NTNU
Kunnskap for en bedre verden

KANDIDAT

10207

PRØVE

TDT4120 1 Algoritmer og datastrukturer

Emnekode	TDT4120
Vurderingsform	Hjemmeeksamen
Starttid	14.12.2021 08:00
Sluttid	14.12.2021 12:00
Sensurfrist	14.01.2022 22:59
PDF opprettet	31.01.2022 22:59

Seksjon 1

Oppgave	Tittel	Oppgavetype
ℹ	Forside	Informasjon eller ressurser
1	Oppgave 1	Langsvar
2	Oppgave 2	Langsvar
3	Oppgave 3	Langsvar
4	Oppgave 4	Langsvar
5	Oppgave 5	Langsvar
6	Oppgave 6	Langsvar
7	Oppgave 7	Langsvar
8	Oppgave 8	Langsvar
9	Oppgave 9	Langsvar
10	Oppgave 10	Langsvar

1 Oppgave 1

Hva er stabil sortering? Forklar kort med egne ord.

Skriv ditt svar her

Stabil sortering er når like elementer i en liste har samme rekkefølge etter sorteringen, som før sorteringen. Noen eksempler på stabile sorteringsalgoritmer er:
- Merge sort
- Counting Sort

2 Oppgave 2

Hva er topologisk sortering? Forklar kort med egne ord.

Skriv ditt svar her

Topologisk sortering kan kun finne sted i rettete asykliske grafer("dag"). Sorteringen skjer ved at hvis en kant (u,v) vil "u" havne før "v" i sorteringen. Dermed hvis grafen er syklisk vil man ikke kunne finne en rekkefølge.

3 Oppgave 3

Når vi finner minimale spenntrær, bruker vi den samme strategien i begge pensumalgoritmene. Forklar denne strategien kort med egne ord.

Merk: Her er vi ute etter noe mer spesifikt enn f.eks. grådighet.

Skriv ditt svar her

De to algoritmene for å finne minimale spenntrær er Prim og Kruskal. Strategien handler om å bestemme hva som er en trygg kant. Hver av algoritmene har sine egne regler for å finne den neste trygge kanten.

4 Oppgave 4

Hva er forskjellen mellom den sentrale egenskapen til et binært søketre og den sentrale egenskapen til en binærhaug? Forklar kort med egne ord.

Skriv ditt svar her

I et binært søketre vil en node sitt venstre "barn" vøre mindre enn noden, og høyre "barn" vil vil være større eller lik noden. Dette gjelder rekursivt gjennom hele treet. I hauger vil alle "barn" være mindre en roten, det gjør hauger lettere å bygge, men samtidig begrenser funksjonalitet.

5 Oppgave 5

Følgende algoritme har kjøretid $\Theta(n \lg n)$ i beste tilfelle og $\Theta(n^2)$ i verste:

```
LOREM(A)
1  n = A.length
2  QUICKSORT(A, 1, n)
3  for i = 1 to n
4        IPSUM(A, i)
```

Hva kan du si om kjøretiden til **Ipsum** i beste og verste tilfelle? Forklar kort.

Skriv ditt svar her

Beste:
Quicksort har i beste tilfelle kjøretiden $\Theta(n \lg n)$

Verste:
Quicksort har i verste tilfelle kjøretiden $\Theta(n^2)$

6 Oppgave 6

Forenkle følgende uttrykk:

$$\frac{\Omega(n^3)}{O(n^2)} + \frac{\Theta(n^5)}{\Theta(n^3)} + \frac{O(n^7)}{\Omega(n^4)}$$

Uttrykk svaret med asymptotisk notasjon. Forklar og diskuter kort.

Du kan anta at uttrykket er veldefinert.

Skriv ditt svar her

$$\frac{\Omega(n^3)}{O(n^2)} + \frac{\Theta(n^5)}{\Theta(n^3)} + \frac{O(n^7)}{\Omega(n^4)} = \Theta(n^5)$$

Man kan dele utrykket opp i tre leddene. Hvis man ser på hvert ledd kan man forenkle brøkene uten å miste nøyaktighet. Deretter kan man sammenligne de tre leddene å forenkle igjen uten å miste nøyaktighet.

7 Oppgave 7

Løs følgende rekurrens eksakt, med iterasjonsmetoden:

T(0) = 0
T(n) = T(n − 1) + lg(n/(n − 1)) (n ≥ 1)

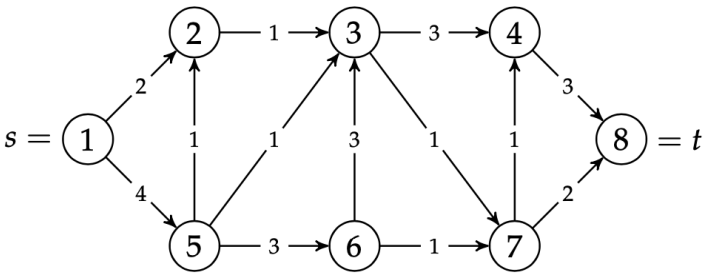
Oppgi svaret uten bruk av asymptotisk notasjon. Vis fremgangsmåten din. Vis hvordan du kan verifisere løsningen ved hjelp av induksjon (altså med substitusjonsmetoden).

Hint: Husk at lg(a · b) = lg a + lg b.
Skriv ditt svar her

$T(1) = 0$
$T(n) = \frac{\lg n}{\lg(n-1)} + T(n-1)$
(1) $= T(n-1) = 0$
(2) $= T(n-2) + \frac{\lg n}{\lg(n-2)}$
(?) $+ T(n-?)$

8 Oppgave 8

Hvilke noder vil **Ford-Fulkerson** traversere i siste iterasjon, i dette flytnettet?



Oppgi svaret ved å liste opp nodene i sortert rekkefølge. Forklar kort.

Hint: Den maksimale flytverdien er 5.
Skriv ditt svar her

{1,5,6,3,4,8}

9 Oppgave 9

La u og v være to forskjellige noder i en vektet, rettet graf G. La p være den korteste stien fra u til v, og la p' være den korteste av alle de stiene fra u til v som ikke inneholder negative sykler.

Betrakt følgende påstander:

1. Stien p vil aldri inneholde en negativ sykel.
2. Hvis G inneholder en negativ sykel, finnes ikke p.
3. Hvis G inneholder en negativ sykel, kan vi likevel finne p'.

Forklar om utsagnene stemmer eller ikke, ev. med hvilke unntak, antagelser eller forbehold.

Svar relativt grundig (f.eks. ca. 100–200 ord).

Du kan anta at p og p' er unike; ingen andre stier av samme type er like korte.

Skriv ditt svar her

1.
Stip vil aldri kunne inneholde en negativ sykel, fordi det vil ikke finnes en kortest vei hvis det er en negativ sykel.

2.
Hvis G inneholder en negativ sykel vil man ikke kunne finne korteste vei, da vil ikke p finnes. Dette problemet er NP-hardt.

3.
Hvis man spesifikt unngår negative sykler, kan man fortsatt finne korteste sti.

10 Oppgave 10

Konstruer og beskriv kort en effektiv algoritme som tar inn to rettede asykliske grafer og finner den lengste stien som forekommer i begge.

Om det er flere, skal du finne én av de lengste stiene; det er samme hvilken. Du får altså inn grafene $G = (V,E)$ og $G' = (V',E')$ og skal finne en lengst mulig sekvens $\langle v_1,\ldots,v_k \rangle$, der $(v_i,v_{i+1}) \in E \cap E'$ for $i=1\ldots k-1$.

Skriv ditt svar her

```
FIND-LONGEST(G, G'): //A = graf A, B= graf B
 commonEdges = [] //Liste over kanter som finnes i begge grafene
 for edge in G: //Finner kanter som finnes i begge grafene.
   if (edge in G'):
     commonEdges.append(edge)

 //Traverserer gjennom alle noder, og lagrer den lengste grafen
  longestPath = []
  for edge in edges:
     path = FIND_LENGTH(commonEdges, edge)
     if (size(path) > size(longestPath):
       longestPath = path

  return longestPath


FIND_LENGTH(edges, root):
  if not root.children: //Hvis roten ikke har noen barn har vi kommet til slutten
    return [root]

  longestPath = []

  for child in root.children:
    if (child in longestPath):
      continue //Hvis kanten allerede finnes in den lengste stien, har vi allerede oppdaget
           en lengre vei
    childPath = FIND_LENGTH(edges,child, path)
    if (size(childPath) > size(longestPath)):
      longestPath = childPath

  return longestPath
```

Siden grafen er rettet og asyklisk, trenger vi kun å sjekke barn, siden vi allerede vet at det ikke vil oppstå noen uendelige looper.

Steg:

1. Finner alle kanter som er like i begge grafene. Hvis det finnes unike kanter, kan vi ignorere de.
2. Finner lengste vei rekursivt fra hver kant
3. returnerer den lengste veien