TDT4120 Algoritmer og datastrukturer

Eksamen, 5. august 2024, 09:00–13:00

Faglig kontakt	Magnus Lie Hetland
Hjelpemiddelkode	E

Oppgaver

1 Det følgende er hentet fra ENQUEUE:

```
1  Q[Q.tail] = x
2  if Q.tail == Q.size
3       [redacted]
4  else Q.tail = Q.tail + 1
```

Hva skal den sensurerte biten være?

2 Anta at du kjører MST-PRIM og MST-KRUSKAL på en usammenhengende graf. Hvilken av algoritmene vil finne et minimalt spenntre for hver av de sammenhengende komponentene i grafen? Forklar kort.

Det vil si, hvilken av dem vil konstruere en usammenhengende løsning som dekker hele grafen?

3 En av løkkene i COUNTING-SORT går fra n ned til 1 (**for** $j = n$ **downto** 1). Hva er konsekvensen av å skifte retning på løkka (**for** $j = 1$ **to** n)?

Merk: Her kreves ingen forklaring.

4 Hvis du skal beskrive den beste kjøretiden til en algoritme (*best-case*), hvilken asymptotisk notasjon (av O, Ω eller Θ) er det best å bruke, om mulig?

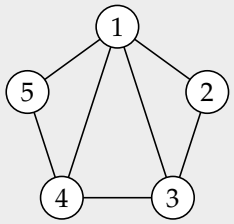
5 I en hashtabell med hashfunksjon h, hva vil det si at nøklene k_1 og k_2 kolliderer?

6 Hva er den amortiserte kjøretiden til TABLE-INSERT?

Det er altså snakk om innsetting i en dynamisk tabell, der vi enten kan sette elementet rett inn, om det er plass, eller må allokere en ny og større tabell ellers. Oppgi svaret med Θ-notasjon.

7 Du har en rettet, uvektet graf $G = (V, E)$, og skal finne korteste veier *fra alle noder* i V *til én gitt node* t. Hvordan vil du gå frem?

Figur 1 Graf til oppgave 10



8 Du ønsker å finne lengste enkle vei fra node s til node t i en vektet graf. Hvordan kan du gjøre det? Er det tilfeller der metoden ikke vil fungere? Forklar kort.

9 Løsningen på det binære ryggsekkproblemet (*0-1 knapsack*) har kjøretid $\Theta(n\mathrm{W})$, der n er antall gjenstander og W er kapasiteten til ryggsekken. Er dette en polynomisk algoritme? Forklar kort.

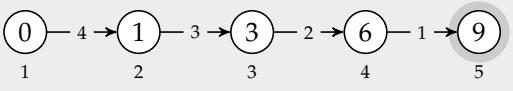
10 Du skal representere grafen i figur 1 som en nabomatrise. Fyll inn 0 og 1 i tabellen under.

	1	2	3	4	5
1					
2					
3					
4					
5					

11 Hva er et nodedekke (*vertex cover*)?

12 I pensumdefinisjonen av flytnett (*flow networks*) tillates ikke antiparallelle kanter, altså at vi har en kant både fra v_1 til v_2 og fra v_2 til v_1. Dersom vi likevel har slike kanter, hvordan kan vi håndtere det?

13 På tilsvarende måte som i Bellman-Ford, skal du utføre Relax på alle kantene i den følgende grafen én gang. Rekkefølgen er ikke gitt.



Når du er ferdig, hva er den minste og største verdien 5.d kan ha (altså $v.d$ for node 5, uthevet)? Oppgi svaret som to tall, adskilt med komma.

Hver nodes d-verdi før du starter er angitt i noden i figuren, så f.eks. 4.$d = 6$.

Du skal altså *ikke* utføre hele BELLMAN-FORD, men oppdatere estimatet én gang langs hver kant, i en eller annen rekkefølge.

14 Løs følgende rekurrens:

$T(n) = T(n-1) \cdot 2^{2^n} \qquad (n \geqslant 1)$

$T(0) = 2$

Oppgi svaret eksakt, dvs. uten asymptotisk notasjon.

15 I TRANSITIVE-CLOSURE angir t_{ij} om det finnes en sti fra i til j. Anta nå at den rettede grafen du får som input er asyklisk. Hvordan kan du endre algoritmen så t_{ij} blir *antall* stier fra i til j?

Du kan ev. beskrive løsningen din som en endring av FLOYD-WARSHALL.

16 Et problem med QUICKSORT er at kjøretiden blir dårlig om pivotelementet er dårlig. Kan man velge pivot slik at kjøretiden garantert blir $\Theta(n \lg n)$? Forklar.

Her er det snakk om å kun modifisere hvordan man velger pivot; resten av QUICKSORT skal utføres som normalt. Hvert rekursive kall skal også utføres på samme måte, så man kan ikke f.eks. bruke MERGE-SORT til å begynne med for å «jukse seg til» riktig kjøretid.

17 Du har n gjenstander og skal gi én til hver av n personer. Personene kan foretrekke ulike gjenstander. Helst vil du at ingen skal misunne noen andre, men du innser at det neppe er mulig.

I stedet lager du et lotteri for hver gjenstand. Heller enn å gi ut gjenstandene direkte, får hver person en tilfeldig *prioritet* for hver gjenstand. Målet ditt er at ingen skal misunne noen som har lavere prioritet.

Vil det alltid være mulig å fordele gjenstandene slik? Hvordan?

Om du har fått gjenstand x, så skal jeg altså ikke misunne deg, med mindre jeg har lavere prioritet for gjenstand x.

18 Din venn Lurvik studerer to beslutningsproblemer, A og B, der han har en eksponentiell algoritme for A og en polynomisk algoritme for B. Han har vist at A ikke kan løses raskere enn eksponentielt.

Lurvik har også funnet reduksjoner fra A til B og fra B til A. Hva kan du si om kjøretiden til hver av disse reduksjonene? Forklar kort.

Om vi ser på A og B som formelle språk, har Lurvik altså funnet to reduksjonsfunksjoner f og g, der

$x \in \mathrm{A}$ hvis og bare hvis $f(x) \in \mathrm{B}$ og

$x \in \mathrm{B}$ hvis og bare hvis $g(x) \in \mathrm{A}$.

Algoritme 1 Inversen av ZIP

```
UNZIP(x)
 1  if x == NULL
 2      let L, R be new lists
 3  else allocate new nodes y and z
 4      y.key = x.key[1]
 5      z.key = x.key[2]
 6      L = UNZIP(x.next)[1]
 7      R = UNZIP(x.next)[2]
 8      LIST-PREPEND(L, y)
 9      LIST-PREPEND(R, z)
10  return ⟨L, R⟩
```

Spørsmålet er hva du kan si om kjøretiden som kreves for å beregne reduksjonsfunksjonene f og g.

I pensum antas en reduksjon generelt å ha polynomisk kjøretid, men her kan du se bort fra det.

19 I mange programmeringsspråk har man en funksjon som heter ZIP, som tar inn to lister, og returnerer en liste av par, der par i består av element i fra hver av de to listene.

Prosedyren UNZIP (algoritme 1) gjør det motsatte. Den tar inn hodet til en lenket liste (*linked list*) av par (tabeller av lengde 2) og fordeler dem i to lister L og R.

Hvordan ville du ha endret algoritmen for å forbedre kjøretiden? Hva blir kjøretiden før og etter forbedringen din? Forklar.

20 COUNTING-SORT(A, n, k) tar inn en tabell $\mathrm{A}[1:n]$ med heltall i området $0, \ldots, k$ og fyller en tabell $\mathrm{C}[0:k]$ med antall forekomster i A av hver mulige verdi, og bruker $\Theta(n+k)$ operasjoner på dette.

Du skal nå gjøre en lignende telling, der A allerede er sortert. Du kan anta at du også får inn C som parameter, og at C er initialisert, så $\mathrm{C}[i] = 0$ for $i = 0, \ldots, k$.

Bruk metoden *splitt og hersk* til å konstruere en algoritme som løser problemet med kjøretid $\mathrm{O}(n)$ generelt, men som er raskere enn dette når A inneholder mange duplikater.