Tildatenta 2018-03-17 lösningsförslag

E-delen

1 KMP

```
S K I D  S K Y  T T E  S K I  D O R
0 1 1 1  0 1 3  1 1 1  0 1 1  1 5 1
```

Ett fel är OK och ger fortfarande E. Man behöver inte ange nextvektorn utan det räcker med att rita.
Youtube-variant

```
S K I D  S K  Y T T E  S K I D  O R
0 0 0 0  1 2  0 0 0 0  1 2 3 4  0 0
```

2 Ordo

```
v = []      # vektor, pythonlista
for x in range(1, 10000) : v.append(x)
v.insert(0,17)    # fråga a)
v.append(17)      # fråga b)
```

a) O(N) för insert flyttar elementen
b) O(1) för lägga till sist för det finns extra plats i vektorn

3 Graf på matrisform.

	röd	orange	gul	korridor
röd		**x**		**x**
orange			**x**	**x**
gul				**x**
korridor				

Förbindelserna är precis som i verkligheten. Det går inte att gå mellan från gul till röd sal. Det räcker med triangel men att fylla i hela kvadraten är OK. Det borde stå 0 i GUL-RÖD ruta men tom ruta är godkänt.

Det är fel att fylla i alla rutor med kostnaden att gå från ett rum till ett annat, då tappar man information om hur förflyttningen ska gå till.

4 Syntax

Rekursion måste vara med. Avbrottsfall ska vara med.

<SKJUTSERIE> ::= <SKOTT> | <SKOTT> <SKJUTSERIE>
<SKOTT> ::= PANG | BOM

reguljärt uttryck är FEL
<SKJUTSERIE> ::= **<SKOTT> ***
<SKJUTSERIE> ::= **<SKOTT> +**

En tom SKJUTSERIE rekursion är FEL
<SKJUTSERIE> ::= <SKOTT> | **<SKJUTSERIE>**
<SKOTT> ::= PANG | BOM

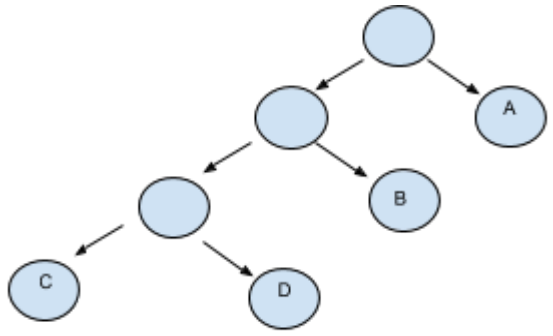
Små fel godkänns. t.ex i nedanstående är endast flera PANG tillåtna. BOM <SKOTT> saknas.

<SKJUTSERIE> ::= <SKOTT>
<SKOTT> ::= PANG | BOM | PANG <SKOTT>

Att missa ett fall som PANG PANG godkänns, se nedanstående.

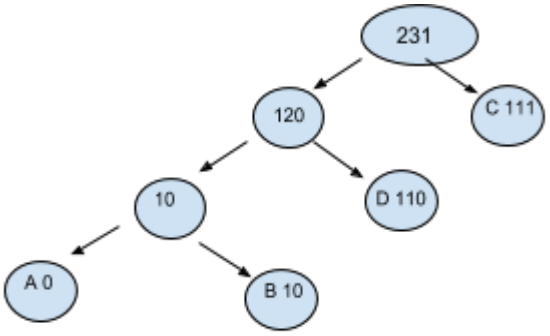
<SKJUTSERIE> ::= <SKOTT> | **<SKOTT> <SKOTT>** <SKJUTSERIE>
<SKOTT> ::= PANG | BOM

5 Huffman



A = 0, B = 10. C = 111, D = 110
Procent ges av A > B > C + D t.ex. 40 > 30 > 15 + 15

Om läsförståelsen har brustit så är det även godkänt att inte kunna läsa uppgiftslydelsen och göra ett huffmanträd där man tolkat siffrorna som teckenfrekvens och därefter följt hufmanalgoritmen.



Teckenkods**längderna** kommer att vara samma som nedan
A 1111
B 1110
D 10
C 0
och procenten t.ex för C = 111/231

Det har även förekommit att man tolkat det som att teckenförekomsten gets i binärform d.v.s.
A = 0 = 0
B = 10 = 2
C = 111 = 7
D = 110 = 6 decimalt.

E 6

b) **siffran 5 hamnar i mitten.**

a) Har några möjliga lösningar.

1 3 2 **5** 7 8 9 # pivotelement till vänster (denna gäller även dance-partition)
3 1 2 **5** 7 8 9 # skiftat pivot element till höger dvs starta med 5 9 2 3 7 8 1
1 2 3 **5** 7 8 9 # Algorithm unlocked-boken där man har två pekare där den ena går fortare än den andra i det här fallet blir det som lite bubblesort i början och sedan går den snabba pekaren till slutet.

Det är **fel** att göra en insert av siffran 5 på slutet, **se fråga 2) !!!!!** och få 1 3 2 5 **9 7 8**

Att börja med att skifta in 5 på rätt plats är FEL

1 9 2 3 7 8 5
1 9 2 **5** 7 8 3 // 5 antas hamna i mitten - FEL

Det finns en annan partitioneringsalgoritm som utgår från att det mittersta värdet är pivotelement och flyttar runt pivotelementet allt eftersom. Det värde som finns där “fingrarna korsar” blir det nya rättsorterade elementet och det ursprungliga pivotelementet blir inte rättsorterat.

C-delen

tal 7 Alla tre funktionerna har problem.

f1 - random key gör att man stoppar in på random ställe och plockar ut något annat helt random. Det gör att hashtabellen inte fungerar alls!

f2 - har ingen bra distribution. Hanna ÖBERG elva bokstäver * bokstaven a
len(“HANNA ÖBERG”) * ord(a) ~= 100 * 11 = 1100 * 11 bokstäver

f3 - tar inte hänsyn till datum så alla skjutserier för en skidskytte hamnar på samma plats.

a] Lämpliga krav i detta fall:

A) Man ska få samma index varje gång givet en viss nyckel! Om man stoppar in nyckel/värde ska man få tillbaka samma värdet givet samma nyckel
B) Hashfunktionen ska undvika klustring, att värdena klustras ihop.
C) Hela vektorn ska användas, hashfunktionen ska kunna ge stora värden

b] Sett ur detta perspektiv, obs fylligare (se problem ovan) beskrivning krävs

f1 A - inte alls!!!
B - utmärkt!
C - utmärkt!

f2 A - JA!
B - JA, men det finns förbättringspotential, datumen bidrar enahanda
C - NJAE, namnen når inte hela vektorns längd.

f3 A - Jo
B - Nej, datumet tas det ingen hänsyn till
C - Ja, längre namn når hela vektorn

c] *Sammanfattningsvis. Ingen är tillräckligt bra men en modifierad f3 med s+= datum fungerar bäst.*

Tal 8

A ska bort man kan inte äta 3:an före 5:an
D ska bort man kan inte äta 2:an före 3:an i D

För kö finns det inga alternativa våffelätningsserier, det blir köordning 1,2,3,4,5 …

Tal 9

Talet består av fem uppgifter. Två algoritmkonstruktioner samt deras respektive komplexitet och att verifiera med ett litet exempel. Man behöver klara alla fem uppgifter för betyg A. Algoritmbeskrivningen ska vara förståelig. För betyg B tillåts högst ett grovt fel i algoritmen/komplexiteten.

a) En korrekt algoritm har linjär komplexitet O(N). Det är OK att linjärsöka. Om man kan anta slumpmässig fördelning (vilket är OK att göra) så kan man konstruera en effektivare algoritm som kollar i respektive barn om det är lönt att söka vidare i det barnet. Det är fel att påstå att en sådan algoritm blir O(log N), den blir i bästa fall O(log N) men i allmänhet O(N) eftersom algoritmen flera gånger kommer söka i bägge barnen. En icke godkänd oeffektiv metod är att plocka ut alla element och undersöka och sedan stoppa in dem igen.

b) En korrekt algoritm har komplexitet O(logN).
   i) Om det nya värdet är bättre än sina föräldrar ska en upptrappning ske. Den kan ske rekursivt eller iterativt och pågår så länge som värdet är bättre än sina föräldrar. Man ska ha med matematiken i svaret t.ex. while (v[i] > v[i // 2)
   ii) Om det nya värdet är sämre än sina barn så ska en nedtrappning ske. Det är viktigt att påpeka att det är bästa (och inte nästbästa) barnet som det ska bytas plats med. Även här ska man hålla på iterativt (eller rekursivt) så länge något barn är bättre. Man ska ange hur man når barnen med 2*n eller 2*n+1. Det är OK med liknande matematik t.ex.2 * N+1 och 2 * N+2 om man är konsekvent även i upptrappningen.

c) Exemplen bör innehålla en upptrappning och en nedtrappning men eftersom detta inte var specat noga så räcker det med ett exempel.