NTNU

Kunnskap for en bedre verden

Institutt for datateknologi og informatikk

Eksamensoppgave i Algoritmer og datastrukturer, IDATT2101

Eksamensdato: 15. desember 2023

Eksamenstid (fra–til): 09:00–13:00

Tillatte hjelpemiddel: ett A4-ark med notater

Faglig kontakt under eksamen: Helge Hafting

Tlf.: 924 386 56

Annen informasjon: Løsningsforslag

Målform/språk: bokmål

Antall sider (uten forside): 6

Antall sider vedlegg: 0

Informasjon om trykking av eksamensoppgave			
Originalen er:			
1-sidig	☐	**2-sidig**	☒
sort/hvit	☐	**farger**	☒
		Flervalgskjema?	☐

Kontrollert av

....

Dato Sign

Merk! Studentene finner sensur i Studentweb. Har du spørsmål om sensuren må du kontakte instituttet ditt. Eksamenskontoret vil ikke kunne svare på slike spørsmål.

Oppgave 1 20%

Analyser disse programmene. Bruk Θ om mulig. Om ikke, bruk O og Ω. Alle parametre er positive.

```
int prog_a(int a, int b, int c) {
    int sum = 0;
    for (int i=1; i<a ; ++i) {
       for (int j=1; j<c; ++j) {
          for (int k=a; k>0; --k) {
            sum += i*j-k;
            if (sum > b) return;
          }
       }
    }
    return sum;
}
```

```
int prog_b(int q, int r, int p) {
    int sum = 0;
    for (int i=0; i<q ; i += 1) {
       sum += i*r;
       if (sum > p) sum -= p;
    }
}
```

```
int prog_c(int n, int [] tab) {
    int sum = 0;
    if (n > 0) {
       sum += 4 * prog_c(n/2, tab);
       for (int i=0; i<n; ++i) sum += tab[i];
       tab[0]--;
       sum += 4 * prog_c(n/2, tab);
    }
    return sum;
}
```

```
double prog_d(int n, float x) {
    if (n == 0) return 0.0;
    else return x + prog_d(n - 1, x);
}
```

```
int prog_e(int a, int b, int c) {
    int sum = 1;
    if (a < b) {
        for (int i = 1; i < a; ++i) sum *= i;
        for (int j = a; j < b; ++j) sum += j;
    }
    return sum;
}
```

a: $O\left(a^2c\right)$, $\Omega(a)$ muligens $\Omega(ac)$, begge godtas.

b: $\Theta\left(q\right)$c: $\Theta\left(n \log n\right)$ d: $\Theta\left(n\right)$ e: $\Omega(1)$, $O\left(b\right)$

Oppgave 2 — 20%

a) Jeg bruker en heap som prioritetskø. Den inneholder n elementer. Hva blir kjøretidene for disse tre operasjonene: å sette inn et element til, å ta ut elementet på toppen, å endre prioritet for et element.

sette inn $O\left(\log n\right)$ $\Omega\left(1\right)$

hent_min $O\left(\log n\right)$ $\Omega\left(1\right)$

ny_prio $O\left(\log n\right)$ $\Omega\left(1\right)$

b) Dijkstras algoritme trenger en prioritetskø. Algoritmen brukes på en graf med N noder og K kanter. Vi kan velge en heap som prioritetskø, eller bruke en usortert tabell.

Vil et av valgene alltid være best? Eller vil dette avhenge av grafen på noe vis? Forklar, bruk gjerne kjøretider som funksjon av N og K.

Dette avhenger av grafen.

Dijkstra henter N noder ut av køen, og endrer i verste fall prioritet K ganger. Med heap er hver operasjon $O\left(\log N\right)$, så summen blir $O\left((N+K)\log N\right)$. Med usortert tabell vil det å hente en node koste $O\left(N\right)$ tid, mens prioritetsendring går i $O\left(1\right)$ tid. Summen blir $O\left(K+N^2\right)$

Hvis K er proporsjonal med N, blir heap $O(N \log N)$, bedre enn tabellens $O\left(N^2\right)$. Dette gjelder for eksempel for veisystemer.

Hvis K er nærmere N^2, vil derimot $O(N^2 \log N)$ være verre enn $O\left(N^2\right)$. For slike tette grafer vil usortert tabell lønne seg bedre.

Oppgave 3 — 15%

a) Velg enten Lempel-Ziv eller Lempel-Ziv-Welsh, og forklar kort hvordan algoritmen komprimerer data.

LZ

Leser seg gjennom fila. For hver tegnposisjon, sjekk om det som ligger der matcher med en viss minstelengde noe sted tidligere i fila. I så fall, skriv en bakoverreferanse som er kortere

enn denne minstelengden. Hvis ingen match, skriv tegnet til output. Uansett, fortsett med neste posisjon.

For å kunne dekode, må vi også ha data som forteller når det kommer en sekvens ukomprimerte tegn.

Kompresjon oppnås fordi minstelengden er satt lenger enn størrelsen på en bakovverref.

LZW

Har en ordliste som i starten inneholder de mulige tegnene som «ord». Leser seg gjennom fila. For hver posisjon, output nummeret til ordet som ligger der, med lengst mulig match. Legg til lovende nye ord i ordlista etterhvert som de passeres, slik at vi kan matche lengre ord etterhvert. Utpakker kan gjøre det samme, og få generert samme ordliste.

Kompresjon oppnås om vi får ord som er lengre enn sine numre, noe som skjer hvis det er en del gjentatte ord.

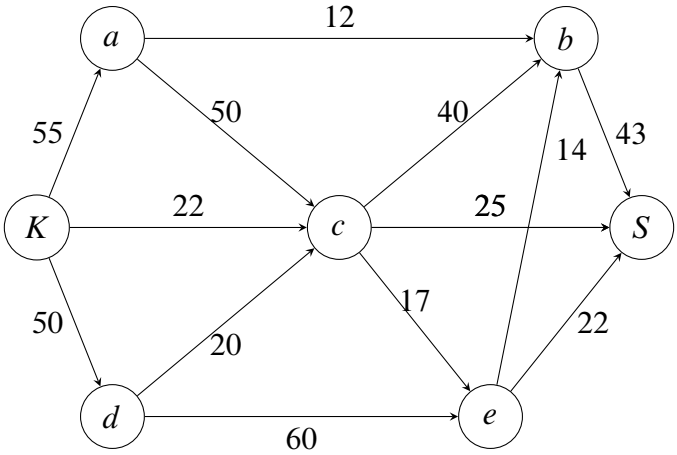
b) A*-algoritmen er en videreutvikling av Dijkstras algoritme. Hva er det A* gjør anderledes? Forklar.

Dijkstras algoritme prioriterer kun ut fra avstand til startnoden. Dermed ekspanderer søket som en voksende sirkel rundt start.

A* prioriterer med summen av avstand til startnoden, og estimert avstand til mål. (Estimatet bør være best mulig, men *aldri* for høyt.) Dermed ekspanderer søket mer i retning av målet, og mindre i andre retninger. Håpet er at dette fører til mindre arbeid og raskere beregning.

Oppgave 4 **30%**

Bruk denne grafen.



a) Finn maksimal flyt fra K til S. Nytt flytøkende veier, og skriv opp hver vei og hvor mye flyt du legger til langs veien.

Mulige flytøkende veier:

```
Maksimum flyt fra 0 til 6 med Edmond-Karp
Økning : Flytøkende vei
    22   K c S
    22   K d e S
     3   K d c S
    12   K a b S
    14   K d e b S
    11   K d c b S
     6   K a c b S
Maksimal flyt ble 90
```

Det er mulig å ha andre flytøkende veier enn dette, men summen må bli 90.

b) Sorter grafen topologisk, eller forklar hvorfor det ikke er mulig.

Det er to korrekte rekkefølger for topologisk sortering.

KadcebS
KdacebS

c) Se bort fra retningen på kantene, og finn et minimalt spenntre for grafen. Skriv opp hvilke kanter som blir med, og total vekt på spenntreet.

Kc:22, dc:20, ce:17, eS:22, eb:14, ab:12. Samlet vekt 107.

d) Finn et annet minimalt spenntre i denne grafen, eller forklar hvorfor det ikke er mulig.

Det er ikke mulig. Hvis vi f.eks. bruker Kruskals algoritme, må vi aldri velge mellom kanter med lik vekt. De få kantene med lik vekt, er enten begge med eller begge unntatt. Ingen andre spenntrær kan være minimale.

Et annet like godt argument: Etter å ha lagd det minimale spenntreet, ser vi at alle gjenværende kanter har høyere vekt enn de i spenntreet. Det er dermed ikke mulig å bytte ut kanter uten å øke vekten, og derfor finnes ikke noe annet spenntre i denne grafen.

Oppgave 5 — 15%

a) Forklar kort hvordan quicksort virker.

Quicksort deler tabellen i to, en del med små nøkler og en del med store. Nøkler som står feil, byttes over til den andre delen.

Når dette er gjort, har vi en deltabell med små nøkler og en med store. De to delene har rett rekkefølge i forhold til hverandre, men kan ha uorden internt. Delene sorteres rekursivt med quicksort, og deles videre opp til hver deltabell har størrelse 1. Da er hele tabellen sortert.

b) Fortell om hvordan quicksort og innsettingssort kan kombineres til en bedre/raskere sorteringsalgoritme. Hvorfor blir den kombinerte algoritmen raskere?

Overlat tilstrekkelig små deltabeller til innsettingssort. Raskere fordi innsettingssort er raskest på tilstrekkelig små tabeller. (Enklere kode, ingen rekursjon.)

c) Tellesortering egner seg ikke for desimaltall (float/double). Hvorfor ikke?

Tellesortering bruker tallene som indeks i en tabell. Det fungerer ikke med desimaltall: 5,4 og 5,3 er ulike, men vil velge samme posisjon i tabellen likevel. De kan dermed komme ut i feil rekkefølge.