Tentamensuppgifter

Exempeluppgifter – kan uppdateras

Använd bilder och kod i dina svar när det passar. Förklara vad som händer. Framhäva huvudidéer. Identifiera speciella fall.

Uppgift 1

Skapa en rekursiv metod som bestämmer summan av första n naturliga heltal.
Vad händer när metoden anropas med argument *5*? Rita den motsvarande anropskedjan.
Bestäm djupet av rekursionen. Hur många records ska maximalt finnas på systemstacken? Vad innehåller dessa records?
Bevisa att metoden korrekt beräknar summan.

Finns det en bättre lösning för samma problem?

Uppgift 2

Skapa en rekursiv metod som bestämmer n-te Fibonaccital.
Rita rekursionsträdet när metoden anropas med argument *5*. Varför är metoden så ineffektiv för stora n?

Skapa en iterativ metod som bestämmer n-te Fibonaccital. Är den effektivare än den rekursiva metoden? Varför?

Uppgift 3

Skapa en rekursiv metod som flyttar n ringar från ett torn till ett annat torn, som i spelet ”Tornen i Hanoi” (eng. ”Towers of Hanoi”).
Bevisa att algoritmen är korrekt.

Uppgift 4

Bestäm storleksordning för följande komplexitetsfunktioner.
$g_1(n) = 3n^2 + 100n^{¨} + 5$
$g_2(n) = 3n^2 + 0.1\, n^3$
$g_3(n) = 5 + 10n + log_2$ n
$g_4(n) = 5 + 10n + nlog_2$ n
$g_5(n) = 2^n + 3^n$
$g_6(n) = 2^n + 5n!$

Uppgift 5

Definiera mängden $\Theta(n^2)$.
Ange en komplexitetsfunktion som tillhör till mängden $\Theta(n^2)$. Bevisa att den angivna komplexitetsfunktionen tillhör till denna mängd. Rita motsvarande asymptotiska begränsningar för komplexitetsfunktionen.

Ange en komplexitetsfunktion som inte tillhör till mängden $\Theta(n^2)$. Bevisa att den angivna komplexitetsfunktionen inte tillhör till denna mängd.

Uppgift 6

Definiera mängden $O(n^2)$.

Ange en komplexitetsfunktion som tillhör till mängden $O(n^2)$. Bevisa att den angivna komplexitetsfunktionen tillhör till denna mängd. Rita motsvarande asymptotisk begränsning för komplexitetsfunktionen.

Ange en komplexitetsfunktion som inte tillhör till mängden $O(n^2)$. Bevisa att den angivna komplexitetsfunktionen inte tillhör till denna mängd.

Uppgift 7

Definiera mängden $\Omega(n^2)$.

Ange en komplexitetsfunktion som tillhör till mängden $\Omega(n^2)$. Bevisa att den angivna komplexitetsfunktionen tillhör till denna mängd. Rita motsvarande asymptotisk begränsning för komplexitetsfunktionen.

Ange en komplexitetsfunktion som inte tillhör till mängden $\Omega(n^2)$. Bevisa att den angivna komplexitetsfunktionen inte tillhör till denna mängd.

Uppgift 8

Skapa en metod som sorterar en heltalssekvens. Använd urvalssorteringen (eng. ”selection sort”).

Bestäm algoritmens tidskomplexitet.

Uppgift 9

Skapa en metod som sorterar en heltalssekvens. Använd insättningssorteringen (eng. ”insertion sort”)..

Bestäm algoritmens tidskomplexitet i värsta fall.

Bestäm algoritmens tidskomplexitet i ett genomsnittligt fall.

Uppgift 10

Skapa en metod som sorterar en heltalssekvens. Använd ”merge sort”.

Bestäm algoritmens tidskomplexitet i värsta fall.

Vilken storleksordning har den komplexitetsfunktion som ger algoritmens genomsnittliga tidskomplexitet?

Uppgift 11

Jämför tidskomplexiteten och minneskomplexiteten för ”quick sort” och ”merge sort”.

Uppgift 12

Skapa en rekursiv metod som söker ett givet heltal i en sorterad heltalssekvens. Använd binär sökning (eng. ”binary search”).

Bestäm algoritmens tidskomplexitet i värsta fall.

Uppgift 13

Implementera en stack. Använd en vektor som underliggande datastruktur.

Uppgift 14

Implementera en stack. Använd en sekvens av noder som underliggande datastruktur.

Uppgift 15

Skapa en metod som undersöker om parenteser i en text är i balans. Använd en stack.

Uppgift 16

Skapa en metod som beräknar ett heltalsuttryck givet i postfix notationen. Använd en stack.

Uppgift 17

Implementera en FIFO-kö. Använd en vektor som underliggande datastruktur.

Uppgift 18

Implementera en FIFO-kö. Använd en sekvens av noder som underliggande datastruktur.

Uppgift 19

En heap skapas ifrån följande heltal: *5, 7, 4, 6, 2, 1, 9, 0, 8, 3*. Ju större ett heltal är, desto större prioritet det har.
Ett heltal tas sedan ut från heapen.

Rita en serie bilder som illustrerar insättningarna och uttagningen.

Uppgift 20

Rita en heap med heltal. Lagra den sedan i en vektor.

Vilka matematiska relationer som finns mellan en förälder och dess barn när en heap lagras i en vektor? Bevisa det.

Uppgift 21

En heap innehåller n element. Bestäm antalet nivåer i heapen.

Bestäm tidskomplexiteten för insättningsoperationen och borttagningsoperationen i värsta fall.

Uppgift 22

Följande heltalssekvens sorteras med ”heap sort”: *25, 17, 36, 2, 3, 99, 1, 19, 7*.

Rita en serie bilder som visar sekvensens omvandlingar under sorteringen.

Uppgift 23

Skapa en metod som bestämmer om ett givet element finns i den aktuella mängden.

Skapa metoden både i fall att mängden implementeras med en vektor och i fall att mängden implementeras med en sekvens av länkade noder.

Uppgift 24

Skapa en metod som tar bort ett givet element från den aktuella mängden.

Skapa metoden både i fall att mängden implementeras med en vektor och i fall att mängden implementeras med en sekvens av länkade noder.

Uppgift 25

Skapa en metod som returnerar en iterator till den aktuella mängden.

Skapa metoden både i fall att mängden implementeras med en vektor och i fall att mängden implementeras med en sekvens av länkade noder.

Skapa sedan en mängd och en iterator till den, och använd iteratorn för att visa mängdens element.

Uppgift 26

Skapa en metod som sätter in ett givet element på en given position i den aktuella listan.

Skapa metoden både i fall att listan implementeras med en vektor och i fall att listan implementeras med en sekvens av dubbellänkade noder.

Uppgift 27

Skapa en metod som tar bort från den aktuella listan det element som finns på en given position.

Skapa metoden både i fall att listan implementeras med en vektor och i fall att listan implementeras med en sekvens av dubbellänkade noder.

Uppgift 28

Varför kan en ”backtracking” algoritm vara mycket snabbare än en ”brute force” algoritm?
Illustrera det i samband med ”*n*-queens problem” (placera *n* damer på ett *n x n* schackbräde).

Uppgift 29

Följande element läggs till i tur och ordning till ett binärt sökträd: *M, T, P, G, U, J, O, R* och *V*. Sedan tas bort *T* från trädet. Till sist balanseras trädet.

Rita en serie bilder som visar trädet efter varje enskild operation.

Uppgift 30

Skapa en metod som skriver ut element i det aktuella binära sökträdet i stigande ordning.

Uppgift 31

Skapa en metod som bestämmer antalet element i det aktuella binära sökträdet.

Skapa metoden både med och utan rekursion.

Uppgift 32

Skapa en metod som söker ett givet element i det aktuella binära sökträdet.

Skapa metoden både med och utan rekursion.

Uppgift 33

Skapa en metod som lägger till ett givet element till det aktuella binära sökträdet.

Skapa metoden både med och utan rekursion.

Uppgift 34

I ett binärt sökträd lagras uppgifter av formen `(Integer, String)`. Först lagras följande uppgifter:

```
(5, ”five”), (9, ”nine”), (7, ”seven”), (3, ”three”), (8, ”eight”), (15,
“fifteen”), (6, “sex”), (5, “Five”), (20, “twenty”), (10, “ten”), (25,
“twentyfive”)
```

Efter det letar man efter det element som har nyckel `7`. Till sist tar man bort det element som har nyckel `15`.

Illustrera de angivna operationerna med en serie bilder.

Uppgift 35

En hashtabell har *4* rader, som implementeras med sekvenser av länkade noder. I tabellen lagras uppgifter av formen `(Integer, String)`. För att hantera de olika uppgifterna används följande hashfunktion:

Math.abs (key.hashCode ()) % table.length

Metoden `hasCode` i klassen `Integer` returnerar ett heltal som är likadant som det heltal som inkapslas i objektet.

Först lagras följande uppgifter i hashtabellen:

```
(5, ”five”), (9, ”nine”), (7, ”seven”), (3, ”three”), (8, ”eight”), (15,
“fifteen”), (6, “sex”), (5, “Five”), (20, “twenty”), (10, “ten”), (25,
“twentyfive”)
```

Efter det letar man efter det element som har nyckel `7`. Till sist tar man bort det element som har nyckel `15`.

Illustrera de angivna operationerna med en serie bilder.
Vad skulle hända om man använde någon av följande hashfunktioner:

Math.abs (key.hashCode ()) % 2
Math.abs (key.hashCode ())

Vad skulle hända om tabellen hade *15* rader?

Uppgift 36

Hur kan man lagra information om en grafs noder?
Hur kan man använda en grannmatris för att lagra information om en grafs kanter?
Hur kan man använda en vektor av nodsekvenser för att lagra information om en grafs kanter?

Uppgift 37

Djupet-först sökning (eng. depth-first searching) och bredden-först sökning (eng. breadth-first searching) är två algoritmer som används för sökning i en graf.
Rita en graf, och illustrera användningen av dessa algoritmer i samband med den.

Uppgift 38

Dijkstras algoritm hittar de kortaste vägarna från en nod till alla andra noder i en riktad, viktad graf (”single-source shortest-paths problem”).

Rita en graf, och illustrera användningen av denna algoritm i samband med den.

Uppgift 39

Prims algoritm bestämmer ett minimalt uppspännande träd i en sammanhängande, viktad graf.

Rita en graf, och illustrera användningen av denna algoritm i samband med den.

Uppgift 40

Kruskals algoritm bestämmer ett minimalt uppspännande träd i en sammanhängande, viktad graf.

Rita en graf, och illustrera användningen av denna algoritm i samband med den.