Linköpings Tekniska Högskola
Institutionen för Datavetenskap
Filip Strömbäck och Magnus Nielsen

28 november 2022

TDDI16
Datastrukturer och algoritmer
Datortentamen (DAT1)
2022-10-28, 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.
- 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** (4 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).

0	8	2	22	13	Null	16	Null	28	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 28 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)

2. **Sorteringsalgoritmer** (4 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* och *4*.

Original:	79	24	91	0	33	96	26	32	88	18	65	71	60	9	68
1:	0	9	18	79	33	96	26	32	88	91	65	71	60	24	68
2:	9	0	18	24	33	65	26	32	60	68	79	71	88	96	91
3:	0	24	79	91	33	96	26	32	88	18	65	71	60	9	68
4:	79	65	71	32	33	60	68	24	0	18	9	26	88	91	96
Sorted:	0	9	18	24	26	32	33	60	65	68	71	79	88	91	96

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 4 korrekta svar.

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

(b) Heapsort

(c) Insertionsort

(d) Selectionsort

3. **Onlinecasinokonsultation** (8 p)

Onlinecasinot AllYourMoneyAreBelongTo.us har just öppnat portarna. Efter en mycket lyckad lansering har de fått massor av kunder, vilket gjort att spelen ibland laggar och det kan vara svårt att komma in. Istället för att expandera och åtgärda problemen har de valt att begränsa antalet aktiva spelare till 1000 åt gången. För att komma in måste man ställa sig i kö. Man kan dessutom köpa ett VIP-medlemsskap som gör att man får gå före vanliga medlemmar i kön. Därutöver ger man anställda företräde, till och med över VIP-medlemmar.

- `enqueue(user, level)`: Ställ användaren `user` i kö. `level` anger om användaren är en vanlig användare (0), en VIP-medlem (1), eller en anställd (2).
- `dequeue()`: Anropas när nya användare kan släppas in på servern. Returnerar användaren som är näst på tur.

Designteamet på företaget har kommit fram till följande tre implementationer. Eftersom de inte vet vilken som kommer att fungera bäst har de bett dig att utvärdera alternativen.

1: Lagra data i en dynamisk array. Varje element är en tupel, `(user, level)`. `enqueue` sätter in den nya användaren i slutet av arrayen. `dequeue` sorterar först arrayen med hjälp av quicksort (`level` används som nyckel). Sedan tas första elementet ur arrayen bort, och användaren som var där returneras.

2: Lagra data i en heap (med dynamisk storlek). Varje element är en tupel, `(user, level)`, men endast `level` används som nyckel för heapen. `enqueue` sätter helt enkelt in användaren i heapen, och `dequeue` tar bort rotnoden ur heapen och returnerar användaren på den platsen.

3: Lagra data i tre (dynamiska) köer. Köerna är numrerade 0, 1, och 2. De innehåller endast `user`. `enqueue` sätter in användaren i kö nummer `level`. `dequeue` undersöker köerna i ordningen 2, 1, 0 och plockar en användare från den första kön som inte är tom.

Svara på följande frågor med avseende på de tre alternativen ovan:

(a) Vilken tidskomplexitet har `enqueue` för vart och ett av alternativen 1–3 ovan? Beskriv kort ditt resonemang. (3)

(b) Vilken tidskomplexitet har `dequeue` för vart och ett av alternativen 1–3 ovan? Beskriv kort ditt resonemang. (3)

(c) Fungerar alla alternativ korrekt? För alla inkorrekta alternativ, beskriv kort vad som är fel. (2)

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 = 3; i < n; i = i * 2) {
    result += i * 3;
  }
  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*n; ++i) {
    for (int j = 1; j < n; ++j) {
      result += -1;
    }
    result += i;
  }
  return result;
}
```

(c) Beräkna tidskomplexiten med avseende på parametern n för följande funktion: (2)

```
int funnieast(int n) {
  if (n > 0)
    return 1 + funnieast(n / 2);
  else
    return 0;
}
```

(d) Beräkna tidskomplexiten med avseende på storleken av x för följande funktion. Antag att `Array` är en dynamisk arraylista. Tänk på bästa- och värstafall. Beskriv också kortfattat hur du kom fram till ditt svar. (2)

```
Array<int> funniwest(Array<int> x) {
  for (int i = 1; i < x.length(); i++) {
    if (x[i-1] > x[i]) {
      x.remove(i-1);   // Ta bort element med index i-1.
    }
  }
  return x;
}
```

5. **Datorbutiken i Troja** (8 p)

En datorbutik i stan, Trojan Computing, har nyligen fått erfara större IT-problem. Deras POS-system (Point of Sales) har nyligen kraschat. Det fungerar fortfarande så mycket att det kan lagra nya transaktioner, men det kan inte hämta dem igen. Det värsta med detta problem är att skärmen med personalens highscore i fikarummet inte längre fungerar. Tyvärr får de inte tag på de som byggde systemet, och ingen har lyckats förstå sig på hur systemet lagrade information om de transaktioner som har gjorts under systemets livstid. En av de anställda lyckades dock gräva fram lite data med ett stort reguljärt uttryck, så att transaktionerna i alla fall är läsbara. Dock är de inte ordnade på något sätt, eftersom programmet använde någon ”smart” datastruktur som ingen har förstått sig på för att ordna datan internt.

Du, som nyutexaminerad DALG-student, har fått i uppgift att sammanställa datan och få liv i skärmen med highscores igen. Skärmen uppdateras varje minut, så din lösning måste vara effektiv. Skärmen visar dels det totala värdet av alla transaktioner för varje säljare, och antalet transaktioner för varje säljare. Skärmen visar också en scrollande lista över transaktioner för varje säljare.

Ditt program måste alltså producera följande för varje säljare: en lista av alla transaktioner för säljaren, summan av beloppet för transaktionerna, och antalet transaktioner.

Exempel på data:

ID	Datum	Belopp	Säljare	Beskrivning
8	2022-09-01	5000	Hanna	Processor och moderkort
10	2022-09-28	300	Magnus	Tentakaffe
5	2022-09-25	250	Filip	Tentagodis
215	2022-05-09	4500	Elin	Nästan virusfri dator
17	2022-08-05	400	Axel	2 GHz (lösplock)
31	2022-09-18	500	Janos	4 GB DDR4 (lösplock)
78	2022-03-18	0	Hanna	Installation av bakdörr (inte baklucka)
15	2022-06-13	500	Elin	Trojansk häst (uppstoppad)
999	2025-12-31	1000	Janos	Fluxkondensator
...	...	...	...	...

Svara på följande frågor:

(a) Beskriv hur du kan implementera programmet som hämtar data till skärmen, enligt beskrivningen ovan. Fokusera särskilt på de datastrukturer du använder för att lagra data, och de algoritmer du behöver för att manipulera datastrukturerna. (4)

(b) Vilken tidskomplexitet har din implementation, uttryckt i antalet rader i tabellen, n? (1)

(c) Vilken minneskomplexitet har din implementation, uttryckt i antalet rader i tabellen, n? (1)

(d) Efter ett tag ber datorbutiken dig om en ny feature till skärmen. De vill också att den ska kunna visa en topplista över de k dyraste transaktionerna (k är mycket mindre än n). För att hinna uppdatera skärmen inser du att ditt program måste kunna göra detta i $\mathcal{O}(n \log n)$. Du inser också att du bara har plats att lagra $\mathcal{O}(k)$ element i ditt program (dvs. inte alla element i tabellen). Du kan inte heller ändra tabellen. Beskriv hur du kan lösa problemet inom de givna ramarna. (2)

6. **Assistentanställningsautomatisering** (10 p)

För att undervisa kurser på universitetet krävs en smärre armé av assistenter. Det största problemet med detta är att det tar mycket tid att anställa bra assistenter. På grund av detta är kursledningen intresserade av att automatisera processen med hjälp av big data från LinkedIn!

Idén är att utnyttja kompetenser på LinkedIn för at utvärdera hur kompetent en person är, och personens kontaktnätverk för att utvärdera hur mycket det går att lita på att kompetenserna är korrekta. För att illustrera processen, beakta tabellerna här nedanför:

Lista över personer

Namn	Kompetenser
Filip	C++, Java, DALG, Storm, ...
Magnus	Java, Ada, C++, DALG, ...
Christoffer	C++, JavaScript, PHP, Java, DALG, ...
Anna	C++, JavaScript, PHP, ...
Fredrik	DALG, Java, ...
...	...

Lista över kontakter

Person	Person
Filip	Magnus
Magnus	Christoffer
Filip	Christoffer
Christoffer	Anna
Magnus	Fredrik
...	...

Antag att Anna har ansökt om att vara med i DALG-kursen med Filip som examinator. För att vara med i kursen anser examinatorn att kunskaper inom Java, C++, och DALG är viktiga. För att utvärdera hur väl Anna passar beräknar vi en TrustMultiplier™ på följande sätt: Vi hittar en eller flera kedjor från Filip till Anna. Det finns åtminstone två sådana: Filip ⇒ Christoffer ⇒ Anna, samt Filip ⇒ Magnus ⇒ Christoffer ⇒ Anna. Den totala TrustMultiplier™ blir då den kedjan med lägst TrustMultiplier™. Varje kedjas TrustMultiplier™ är summan av TrustMultiplier™ för varje steg i kedjan. Denna beräknas enligt följande:

$$s = 1 - \frac{k}{t}$$

Där k är antalet kompetenser som personen uppfyller, och t är det totala antalet kompetenser som efterfrågas. I exemplet ovan, är TrustMultiplier™ för Filip ⇒ Christoffer alltså $1 - \frac{3}{3} = 0$ eftersom alla kompetenser uppfylls av Christoffer. För Christoffer ⇒ Anna är den $1 - \frac{1}{3} = 0.66$, då endast 1 av 3 efterfrågade kompetenser uppfylls. För kedjan Filip ⇒ Christoffer ⇒ Anna är TrustMultiplier™ alltså $0 + 0.66 = 0.66$, vilket i det här exemplet råkar vara minimalt.

Med hjälp av denna metrik kan examinatorn sedan rangordna alla ansökande efter deras TrustMultiplier™, och välja det sökta antalet med lägst TrustMultiplier™.

Det är din uppgift att implementera en funktion som beräknar TrustMultiplier™ givet en examinator, en sökande, de kompetenser som är viktiga, och data motsvarande tabellerna ovan (ange hur du väljer att lagra den datan i ditt svar).

(a) Beskriv hur du kan implementera programmet som beräknar TrustMultiplier™ enligt beskrivningen ovan. Fokusera särskilt på de datastrukturer du använder för att lagra data, och de algoritmer du behöver för att manipulera datastrukturerna. (6)

(b) Vilken tidskomplexitet har din implementation, uttryckt i antalet personer som finns (p) och antalet kontakter som finns (e). (1)

(c) Vilken minneskomplexitet har din implementation, uttryckt i antalet personer som finns (p) och antalet kontakter som finns (e). (1)

(d) Givet att din funktion i stället för att få in en kandidat får in en lista över n kandidater. Kan du då effektivisera din implementation, så att den blir mer effektiv än n stycken anrop till den förra versionen? I så fall, hur gör du, och vad blir den nya tidskomplexiteten? (2)