Linköpings Tekniska Högskola
Institutionen för Datavetenskap
Filip Strömbäck och Magnus Nielsen

29 oktober 2020

TDDI16
Datastrukturer och algoritmer
Datortentamen (DAT1)
2020-10-29, 14–18

Examinator:	Filip Strömbäck
Jour:	Filip Strömbäck (Zoom: 623 8655 4242, telefon 013-28 77 42)
Antal uppgifter:	3
Max poäng:	40 poäng
Preliminära gränser:	Betyg 5 = 35p, 4 = 27p, 3 = 20p.
Hjälpmedel:	Inga hjälpmedel tillåtna!

REGLER

- Tentan genomförs individuellt.
- Tentan ska göras i en lugn miljö utan andra personer i samma rum.
- All kommunikation är förbjuden, undantaget frågor till kurspersonal.
- Alla källor du tar inspiration av ska redovisas.
- Dina lösningar skickas in via Lisam enligt nedan och kommer att köras genom Urkund.
- Var beredd på att i efterhand kunna redogöra för dina svar.
- Markera tydligt med rubriker eller dylikt vilken uppgift och deluppgift du svarar på (vi vill snabbt kunna hitta svaret på exempelvis uppgift 2 d).
- Observera att betygsgränserna kan komma att justeras, i samtliga kurser.
- Vid frågor om tidskomplexitet, svara alltid på den form som är mest relevant.
- Om inte annat framgår ska indexering av arrayer/listor börja från 0.
- Om ett problem medger flera olika lösningar, t.ex. algoritmer med olika tidskomplexitet, ger endast optimala lösningar maximalt antal poäng.

INLÄMNING

Lämna in dina svar som ett dokument med tydliga rubriker i det Lisam-rum som du har fått information om via e-post tidigare. Använd en ordbehandlare som låter dig formatera text med rubriker och dylikt. Sikta på att lämna in **2-4 sidor text**. Skicka sedan in antingen som *pdf*, *doc*, *docx*, *odt*, *org* eller *txt*.

FRÅGOR

Har du frågor om någon uppgift, eller om någonting går snett med din utrustning, hör av dig till kursledare antingen via Zoom (623 8655 4242) eller telefon 013-28 77 42.

Lycka till!

1. **Regler** (0 p)

Läs tentans regler ovan. Skriv sedan följande text i ditt dokument för att bekräfta att du har läst och förstått dem. Accepterar du inte reglerna så rättar vi inte tentan.

Jag har läst och förstått tentans regler, och jag lovar att jag har följt dem under tentans gång.

2. **En gastronomisk hantering** (20 p)

Din kompis driver en multikulturell restaurang i staden (du kallar den så eftersom du inte förstår namnen på rätterna eller valutan). Det har på senaste tid gått väldigt bra för din kompis, så denne har nyligen bett dig om hjälp med att hantera menyn, eftersom det har blivit för jobbigt att hantera den manuellt på sistone (då det är rätter som hanteras får det absolut inte bli fel).

Nedan följer ett utdrag ur menyn:

Namn på rätten	**Pris**	**Antal beställningar**
Nonomsidomsi	17	10
Notsonomsidomsi	3	8
Kalltikuli	15	2
Koodimahti	90	999
Varmtivattni	19	80
Kalltivattni	18	1
...	...	...

Systemet ska hålla reda på vilka rätter som finns, vad varje rätt kostar, och hur många gånger varje rätt är beställd. Systemet måste kunna göra följande saker:

- Lägga till en ny rätt, givet namnet på rätten
- Ta bort en gammal rätt, givet namnet på rätten
- Öka antalet gånger en rätt har blivit beställd
- Ta fram de k populäraste (dvs. mest beställda) rätterna för att se vad som behöver köpas in

Som DALG-student får du direkt ett antal idéer på hur du kan implementera detta:

1: En länkad lista som är sorterad efter antal beställningar

2: En osorterad array

3: Ett balanserat sökträd som är sorterat efter antal beställningar

(a) För var och en av de tre datastrukturerna ovan, beskriv hur du sätter in en ny rätt, givet rättens namn och pris. Ge även tidskomplexiteten för operationen. (3)

(b) För var och en av de tre datastrukturerna ovan, beskriv hur du tar bort en rätt, givet rättens namn. Ge även tidskomplexiteten för operationen. (3)

(c) För var och en av de tre datastrukturerna ovan, beskriv hur du ökar antalet gånger en rätt har blivit beställd, givet namnet på rätten. Ge även tidskomplexiteten för operationen i vart och ett av fallen. (6)

(d) För var och en av de tre datastrukturerna ovan, beskriv hur du hittar de k rätterna som har beställts flest gånger. Ge även tidskomplexiteten för operationen i vart och ett av fallen. (6)

(e) Vilken datastruktur skulle du välja baserat på din analys ovan? Motivera ditt svar. (2)

3. **Skrattletarna** (20 p)

Världens roligaste pirat (Guybrush Threepwood, känd bland annat för att kunna hålla andan i tio minuter[1]) har nyligen gått ur tiden, och efterlämnat sig sin skattkarta till den enda arvingen, du. I och med att du vet att skatten består av något mycket bättre än guld, nämligen årtionden av samlade ordvitsar, ger du dig naturligtvis direkt ut för att leta efter den!

På kartan finns ett antal grottor utmärkta i form av röda punkter. Du vet att skatten är utspridd i alla dessa grottor, så du måste se till att leta i alla för att inte missa någon del av skatten.

(a) Först måste du få tag på ett skepp. Du planerar också att använda kanoner för att underlätta transporten av personer. Kanonerna säljs separat. För att maximera dina odds att lyckas med skattletandet vill du därför hitta ett skepp och en uppsättning kanoner som tillsammans är så nära din budget som möjligt. (7)

Efter att ha frågat det lokala skeppsvarvet efter vad de har i lager så har du fått två listor. En lista med de n tillgängliga skeppen (inklusive pris), och en lista med de m tillgängliga kanonerna (också inklusive pris).

För att underlätta ditt arbete så vill du skriva ett program som hittar den bästa kombinationen åt dig, givet att du har en budget på k kr. Programmet ska alltså hitta det skepp och de kanoner som totalt kostar så nära under k som möjligt.

Beskriv hur programmet kan implementeras, samt ge tidskomplexiteten.

Uppgiften fortsätter på nästa sida!

[1] `https://www.quotes.net/mquote/1115487`

(b) Nästa steg är att få tag på en besättning. I och med att du inte har tillgång till mycket likvida medel, så har du insett att du kommer behöva betala besättningen i humor. Du har därför kallat de 100 roligaste wannabe-piraterna i staden till ”anställningsintervju”. Planen är att varje person får ställa sig framför en mikrofon medan du drar de 100 sämsta ordvitsarna du kan för närvarande. Mikrofonerna detekterar sedan när varje person skrattar, och du vill välja de 20 personer som har skrattat åt flest av ordvitsarna. (5)

Den mjukvara som kom med mikrofonerna lagrar helt enkelt varje gång ett skratt detekteras av en mikrofon och lagrar den i en array. I slutändan innehåller alltså arrayen ett element per skratt, och varje element innehåller id:t på personen som skrattade.

Exempelvis: [1, 1, 3, 2, 3] innebär att person 1 skrattade, sedan skrattade person 1 igen, sedan person 3, sedan person 2 och till sist person 3.

Du vill nu med hjälp av din trogna dator skriva ett program som hjälper dig plocka de 20 roligaste wannabe-piraterna i så kort tid som möjligt!

Beskriv hur du skulle implementera detta, samt analysera tidskomplexiteten på din lösning.

(c) När du väl kommer fram till öarna som är utmärkta på kartan inser du att *rolig≠atletisk*. Besättningen föredrar alltså att dra dåliga ordvitsar framför att klättra i berg. Du måste därför hitta ett sätt att undvika klättringen. Eftersom du är rolig låter du dig inspireras av clowner, och beslutar dig för att skjuta personer upp på bergen med dina trogna kanoner (då kan du dessutom meddela besättningen att ”det blir kanon!”). (8)

På kartan finns det en uppsättning grottor (märkta med röda punkter på bilden ovan). Planen är att skjuta upp ett team av skrattletare till någon av dessa. Teamet går sedan mellan grottorna (bara nedåt, de orkar inte med att klättra uppåt) tills de till slut kommer till hamnen. Vid behov kan teamet dela sig i två ifall det inte går att hitta en väg som besöker alla grottor. Du kan anta att teamet består av tillräckligt många personer så att de kan dela upp sig så mycket som det råkar behövas. Du kan också anta att det finns en punkt som är en lämplig startpunkt.

Det enda som återstår är att planera i vilken ordning varje team kan besöka grottorna utan att behöva klättra uppåt. Du tar återigen hjälp av din trogna dator, och börjar skriva ett program som kan lösa problemet. Indata till programmet är en lista av alla grottor (exempelvis: a, b, c, ...), sedan en lista över alla vägar mellan grottorna som finns (exempelvis: $a \to b$, $a \to c$, $b \to c$, ...). I och med att besättningen inte vill klättra uppåt, så kan de bara ta sig fram i den givna riktningen

Beskriv hur du skulle implementera detta (vilken algoritm väljer du, samt hur implementeras den på en hög nivå), samt analysera tidskomplexiteten på din lösning.