av to elementer som har byttet plass med hverandre. Hvilken av følgende sorteringsalgoritmer er mest effektive i denne situasjonen?

- Radix-Sort med Counting-Sort
- Insertion-Sort
- Merge-Sort
- Counting-Sort
- Bucket-Sort

Valg av sorteringsalgoritme

Oppgave 14: Ønsker å sortere en liste bestående av 1 000 000 brukernavn på mellom 5 og 8 bokstaver. Hvilken av følgende sorteringsalgoritmer er mest effektive i denne situasjonen?

- Radix-Sort med Counting-Sort
- Insertion-Sort
- Merge-Sort
- Counting-Sort
- Bucket-Sort

Valg av sorteringsalgoritme

Oppgave 14: Ønsker å sortere en liste bestående av 1 000 000 brukernavn på mellom 5 og 8 bokstaver. Hvilken av følgende sorteringsalgoritmer er mest effektive i denne situasjonen?

- Radix-Sort med Counting-Sort - $d(n + k_0) \approx 8 \cdot 10^6$
- Insertion-Sort - $n^2 = 10^{12}$
- Merge-Sort - $n \lg n \approx 2 \cdot 10^7$
- Counting-Sort - $\geqslant 26^8 \approx 2 \cdot 10^{11}$
- Bucket-Sort

Radikssortering eller tellesortering

Oppgave 15: For en liste av n heltall i intervallet $[0, k]$, når ønsker vi å anvende $\textsc{Radix-Sort}$ som anvender $\textsc{Counting-Sort}$ innad over kun $\textsc{Counting-Sort}$?

Radikssortering eller tellesortering

Oppgave 15: For en liste av n heltall i intervallet $[0, k]$, når ønsker vi å anvende RADIX-SORT som anvender COUNTING-SORT innad over kun COUNTING-SORT?

COUNTING-SORT har en kjøretid på $\mathrm{O}(n + k)$

RADIX-SORT har en kjøretid på $\mathrm{O}(d(n + k_0)) = \mathrm{O}(dn + dk_0)$

Radikssortering eller tellesortering

Oppgave 15: For en liste av n heltall i intervallet $[0, k]$, når ønsker vi å anvende RADIX-SORT som anvender COUNTING-SORT innad over kun COUNTING-SORT?

COUNTING-SORT har en kjøretid på $\mathrm{O}(n + k)$

RADIX-SORT har en kjøretid på $\mathrm{O}(d(n + k_0)) = \mathrm{O}(dn + dk_0)$

Vet at $d = log_{k_0} k$, så $dk_0 \leqslant k$.

Radikssortering eller tellesortering

Oppgave 15: For en liste av n heltall i intervallet $[0, k]$, når ønsker vi å anvende $\textsc{Radix-Sort}$ som anvender $\textsc{Counting-Sort}$ innad over kun $\textsc{Counting-Sort}$?

$\textsc{Counting-Sort}$ har en kjøretid på $\mathrm{O}(n + k)$

$\textsc{Radix-Sort}$ har en kjøretid på $\mathrm{O}(d(n + k_0)) = \mathrm{O}(dn + dk_0)$

Vet at $d = log_{k_0} k$, så $dk_0 \leqslant k$.

Må ha $k > dn$, som vil si $k \gg n$.

Radikssortering - Implementasjon

Oppgave 16: Sorter tekststrenger med variabel lengde i lineær tid basert på den totale lengden til alle strengene.

Radikssortering - Implementasjon

Oppgave 16: Sorter tekststrenger med variabel lengde i lineær tid basert på den totale lengden til alle strengene.

```
FLEXRADIX(A, k)
let B[1..k] be a new array
for i = 1 to k
    B[i] = an empty array
for i = 1 to A.length
    add A[i] to B[A[i].length]
C = an empty array
for i = k downto 1
    C = B[i] + C
    sort C on the i-th character
return C
```

Oppgave 17: Select og Randomized-Select har forskjellig kjøretid i verste tilfelle. Hva skyldes dette?

SELECT og RANDOMIZED-SELECT

Oppgave 17: SELECT og RANDOMIZED-SELECT har forskjellig kjøretid i verste tilfelle. Hva skyldes dette?

RANDOMIZED-SELECT plukker et tilfeldig element som pivot.

Select og Randomized-Select

Oppgave 17: Select og Randomized-Select har forskjellig kjøretid i verste tilfelle. Hva skyldes dette?

Randomized-Select plukker et tilfeldig element som pivot.

Select plukker et element som er garantert å ha minst $\frac{3n}{10} - 6$ som er mindre/større.

K-Largest - Implementasjon

Oppgave 18: Finn de *k* største heltallene i A med gjennomsnittlig kjøretid på $\mathrm{O}(\mathrm{n})$ ved unike poengsummer.

K-LARGEST - Implementasjon

Oppgave 18: Finn de k største heltallene i A med gjennomsnittlig kjøretid på $O(n)$ ved unike poengsummer.

K-LARGEST(A, k)
return RANDOMIZED-K-LARGEST(A, 1, A.*length*, A.*length* $- k + 1$)

RANDOMIZED-K-LARGEST(A, p, r, i)
if $p == r$
    return A[$p, \dots,$ A.*length*]
$q =$ RANDOMIZED-PARTITION(A, p, r)
if $i == q$
    return A[$q, \dots,$ A.*length*]
elseif $i < q$
    return RANDOMIZED-K-LARGEST(A, p, $q - 1$, i)
return RANDOMIZED-K-LARGEST(A, $q + 1$, r, i)

Egenkonstruert sorteringsalgoritme

Oppgave 19: $\mathrm{A} = \langle a_1, a_2, \ldots, a_n \rangle$, $0 \leqslant a_i \leqslant n^3 - 1$. Finn en sorteringsalgoritme som sorterer vilkårlige A med en verste tilfelle kjøretid på $\mathrm{O}(n)$.

Egenkonstruert sorteringsalgoritme

Oppgave 19: $\mathrm{A} = \langle a_1, a_2, \ldots, a_n \rangle$, $0 \leqslant a_i \leqslant n^3 - 1$. Finn en sorteringsalgoritme som sorterer vilkårlige A med en verste tilfelle kjøretid på $\mathrm{O}(n)$.

```
SORT(A)
for i = 1 to A.length
      convert A[i] to base A.length
RADIX-SORT(A, 3)
for i = 1 to A.length
      convert A[i] to original base
```

Egenkonstruert sorteringsalgoritme

Oppgave 19: $\mathrm{A} = \langle a_1, a_2, \ldots, a_n \rangle$, $0 \leqslant a_i \leqslant n^3 - 1$. Finn en sorteringsalgoritme som sorterer vilkårlige A med en verste tilfelle kjøretid på $\mathrm{O}(n)$.

```
SORT(A)
for i = 1 to A.length
        convert A[i] to base A.length
RADIX-SORT(A, 3)
for i = 1 to A.length
        convert A[i] to original base
```

$\textsc{Radix-Sort}(\mathrm{A}, d)$ tar $\Theta(d(n+k))$ tid.

Egenkonstruert sorteringsalgoritme

Oppgave 19: $\mathrm{A} = \langle a_1, a_2, \ldots, a_n \rangle$, $0 \leqslant a_i \leqslant n^3 - 1$. Finn en sorteringsalgoritme som sorterer vilkårlige A med en verste tilfelle kjøretid på $\mathrm{O}(n)$.

```
SORT(A)
for i = 1 to A.length
    convert A[i] to base A.length
RADIX-SORT(A, 3)
for i = 1 to A.length
    convert A[i] to original base
```

$\textsc{Radix-Sort}(\mathrm{A}, d)$ tar $\Theta(d(n+k))$ tid.

Her er $\Theta(d(n+k)) = \Theta(3(n+n)) = \Theta(n)$

Finne maksimum

Oppgave 20: Finn en datastruktur som støtter innsetting av nye verdier, uthenting av maksimum og fjerning av maksimumsverdien i sublineær tidskompleksitet ($o(n)$).

Finne maksimum

Oppgave 20: Finn en datastruktur som støtter innsetting av nye verdier, uthenting av maksimum og fjerning av maksimumsverdien i sublineær tidskompleksitet ($o(n)$).

For å oppnå kun to av disse kan man opprettholde en sortert liste.

Finne maksimum

Oppgave 20: Finn en datastruktur som støtter innsetting av nye verdier, uthenting av maksimum og fjerning av maksimumsverdien i sublineær tidskompleksitet ($o(n)$).

For å oppnå kun to av disse kan man opprettholde en sortert liste.

En *maks-haug* oppnår sublineære tidskompleksitet for alle tre operasjonene.

Finne maksimum

Oppgave 20: Finn en datastruktur som støtter innsetting av nye verdier, uthenting av maksimum og fjerning av maksimumsverdien i sublineær tidskompleksitet ($o(n)$).

For å oppnå kun to av disse kan man opprettholde en sortert liste.

En *maks-haug* oppnår sublineære tidskompleksitet for alle tre operasjonene.

Oppgave 21: Er det mulig å ha en datastruktur som kan utføre alle disse operasjonene med $\mathrm{O}(1)$ tidskompleksitet?

Finne maksimum

Oppgave 20: Finn en datastruktur som støtter innsetting av nye verdier, uthenting av maksimum og fjerning av maksimumsverdien i sublineær tidskompleksitet ($o(n)$).

For å oppnå kun to av disse kan man opprettholde en sortert liste.

En *maks-haug* oppnår sublineære tidskompleksitet for alle tre operasjonene.

Oppgave 21: Er det mulig å ha en datastruktur som kan utføre alle disse operasjonene med $\mathrm{O}(1)$ tidskompleksitet?

Nei, da kan vi bruke denne til å sortere en vilkårlig liste med $\Theta(n)$ tidskompleksistet i verste tilfellet.