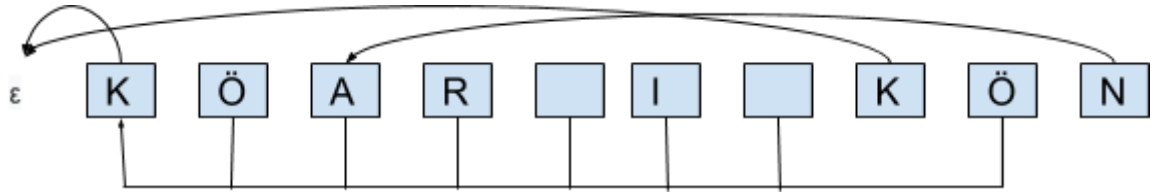
Lösningsförslag DD1320/DD1321 2019-03-13

1. KMP för IT-support

K Ö A R I K Ö N

0 1 1 1 1 0 1 3 eller 0 1 1 1 1 1 1 0 1 3



- Lösning med eller utan mellanslag godkänns
- 5 ettor i följd godkänd (slarvfel)
- sista siffran (3) ett krav men kan avvika om bild rätt (men F om bägge är fel)

2. Vilken datastruktur är det?

*På KTH:s it-support fördelas inkommande ärenden över ett antal supporttekniker som sedan hanterar dessa i den ordning de kom in och de lagras därför lämpligast i en **kö (1=b)**. I samband med att ett ärende skapas skickas också ett mail för att notifiera it-supporten om detta vilket utgör en slags todo-lista för dessa. Ada, som är nyanställd på supporten, har valt att sortera sin inbox i ordningen nyast överst och betar av ärendena uppifrån och ned vilket leder till att Ada betraktar ordningen som en **stack (2=e)**.*

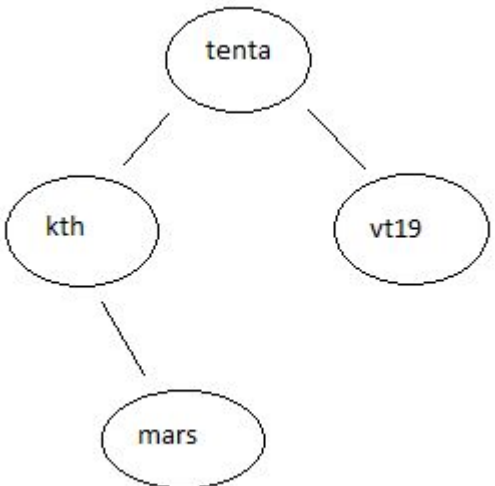
*En del ärenden behöver hanteras mer skyndsamt än andra och då är det lämpligt att lagra dessa i en **prioritetskö (3=f)**. Slutligen, när ett ärende är klart, arkiveras de i en struktur där epostadress (för den som behövde hjälp) ska kunna användas för effektiv sökning och därför används lämpligen en **hashtabell (4=c) eller binärt sökträd (4=a)** .*

En sorterad lista (vektor) kan användas i 4 om det motiveras med binärsökning. En lista (vektor) kan också användas istället för stack eller dubbeländad kö (deque) om det motiveras.

3. Binärt sökträd

a)

Utskrift preorder: *tenta kth mars vt19*



Utskrift inorder: *kth mars tenta vt19*

Utskrift postorder: *mars, kth, vt19, tenta*

b) O(n)

Några har skrivit O (log N). Det är fel. Om man ska skriva ut hela trädet måste alla N noder gås igenom

Lösningsförslag DD1320/DD1321 2019-03-13

4. Syntax för nätverk

Finns ett flertal t ex <NODE> :== <TYPE> | <NODE> <TYPE>

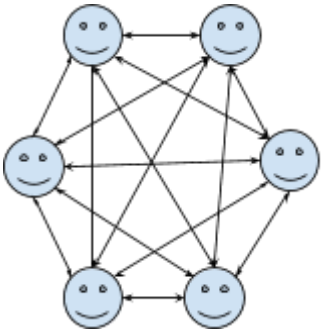
- Ett absolut krav är att rekursion finns med.
- Rekursionen måste ha ett avbrott.
- Det givna exemplet CISCO PC MAC MAC ska klaras av.

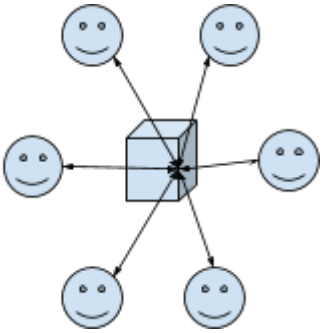
5. Graf

a) n*(n-1)

b) 2*n

c)





6.

Redundans till ASCII (tilda)

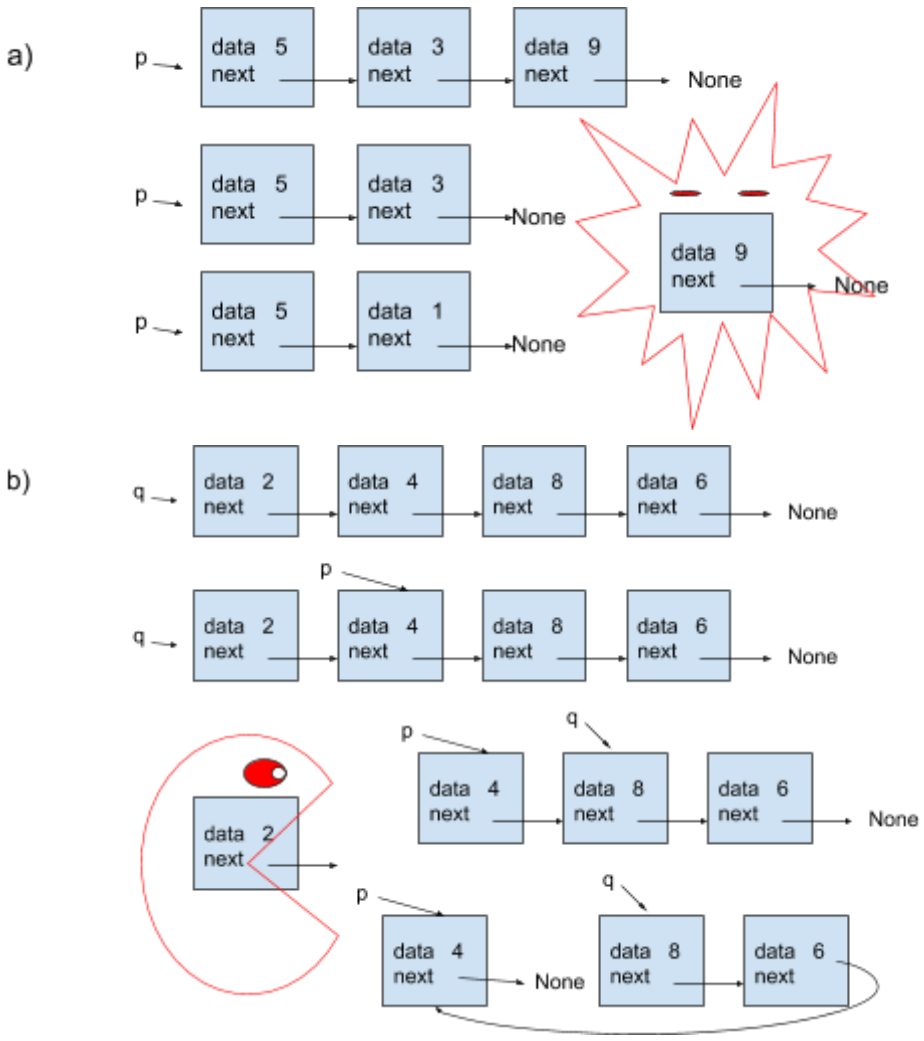
a) Minsta hammingavstånd är 1 (kan vara allt mellan 1 och 7 för de olika tecknen)

- Kan visas med exempel mellan två tecken t.ex. **A** och **a** i lydelsen
- Allt annat än 1 som svar är fel.

b) Om man adderar fler bitar så att minsta hammingavståndet är 3 så kan man gissa sig till vad som skickats om en enstaka bit ändrats på vägen. Man kan addera en paritetsbit och göra alla tal jämna eller udda vilket ökar chansen att detektera en felöverföring.

Lösningsförslag DD1320/DD1321 2019-03-13

6 Pekare (tilpro)



- En miss i något steg är OK och då måste efterföljande satser (**om de ritats**) rättas därefter

C-delen

7. Kryptering med bokchiffer

3 jämförelsekriterier efterfrågas, olika utfall av jämförelserna kan godkännas beroende motivering.

a) Tre jämförelser mellan bokchiffer med digital respektive tryckt bok.

i) **Kryptering** - bokchiffer är starkt krypterat. **Lika starkt** digitalt och i tryckt format

ii) **Nyckelutdelning** - Kan tänkas vara både lika svårt att lösa digitalt och tryckt bok. Det går att argumentera för att det är **både** enklare **eller** svårare att distribuera en exakt likadan digital bok/pdf än en tryckt. T.ex. kan vattenmärkning av en pdf tillföra tecken eller epub/mobi-format vara olika på olika plattformar.

iii) **Prestanda** - Det går **snabbare** att dekryptera/krypterar en digital bok med ett datorprogram.

b) Bokchiffer vs RSA:

i) **Kryptering** - i princip **lika stark** som RSA. Dåligt indata kan göra det knäckbart.

ii) **Nyckeludelning/autentisering** - RSA löser detta **bättre** än bokchiffer. Den publika nyckeln i RSA behöver inte vara hemlig och kan enklare delas ut.

iii) **Prestanda** - Digitalt bokchiffer är potentiellt snabbare än RSA men det är OK det går att argumentera för båda. Man behöver inte motivera skillnad i digital snabbhet. Bokchiffer med tryckt bok är **väsentligen långsammare**.

Det finns andra jämförelsekriterier och jämförelseresonemang som godkänts, t.ex.

iv) Implementation - enklare att implementera bokchiffer, i synnerhet digitalt

v) Autentisering - kan lösas med en tillfredsställande nyckelutdelning. Bara den som har nyckeln/boken kan skicka meddelanden och man kan föra liknande jämförelseargumentation som för nyckelutdelningen

vi) Säker förvaring - större risk för intrång i datorn med digital bok. Det kan finnas spår, digitalt eller fysiskt.

F - Har endast svarat på en delfråga
D - 3 korrekt motiverade och specificerade jämförelser
C - 5 eller 6 korrekt motiverade och specificerade jämförelser

8. Datastrukturer för dictionary

För betyg D så ska man ta hänsyn till att det är **ett okänt antal nycklar** och nämna minst en av

- Binärträd kan växa obehindrat
- Om hashtabellen inte omdimensioneras -> löjligt långa krocklistor
    - eller skriva att dimensionera hashtabellen kräver omhashning av alla element.

Dessutom ska man för betyg D jämföra prestanda i det allmänna fallet

- Binärträd O(logN) vs Hashtabell O(1)

För C så ska man uppfylla D och nämna alla tre ovan eller 2 + en annan bra jämförelse t.ex. risk för träd att bli obalanserad med sorterat indata.

Lösningsförslag DD1320/DD1321 2019-03-13

A-delen

9. Brädspel

Komplexitet:
Komplexiteten efterfrågas inte i uppgiften men det kan vara av värde för framtida lösningsförslagsläsare att resonera kring matematiken.

Ett bräde med en lampa kan lysa på tre sätt. Ett bräde med två lampor kan lysa på nio sätt. Ett bräde med tre lampor kan lysa på 27 sätt. Ett bräde med N lampor lyser på 3^N sätt

RR BR GR

GB RB BB

BG GG RG

två lampor

RRR BRR GRR GBR RBR BBR BGR GGR RGR

RGB BGB GGB GRB RRB BRB BBB GBB RBB

RBG BBG GBG GGG RGG BGG BRG GRG RRG

bräde med tre lampor kan lysa på 3^3 = 27 sätt

RR BR GR GB RB BB BG GG RG RG BG GG GR RR BR BB GB RB RB BB GB GG RG BG BR GR RR
RR RR RR RR RR RR RR RR RR BR BR BR BR BR BR BR BR BR GR GR GR GR GR GR GR GR GR

RR BR GR GB RB BB BG GG RG RG BG GG GR RR BR BB GB RB RB BB GB GG RG BG BR GR RR
GB GB GB GB GB GB GB GB GB RB RB RB RB RB RB RB RB RB BB BB BB BB BB BB BB BB BB

RR BR GR GB RB BB BG GG RG RG BG GG GR RR BR BB GB RB RB BB GB GG RG BG BR GR RR
BG BG BG BG BG BG BG BG BG GG GG GG GG GG GG GG GG RG RG RG RG RG RG RG RG RG

bräde med fyra lampor kan lysa på 3^4 = 81 sätt

Jämför med ordkedjelabben där antal kombinationer för trestaviga ord är 29^3 = 24389 (med *w*)

Komplexiteten för att söka igenom ett sökträd är i värsta fallet att gå igenom hela trädet.
Alla kombinationer kommer inte att undersökas, i ordkedjelabben filtrerar man mot en ordlista.
Beroende på utseendet av brädspelet så kan inte alla lampkombinationer uppnås. Utan att känna till utseendedetaljerna kan man bara uttala sig om en teoretiskt övre gräns.

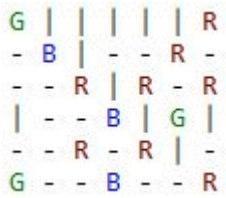
Breddenförstsökning

Det som **efterfrågas** är den **kortaste knapptryckningssekvensen** vilket man får genom att söka i trädet med **bredden först**. Man måste i varje steg **spara** undan den knapptryckning man gör, antingen i noder med föräldrapekare (som i labben) eller genom att bygga på en knapptryckningslista som skickas vidare i varje steg.

Det kan bli en väldigt lång kö om det finns många kombinationer att undersöka.

Lösningsförslag DD1320/DD1321 2019-03-13

Bästaförstlösning kan ge fel svar



Spelbrädet till vänster har ca en halv miljon (531441) kombinationer. Den går att lösa med 6 knapptryckningar. (1, 0), (1, 0), (0, 3), (0, 3), (0, 5), (3, 0) och lösningen hittas efter att ha undersökt 22932 bräden. Om man istället söker i prioritetsordning, t.ex. genom att prioritera bräden där en färg dominerar så kan man hitta en lösning redan efter 39 bräden men det är **inte den kortaste** sekvensen utan har dubbelt (12) så många knapptryckningar.

Undersök inte besökta matriser på nytt

I uppgiften framgår att när man trycker på samma knapp fyra gånger så får man samma bräde igen. Vissa knappar ändrar lampor på samma sätt t.ex. flera **|** i samma kolumn. För att undvika onödiga undersökningar måste man hålla koll på redan undersökta matriser. Om man inte gör det kan det bli orimligt många undersökningar..

Nedan undersöks två olika implementationer av breddenförstsökning för matrisen ovan. Den ena tar inte hänsyn till redan besökta matriser (dumbarn), trycker på alla 28 knappar och lyckas inte hitta rätt lösning innan minnet tar slut. Den andra lösningen sparar bara 12 knappar (7 vertikala och 5 horisontella), undersöker inte redan besökta matriser och hittar rätt lösning. Notera skillnaden i köstorlek i 5:e barngenerationen **sjutton miljoner** visavi fjorton tusen.

breddnivå	antal sökningar	kölängd	antal sökningar	kölängd
	utan att ta hänsyn till dumbarn och med reduntanta knapptryckningar		med hashtabell för redan besökta	
0	1	1	1	1
1	2	28	2	12
2	30	784	14	144
3	814	21952	158	936
4	22766	614656	1094	4224
5	637422	**17210368**	5318	**14652**
6	slut på minne	slut på minne	19970	41184

Det är **viktigt** att man kan spara undan och slå upp om man redan undersökt matrisen t.ex. genom att räkna ut ett hashvärde för matrisen och spara i en hashtabell. Det är också onödigt att undersöka reduntanta knapptryckningar som genererar samma matriser som en annan knapp.

Datastrukturer:

en teckenmatris eller lista av listor (pythonvektorer) exempelvis den ovan
en lista/vektor med radnummer för specialknappen —
exempel radlista [(1, 0), (2, 0), (3, 1), (4, 0), (5, 1)]
en lista/vektor med kolumner för specialknappen |
exempel kolumnlista [(0, 1), (0, 2), (0, 3), (0, 4), (0, 5), (3, 0), (3, 6)]

en kö för breddenförstsökning
en hashtabell eller binärt sökträd för redan undersökta matriser.
en nod med matris och knappkoordinatlista

en hashtabell med lampbytarordning (kan också göras med en funktion)
change = {"G": "R", "B": "G", "R": "B"}
m[3, 4] = change [m[3, 4]]

Funktioner

- Ändra matrisen, det räcker att beskriva i text om hur en rad resp. kolumn ändras. Uttryckt i kod kan man tänka sig en rowchange och en colchange som i princip gör följande

```
for i in range(len(m)):
    if m[i][col] in "RBG":
        m[i][col] = change[m[i][col]]
```

```
for i in range(len(m[0])):
    if m[row][i] in "RBG":
        m[row][i] = change[m[row][i]]
```

- funktion eller funktionalitet som kan jämföra två matriser för insättning i binärträd alternativ funktionalitet för att stoppa in i hashtabell.
- funktion eller funktionalitet som avgör om alla lampor lyser i samma färg måste finnas med

Algoritm

Läs in från filen till en matris och till en rad- och kolumn-lista
Lägg till startmatrisen i redanbesökta-hashtabellen/trädet
Lägg en nod med startmatrisen och en tom knapptryckningslista i kön.
Så länge kön inte är tom:
    plocka ut en matrisnod ur kön.
    kolla om matrisen är lösningen i så fall avbryt och returnera knapptryckslistan
    annars:
        för varje radknapp i rad-listan och varje kolumnknapp i kolumn-listan:
            generera ny matris med rowchange/colchange
            Om nya matrisen inte finns i redanbesökta-hashtabellen/trädet:
                skapa en ny nod med
                - den nya matrisen
                - en knapptryckslista med den nya knappen tillagd
                lägg till matrisen i redanbesökta-hashtabellen/trädet
                lägg till matrisnoden i kön

För betyg B krävs en korrekt **bredden först** lösning. Tydlighet kring hur **barnen genereras** i text eller bild. Att **redan besökta** matriser inte undersöks igen. Något sätt att få reda på **den efterfrågade knapptryckningssekvensen** t.ex. via parentpekare eller en knapptryckningslista som byggs på.