Linköpings Tekniska Högskola
Institutionen för Datavetenskap
Filip Strömbäck och Magnus Nielsen

29 oktober 2021

TDDI16
Datastrukturer och algoritmer
Datortentamen (DAT1)
2021-10-29, 14–18

Examinator:	Filip Strömbäck
Jour:	Filip Strömbäck (telefon 013-28 27 52)
Antal uppgifter:	6
Max poäng:	40 poäng
Preliminära gränser:	Betyg 5 = 35p, 4 = 27p, 3 = 20p.
Hjälpmedel:	Inga hjälpmedel tillåtna!

VÄNLIGEN IAKTTAG FÖLJANDE

- Observera att betygsgränserna kan komma att justeras, i samtliga kurser.
- Vid frågor om tidskomplexitet, svara alltid på den form som är mest relevant.
- Skriv dina lösningar i valfri textredigerare, exempelvis LibreOffice, Emacs, eller liknande. Strukturera din text så att det enkelt går att se vad som svarar på vilka deluppgifter.
- Lösningar till olika problem skall skrivas i en egen fil. Skriv inte två lösningar i samma fil. Delproblem får dela fil.
- Om inte annat framgår ska indexering av arrayer/listor börja från 0.
- Skicka in som lösning till rätt problem i tentaklienten, och döp filen till ett passande namn (exempelvis uppg1.txt/uppg1.odt).
- **MOTIVERA DINA SVAR ORDENTLIGT:** avsaknad av, eller otillräckliga, förklaringar resulterar i poängavdrag. **Även felaktiga svar kan ge poäng** om de är korrekt motiverade.
- Om ett problem medger flera olika lösningar, t.ex. algoritmer med olika tidskomplexitet, ger endast optimala lösningar maximalt antal poäng.
- SE TILL ATT DINA LÖSNINGAR/SVAR ÄR LÄSLIGA. Oläsliga lösningar beaktas ej.

Lycka till!

1. **Hashtabeller** (5 p)

Vi har en hashtabell med linjär adressering och några element instoppade. Hashfunktionen är h(x) = x mod size (arrayindex står under arrayen för tydlighets skull).

30	8	Null	13	24	Null	26	6	18	19
0	1	2	3	4	5	6	7	8	9

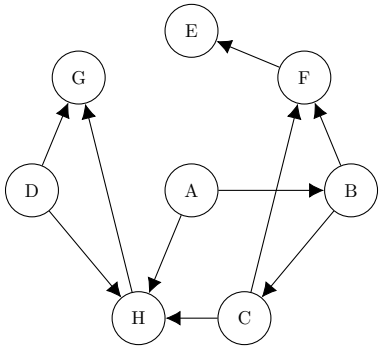
(a) I vilken ordning har elementen stoppats in? Det finns flera lösningar. Ge fyra korrekta lösningar för maxpoäng. (2)

(b) Om vi tar bort 26 från den delvis fyllda hashtabellen ovan (remove / delete), hur kommer tabellen se ut? Det finns flera olika lösningar. Ge två olika lösningar som inte hashar om hela tabellen utom möjligen i värsta fallet, och förklara dem, för maxpoäng. (2)

(c) Ett program sätter in fyra till element i hashtabellen ovan (exempelvis 58, 12, 33 och 78). Alla insättningarna lyckas. Beskriv kort (2-3 meningar) hur implementationen kan ha gjort för att få plats med de nya elementen. (1)

2. **Grafer** (4 p)

Betrakta den riktade grafen nedan:



(a) Gör en breddenförst-traversering av grafen ovan med start i noden A. Antag att grannarna till en nod besöks i alfabetisk ordning. Ange i vilken ordning noderna besöks. Ingen motivering krävs. (2)

(b) Gör en djupetförst-traversering av grafen ovan med start i noden A. Antag att grannarna till en nod besöks i alfabetisk ordning. Ange i vilken ordning noderna besöks. Ingen motivering krävs. (2)

3. **Sorteringsalgoritmer** (5 p)

Svaren behöver ej motiveras.

Efter **ett fåtal** iterationer av några olika sorteringsalgoritmer på den osorterade arrayen *Original* i tabellen nedan (iteration i bemärkelse fullständig körning av inre loop, alternativt rekursivt anrop) har vi resultaten i *1, 2, 3, 4* och *5*.

Original:	30	56	49	75	80	71	45	66	23	58	67	92	84	6	54
1:	6	23	30	45	56	49	75	80	71	54	66	58	67	84	92
2:	75	67	71	66	58	49	54	6	23	45	56	30	80	84	92
3:	30	49	56	75	80	71	45	66	23	58	67	92	84	6	54
4:	6	23	30	75	80	71	45	66	56	58	67	92	84	49	54
5:	30	6	23	45	49	54	56	66	67	58	71	92	84	80	75
Sorterad:	6	23	30	45	49	54	56	58	66	67	71	75	80	84	92

Matcha delvis sorterad array mot en algoritm. Felaktig matchning ger minuspoäng, dock kan uppgiften ej ge total minuspoäng. Utelämnat svar ger ej minuspoäng. För full poäng krävs 5 korrekta svar.

(a) Bubble sort, element bubblas från höger till vänster

(b) Heapsort

(c) Insertionsort

(d) Quicksort, elementet längst till höger i partitionen används som pivot

(e) Selectionsort

4. **Algoritmer och tidskomplexitet** (6 p)

Visa kort hur du kommer fram till tidskomplexiteten på följande uppgifter.

(a) Beräkna tidskomplexiten med avseende på parametern n för följande funktion: (1)

```
int fun(int n) {
  int result = 0;
  for (int i = 0; i < n; i += 2) {
    result += i * i;
  }
  return result;
}
```

(b) Beräkna tidskomplexiten med avseende på parametern n för följande funktion: (1)

```
int funnier(int n) {
  int result = 0;
  for (int i = 1; i < n; i *= 2) {
    for (int j = 1; j < n; j++) {
      result += i * j;
    }
  }
  return result;
}
```

(c) Beräkna tidskomplexiten med avseende på parametern n för följande funktion. Beskriv också kortfattat hur du kom fram till ditt svar. (2)

```
int funniest(int n) {
  int result = 0;
  for (int i = 0; i < n; i++) {
    result += fun(i);
  }
  return result;
}
```

(d) Beräkna tidskomplexiten med avseende på parametern n för följande funktion. Beskriv också kortfattat hur du kom fram till ditt svar. (2)

```
int boring(int n) {
  int result = funnier(n);
  for (int i = 0; i < n; i++) {
    if (i == 2 || i == n / 2) {
      result += fun(n);
    } else {
      result += 1;
    }
  }
  return result;
}
```

5. **Energimedelvärdesberäkningsprogram** (10 p)

Som ni vet är inte alla föreläsare jättebra på huvudräkning. Som tack för en bra kurs vill du därför skriva ett program som hjälper din föreläsare att lösa problemen som denne annars hade behövt använda huvudräkning till.

Din föreläsare har på sista tiden varit mycket intresserad av att mäta energiförbrukningen av sin dator (för att hitta de algoritmerna som är mest energieffektiva). För att göra det använder vederbörande en digital mätare som är kopplad till datorn. Från den kommer mätvärden (i Watt) en gång i sekunden. Föreläsaren tittar sedan på dem och beräknar medelvärdet för den senaste intressanta tidsperioden med hjälp av huvudräkning (vilket oftast inte går så bra...). Din föreläsare behöver alltså ett program som kan göra följande:

- När programmet startas anger föreläsaren k som motsvarar hur många sekunder programmet ska mäta medelvärdet för. Programmet kan anta att förbrukningen de k sekunderna innan det startade var 0 Watt.
- Programmet ska sedan ta emot alla mätvärden från mätaren (1 i sekunden).
- För varje mätvärde ska sedan programmet mata ut medelvärdet för de senaste k sekunderna.

(a) Vilken eller vilka datastrukturer väljer du att lagra data i programmet? Motivera kort (1-4 meningar) varför denna/dessa är lämpliga i din lösning. (2)

(b) Beskriv hur du initierar dina datastrukturer när föreläsaren startar programmet och anger ett k. (1)

(c) Beskriv vad programmet gör varje gång det tar emot ett nytt mätvärde (x) för att lagra det i datastrukturen och beräkna medelvärdet för de senaste k mätvärdena. Beskriv lämpligtvis algoritmen stegvis som i exemplet nedan. Tänk på att vi är ute efter en algoritm med bra tidskomplexitet. (3)

> **Exempel** (relaterat till uppgiften, men löser såklart inte uppgiften):
>
> i. Läs in nästa mätvärde i x
>
> ii. Lagra det sist i arrayen a
>
> iii. Skriv ut element 10 ur a

(d) Vad har din algoritm för tidskomplexitet för att ta emot **ett** nytt mätvärde och beräkna ett nytt medelvärde baserat på de k senaste mätvärdena? (1)

(e) Vad har din algoritm för tidskomplexitet för att ta emot n mätvärden, och för varje mätvärde beräkna och skriva ut medelvärdet för de senaste k mätvärdena? (2)

(f) Hur mycket minne använder ditt program i fallet som beskrivs i (e), uttryckt i k och/eller n? (1)

6. **Snöplogar i Skottland** (10 p)

Snön ligger tät i Skottland, endast snöplogen är vaken. Dessutom har de bra namn.[1]

Du är vicesnöplogsgarageintendent i Glasgow. Din uppgift är att se till så att det går att ta sig fram mellan alla städer i Skottland. Till din hjälp har du ett garage fullt av snöplogar och en lista över alla större vägar i Skottland. Informationen du har ser ut som listorna nedan.

Från	**Till**	**Längd**
Glasgow	Edinburgh	76,5 km
Glasgow	Inverness	270 km
Glasgow	Perth	102 km
Perth	Inverness	181 km
Perth	Edinburgh	72,1 km
Perth	Dundee	35,2 km
...	...	...

Plognamn
BFG (Big Friendly Gritter)
Han Snow-lo
Gangsta Granny Gritter
Snowbegone Kenobi
Snowkemon Go
Spready Mercury
Sprinkelbell
...

Det har nyligen varit ett stort snöoväder i Skottland, så som vicesnöplogsintendent måste du se till att det snabbt går att ta sig fram i landet igen. Därmed har du beslutat att du måste prioritera att det går att ta sig fram överhuvudtaget, snarare än att den bästa vägen mellan alla städer är skottad (dvs. det ska gå att ta sig fram mellan två städer X och Y, men den kortaste vägen behöver inte vara skottad). För att detta ska vara genomförbart på kort tid vill du skotta så kort vägsträcka som möjligt. Garaget finns i Glasgow, och det är också där alla snöplogarna finns.

(a) Du skriver ett program för att beräkna vilka vägsträckor som ska plogas. Beskriv din algoritm (stegvis, likt i uppgift 5). (3)

(b) Vad har din algoritm i (a) för tidskomplexitet? (1)

(c) Givet att du i (a) har hittat en delmängd vägar som du vill ploga, vill du nu besluta hur många plogar du ska skicka. För att skotta alla vägar så snabbt du kan vill du skicka en plog till varje stad som bara ska få en skottad väg till sig (enligt lösningen i (a)). På det viset behöver plogarna bara åka åt ett håll och inte vända, vilket tar onödig tid i din mening. (3)

För att kunna informera alla plogar hur de ska åka vill du alltså beräkna hur många plogar som behöver åka längs varje plogat vägsegment för att kunna ploga alla vägar utan att någon snöplog behöver vända. Du skriver ett program för att beräkna även detta problem. Beskriv din algoritm (stegvis, likt tidigare).

(d) Vad har din algoritm i (c) för tidskomplexitet? (1)

(e) Dagen därpå kommer du på att lösningen i (a) och (c) inte minimerar sträckan som snöplogarna behöver åka, trots att den plogade vägsträckan är minimerad. Varför? (2)

[1] `https://www.heraldscotland.com/news/18940013.scotland-gritter-names-snowfleet-goes-viral-twitter-incredible-names/`