Linköpings Tekniska Högskola
Institutionen för Datavetenskap
Filip Strömbäck och Magnus Nielsen

25 oktober 2019

TDDI16
Datastrukturer och algoritmer
Datortentamen (DAT1)
2019-10-25, 14–18

Examinator:	Erik Nilsson
Jour:	Filip Strömbäck (telefon 013-28 27 57)
Antal uppgifter:	8
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.
- Du får själv välja om du vill skriva din lösning på papper eller på dator.
- Lösningar till olika problem skall placeras enkelsidigt på separata blad, eller i egen fil. Skriv inte två lösningar på samma papper eller i samma fil. Delproblem får dela papper / fil.
- Om inte annat framgår ska indexering av arrayer / listor börja från 0.
- Papper: Sortera lösningarna innan de lämnas in.
- Filer: Skicka in som lösning till rätt problem i tentaklienten, och döp filen till ett passande namn (exempelvis uppg1.txt).
- **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.
- Papper: Lämna plats för kommentarer.
- SE TILL ATT DINA LÖSNINGAR/SVAR ÄR LÄSLIGA.

Lycka till!

1. **Algoritmer och Tidskomplexitet** (6 p)

Givet följande implementation av en dynamisk array:

```
class Int_Vector {
private:
  // Array som innehåller datan. Kan vara större än vad som behövs för
  // att slippa allokera om den vid varje insättning.
  int *data{new int[2]};

  // Längden av data-arrayen.
  int length{2};

  // Anger hur många element inuti 'data' som innehåller data.
  int used{0};

public:
  // Ger storleken på listan.
  public int size() {
    return used;
  }

  // Fler medlemmar här...
};
```

(a) Beräkna tidskomplexiten med avseende på n (`used`) för följande medlem: (1)

```
int Int_Vector::sum() {
  int result = 0;
  for (int i = 0; i < used; i++) {
    result = result + data[used];
  }
  return result;
}
```

(b) Beräkna tidskomplexiten med avseende på n (`used`) för följande medlem: (1)

```
void Int_Vector::remove(int index) {
  for (int i = index; i < used - 1; i++) {
    data[i] = data[i + 1];
  }
  data[used] = 0;
  used--;
}
```

(c) Beräkna den **amorterade** tidskomplexiten med avseende på n (`used`) för följande medlem: (2)

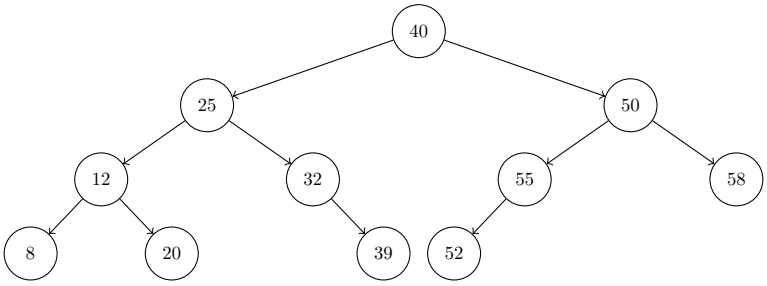
```
void Int_Vector::insertLast(int value) {
  if (data.length == used) {
    int* newData = new int[length*2];
    for (int i = 0; i < used; i++) {
      newData[i] = data[i];
    }
    swap(data, newData);
    delete []newData;
    length = length * 2;
  }
  data[used++] = value;
}
```

(d) Beräkna tidskomplexiten med avseende på n (`used`) för följande medlem: (2)

```
void Int_Vector::clear() {
  while (used > 0) {
    remove(used - 1);
  }
}
```

2. **Träd** (4 p)

Studera följande träd:



(a) Är trädet ett binärt sökträd? Motivera! (1)

(b) Är trädet komplett? Motivera! (1)

(c) I vilken ordning besöks noderna om en preorder-traversering av trädet görs? (2)

3. **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	18	Null	23	3	13	6	Null	28	8
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 0 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)

4. **Sorteringsalgoritmer** (2 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:	23	65	14	37	44	80	66	82	68	34	63	61	24	30	45
1:	14	23	24	37	44	80	66	82	68	34	63	61	65	30	45
2:	14	23	37	65	44	80	66	82	68	34	63	61	24	30	45
3:	23	30	14	24	34	37	44	45	68	66	63	61	80	65	82
4:	66	65	61	37	63	24	45	30	23	34	44	14	68	80	82
Sorterad:	14	23	24	30	34	37	44	45	61	63	65	66	68	80	82

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

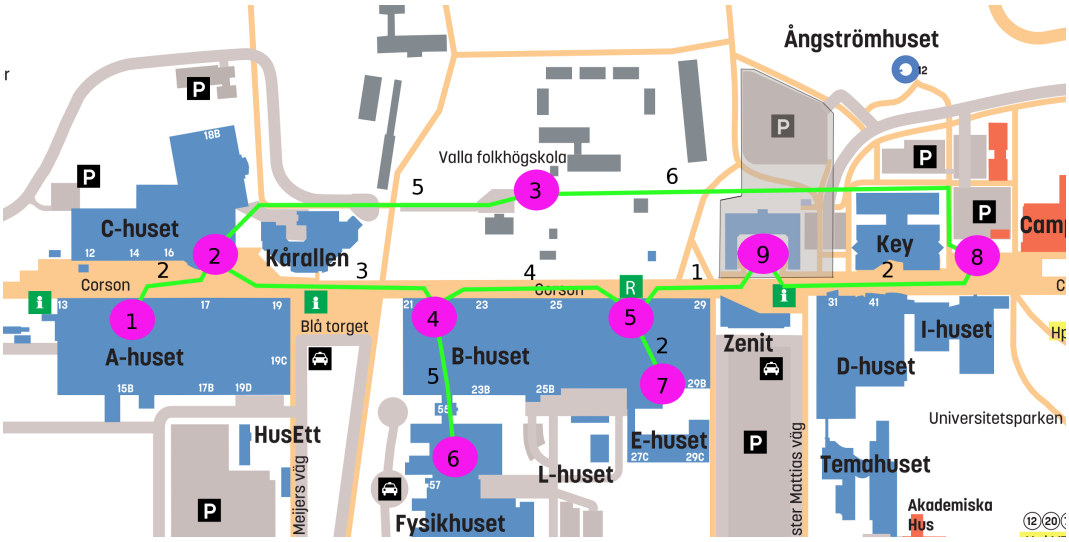
(b) Insertionsort

(c) Heapsort

(d) Selectionsort

5. **Falafelleverans** (6 p)

I bilden nedan finns en karta över Campus Valla. Falafelhuset (nod 8) har tagit det utmärkta beslutet att börja leverera mat till hela Campus Valla. För att effektivisera leveranserna har de bett dig, med dina kunskaper inom DALG, att hjälpa till med planeringen av leveransrutter. Du har till uppgift att skriva ett program som givet en destination ger personalen den kortaste vägen dit. Eftersom de har många leveranser är den viktigaste egenskapen hos programmet att den snabbt kan hitta den kortaste vägen från Falafelhuset till vilken annan plats på campus som helst!

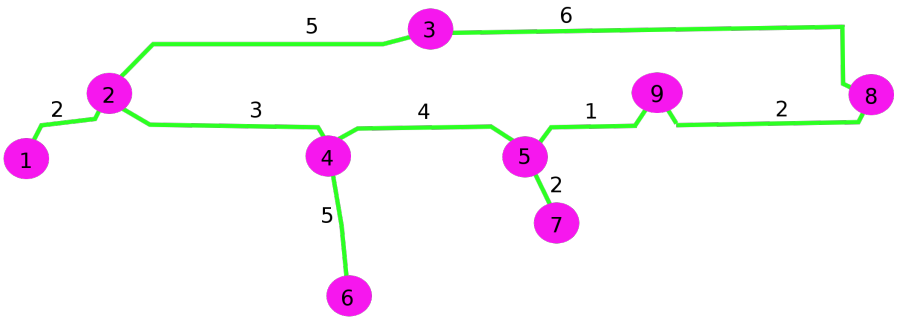


(a) Beskriv hur du implementerar din lösning. Vilka datastrukturer och algoritmer väljer du? (3)

(b) Vad är tidskomplexiteten hos ditt program för att hitta den kortaste vägen till en godtycklig plats på campus? (1)

(c) Efter att ha levererat mat ett tag inser ledningen på Falafelhuset (Falafelbossen) att det är onödigt att leverera en order i taget hela tiden. Det skulle vara effektivare om en person kan ta två ordrar och leverera den ena exempelvis till Valla folkhögskola (nod 3) och den andra till Fysikhuset (nod 6) utan att behöva ta vägen förbi Falafelhuset. Hur kan du modifiera ditt program för att beräkna en väg till två (eller flera) destinationer? (2)

Högkontrastversion av kartan från ovan:



6. **Reseplaneraren** (6 p)

Du är i färd med att planera en utlandsresa till en liten stad på landsbygden i ett land långt borta. I och med att landet ligger långt borta har du beslutat dig att flyga till en av de stora internationella flygplatserna i landet. Det går dock inte att flyga hela vägen till ditt mål, så någonstans på vägen måste du byta transportmedel. Det finns ett flertal internationella flygplatser i landet, och utöver det finns en stor mängd transportmedel från flygplatserna till olika ställen i landet. Exempelvis finns det inrikesflyg, tåg och buss att välja mellan. Dock återstår den svåra, men mycket viktiga frågan: vilken väg är billigast? Du vill inte råka betala för mycket för din resa, då har du ju inga pengar kvar att handla souvenirer för!

Efter en hel dags arbete har du lyckats sammanställa en stor lista över de rutter som finns. Dels vilka flyg som går hemifrån till de olika flygplatserna i landet, och dels en stor lista över vilka transportmedel som finns tillgängliga när du väl kommer fram. Listan är strukturerad som nedan:

Från	**Till**	**Typ**	**Pris**
Arlanda	North int. airport	Flyg	3000
Arlanda	South int. airport	Flyg	2500
Arlanda	West int. airport	Flyg	3500
West int. airport	Westmore	Buss	100
West int. airport	Westmore	Tåg	150
West int. airport	Eastmore	Tåg	200
South int. airport	West int. airport	Flyg	500
Westmore	Målby	Buss	50
...	...	...	

Du har sett att det finns tillräckligt med alternativ i listan för att du garanterat ska kunna ta dig hemifrån till ditt mål. Dessutom har du sett att alla linjer går regelbundet och dygnet runt. Du behöver alltså inte oroa dig för att exempelvis inte hinna i tid till sista bussen, det finns alltid en buss till. Du bryr dig inte heller om hur lång tid resan tar, eller vilken typ av fordon du åker, det enda viktiga är priset!

I och med att din lista har blivit ganska stor, och du har blivit trött efter att ha letat reda på all information beslutar du dig för att skriva ett program som kan räkna fram den billigaste rutten åt dig. Skulle du göra det själv skulle det bara bli fel!

(a) Beskriv hur du kan implementera ett program som hittar den billigaste vägen till ditt mål. Du vill veta både hur mycket resan kostar totalt och vilken väg du ska åka. Beskriv och motivera dina val av datastrukturer och algoritmer. (3)

(b) Vilken tidskomplexitet har din lösning, uttryckt i antalet rutter som finns i listan (N) och antalet platser du kan besöka (P)? (1)

(c) Efter att du har kommit tillbaka blir en av dina vänner också sugen på att resa till samma land, men till en annan stad. Din vän blir dock lätt åksjuk av att åka buss, och vill därmed undvika bussresor så långt som möjligt. Beskriv hur du kan modifiera din lösning så att den hittar den billigaste vägen som innehåller så få bussresor som möjligt. Finns det en väg som undviker buss helt vill vi alltså välja den. Går det inte att komma fram utan bussresor vill vi åka buss så få gånger som möjligt. (2)

7. **Maximal gastronomisk upplevelse** (6 p)

Du är på besök i ett land du aldrig tidigare besökt, och du kan inte ett ord av språket. Dock vill du självfallet ta del av den lokala lyxmaten, trots att du som student har en tight budget. En mystisk, engelsktalande, herre i mörk kappa har gett dig en lista på de lokala restaurangerna samt tillhörande prislistor, men du förstår fortfarande inte vad rätterna är eller vad de innehåller. Det enda logiska är så klart att hitta den dyraste rätten du kan, inom din budget.

Listan innehåller n rader, varje rad innehåller namn på restaurang, namn på rätt samt pris. För närvarande är listan grupperad på först restaurang, sedan i tur och ordning förrätt, varmrätt och efterrätt (du vet dock inte vilket som är förrätt, varmrätt eller efterrätt). Eftersom det finns väldigt många restauranger, och många rätter på varje restaurang, behöver du bygga ett hjälpmedel för att välja ut den dyraste rätten du har råd med, givet din budget. Du har inget emot om det råkar vara en dessert, så länge den är dyr och inom din budget!

När du står och funderar på vilken rätt du ska välja noterar du ett antal turister som ser lika förvirrade ut som du. Vänlig som du är går du fram och frågar hur det står till. Det visar sig att de är i precis samma situation som du är. De vill också ta del av den lokala lyxmaten på en budget och de kan inte heller språket. Då du själv fick hjälp av den mystiska herren anser du att det är inte mer än rätt att du själv står till tjänst (plus att det är galen street-cred att visa upp sina DALG-kunskaper), och utökar ditt hjälpmedel för att även kunna hjälpa dina nyfunna turistvänner. Antag att du hittade k turistvänner, och att $k \approx n$. Alla turister har olika budget för dagen.

Beskriv och motivera dina val av datastrukturer och algoritmer. Notera även vilken tidskomplexitet (gärna även minneskomplexitet) din lösning har.

Exempel på lista:

Restaurang	**Namn på rätten**	**Pris**
Dorsia	Seekoeier ceviche	17
Kolm luud	Pivo od maslaca	3
Los Pollos Hermanas	Pollo Caliente	15
Kafejo Monk	Velika Salata	19
Los Pollos Hermanas	Pollo Frio	10
...	...	...

(a) Beskriv din lösning. Motivera dina (ditt) val! (2)

(b) Vilken tidskomplexitet har din lösning för att beräkna vad **alla** ska äta? (1)

(c) Du kom dock snabbt på ett problem med din lösning. Du är bara i landet en vecka, men du vill ju prova så många rätter som möjligt! Det är därför inte effektivt att du bara äter en rätt om dagen, du måste beställa två! Beskriv hur du kan modifiera din lösning så att den i stället hittar två rätter så att summan av deras priser blir så dyr som möjligt, men ändå inom din budget! (2)

(d) Vilken tidskomplexitet har din modifierade lösning för att komma fram till vad **du** ska äta? (1)

8. **Datorbutiken i Troja** (6 p)

En datorbutik i stan, Trojan Computing, har nyligen fått erfara större IT-problem. Deras POS-system (Point of Sales) har nyligen kraschat ordentligt, och de får inte igång det igen. 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 iordningställa den datan som finns så att butiken kan deklarera sin inkomst innan skatteverket blir upprörda.

Exempel:

ID	**Datum**	**Belopp**	**Säljare**	**Beskrivning**
8	2019-10-01	5000	Kim	Processor och moderkort
10	2019-10-28	300	Magnus	Tentakaffe
5	2019-10-25	250	Filip	Tentagodis
215	2019-05-10	4500	Frida	Nästan virusfri dator
17	2019-08-05	400	Alojz	2 GHz (lösplock)
78	2019-03-18	0	Kim	Installation av bakdörr (inte baklucka)
15	2019-06-13	500	Frida	Trojansk häst (uppstoppad)

(a) Beskriv hur du effektivt kan beräkna den totala summan av alla transaktioner för varje dag under det senaste året. (3)

(b) Vilken tidskomplexitet har din lösning, uttryckt i det totala antalet transaktioner, n? (1)

(c) Efter att ha sett resultatet blir ledningen mycket glad. Dock kom de på att de håller en tävling bland säljarna. Den som har sålt för högst belopp varje dag av säljarna får en poäng. Den som sedan har fått flest poäng under det senaste året vinner. Dessvärre saknas, som du kan se från exemplet ovan, den informationen helt från datan. Hur kan du effektivt räkna fram den igen, så att säljarna kan se en topplista över sina poäng? (2)