Skriftlig reeksamen i ADA5

Algoritmer og datastrukturer

27. februar, 2024

Der er i alt 7 opgaver til denne eksamen.

Der afleveres **ét og kun ét** pdf-dokument. Kode, som du ønsker testet, **skal** være editérbar og derfor *ikke* i form af screenshots.

I programmeringsopgaverne kan du vælge frit mellem Java og C++.

Du må gerne skrive dine forklaringer på dansk.

Opgave 1 (12 %)

Skriv en metode/algoritme med følgende signatur:

```
bool sumAfToTalLigParameter(int[] arr, int l, int X)
```

Den første parameter er et array af positive, sorterede heltal uden dubletter; anden parameter er længden af arrayet; og tredje parameter er den værdi, der ledes efter.

Metoden skal finde af om der findes to tal i arrayet, hvis sum er lig med parameteren X. Hvis svaret er ja, returneres `true`. Hvis svaret er nej, returneres `false`. Det samme tal må ikke bruges to gange.

Opgaven skal løses i to versioner således at:

- Version 1 har kvadratisk tidskompleksitet – $O(N^2)$.
- Version 2 har lineær tidskompleksitet O(N), det vil sige, at hvert element i tabellen kun må læses én gang.

De to løsninger tæller lige meget, altså 6 %.

Opgave 2 (10 %)

Hvad er Store O (Big-Oh) tidskompleksiteten for nedenstående metode? Begrund dit svar.

```
public static long myM(int N)
{
    long x = 0;
    for (int i = 0; i < 10000; i++)
    {
        for (int j = 0; j < N*10; j++)
        {
            for (int k = N; k > 0; k = k/10)
            {
                x++;
            }
        }
    }
    return x;
}
```

En brugbar strategi for at løse opgaven, udover at analysere koden, kan være at lave nogle eksperimenter med koden, hvor du tæller antal af operationer, dvs. ser på returværdien for forskellige værdier af parameteren (N) for at skitsere et estimat af Store O tidskompleksiteten.

Opgave 3 (18 %)

Skriv en *rekursiv* metode/algoritme med følgende signatur:

```
int antalVokaler(String str,int l)
```

Parameteren `str` indeholder små bogstaver i det engelske alfabet:

{a,b,c,d,e,f,g,h,i,j,k,l,m,n,o,p,q,r,s,t,u,v,w,x,y,z}.

Og metoden skal returnere antallet af vokaler i strengen `str`, og vi antager, at vokalerne er:

{a,e,i,o,u,y}

Kaldt med strengen ”stationsbygninger” returneres 6, eftersom vokalerne i strengen er ”a,i,o,y,i,e”.

Den anden parameter i metoden angiver længden af strengen minus 1, idet det antages, at strengen opfattes som et array af karakterer (chars).

Det kan anbefales at skrive en hjælpemetode med returværdi `true` ellers `false`, som kan afgøre om den enlige parameter af datatypen `char`, er en vokal eller en konsonant.

Opgave 4 (15 %)

Opgaven går ud på at skrive en metode, der kan sortere et array, som altid består af 100 positive heltal, hvorom det gælder, at alle tal i tabellen er større end eller lig med en (1) og mindre end eller lig med 200.

Metoden har ingen returværdi, men skal blot udskrive tallene fra tabellen i stigende rækkefølge, og det er ikke tilladt at benytte sorteringsmetoder fra standardbibliotekerne.

For at få fuldt points skal din metode have lineær - O(N) - tidskompleksitet.

Metoden kunne have følgende signatur:

```
void minSortering(int[] arr);
```

Opgave 5 (15 %)

En prioritetskø kan repræsenteres af et array, hvor indeks 0 altid har værdien 0 og som udgangspunkt ikke benyttes.

Betragt følgende array:

{0,9,23,106,10,36,38,98,84,12,14,50,55,35,68}

Dette array udgør **ikke** en prioritetskø.

Opgaven går ud på at finde de elementer, som forbryder sig imod/ødelægger *heap-order,* og at angive det interval af værdier, de skulle antage, for at (gen)skabe *heap-order.* Det forudsættes, at der ikke må være dubletter i en prioritetskø.

Hvis det opdaterede array, altså prioritetskøen, (minus indeks 0) blev implementeret/set som et binært træ, hvilke egenskaber ville dette træ så have?

Opgave 6 (15 %)

Denne opgave går ud på at etablere to hashtabeller, begge med plads til 16 elementer.

I den første (7 %) skal elementerne med nøglerne D,E,M,O,C,R,A,T indsættes.

Hashfunktionen er

(11*k) % tableSize

hvor k er tallets placering/nummer i det engelske alfabet (se opgave 3).

Eksempel: hash('C') = hash(3) = 11*3%16=33%16=1

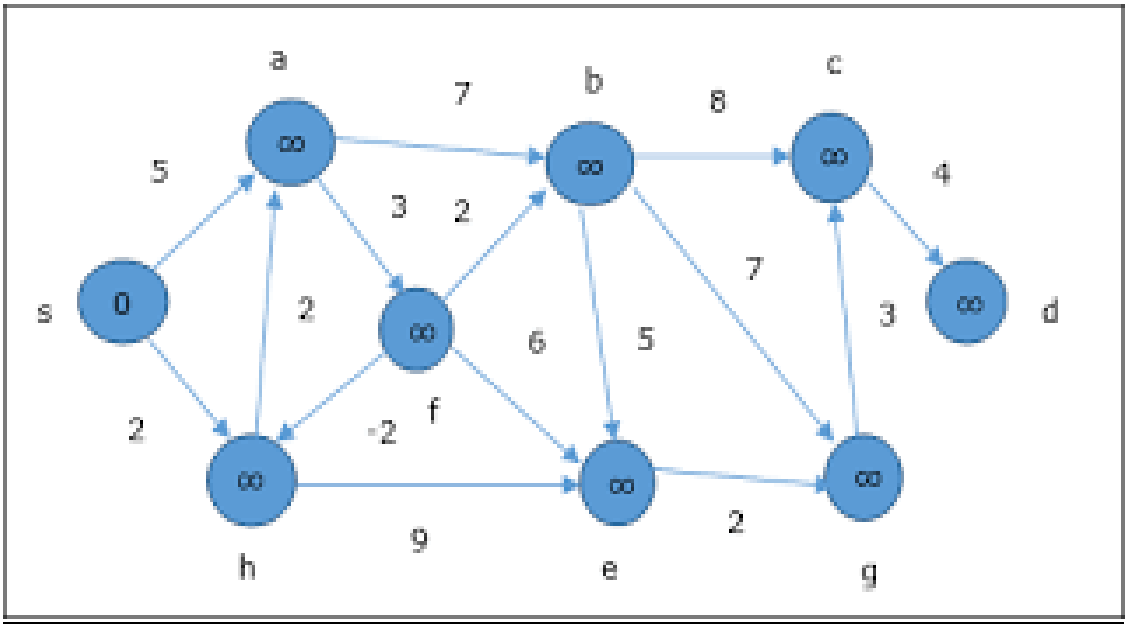
Ved kollisioner (et element hasher til et optaget indeks) anvendes *linear probing.*

I den anden tabel (8 %) skal elementerne med nøglerne R,E,P,U,B,L,I,C,A,N indsættes.

Der anvendes samme hashfunktion som i den første og *quadratic probing* ved kollisioner.

Opgave 7 (15 %)

Betragt nedenstående graf



Bemærk: den lille plet på billedet mellem noderne f og h er ikke et minus. Vægten mellem f og h er 2.

(opgaveteksten fortsætter på næste side)

I nedenstående tabel er angivet en fejlbehæftet slutkonfiguration for traversering af grafen ved anvendelse af *Dijkstras algoritme* med start i node s. Første del af opgaven går ud på at rette konfigurationen, så den bliver korrekt.

v	**known**	$\mathbf{d_v}$	$\mathbf{p_v}$
S	false	0	0
A	false	5	s
B	false	12	a
C	false	20	b
D	false	20	c
E	false	11	h
F	false	7	a
G	false	13	e
H	false	2	a

Anden del af opgaven er at udarbejde et *minimum spanning tree* for grafen, hvis den var undirected. Angiv hvilken algoritme, der er anvendt og træets vægt.

Du behøver ikke at tegne grafen; men du skal, under alle omstændigheder, skrive, i hvilken rækkefølge kanterne (edges) tilføjes. En god løsning kunne derfor være noget lignende nedenstående (som vedrører en anden graf end ovenstående):

Kant nr.	Fra	Til	Vægt
1	C	F	2
2	H	J	3
3	B	D	3