Oblig 1 IN2010

Oppgave 1.

1. Se fil BinaertSoketre.java for kjørbar kode
2. Den ineffektive er O(n) kjøretid og den effektive er O(logn) kjøre tid

Oppgave 2.

Prosedyre: push\_back(int x)

Lager en ny node med x som innhold

Sjekker om første- og siste-peker er lik null

Legger noden til køen og oppdaterer pekere

Legger så til noden i arrayet

Oppdaterer pekere for den nye noden og sistenoden

Legger den nye noden inn i arrayet på slutten

Prosedyre: push\_front(int x)

Lager en ny node med x som innhold

Sjekker om første- og siste-peker er lik null

Legger noden til i køen og oppdaterer pekere

Legger den nye noden inn i arrayet

Oppdaterer pekere for den nye noden og førstenoden

Legger den nye noden inn i arrayet på starten

Prosedyre: push\_middle(int x)

Lager en nye node med x som innhold

Sjekker om første- og siste-peker er lik null

Legger noden til i køen og oppdaterer pekere

Legger så noden inn i arrayet

Sjekker så om midt-pekeren er lik null

Oppdaterer midt-pekeren

Oppdaterer pekere og legger den nye noden inn i køen

Legger den nye noden inn i arrayet

Oppdaterer midt-pekeren

Prosedyre: get(int x)

Returnerer innholdet på plassen x i arrayet

1. Se fil Teque.java for kjør bar kode
2. O(1) på alle operasjonene
3. Hvis du vet det verste tilfelle for antall operasjoner kan du substituere n med det tallet, altså i dette tilfellet 10^6. O(10^6) blir O(1), og dermed vil alle operasjoner bli konstanttid.

Oppgave 3.

1. s 🡨 skanner

kp 🡨 s.nextLine

h 🡨 new hashmap

l 🡨 scanner.nextLine

While not –1 then

s[] 🡨 l.split()

for i 🡨 string[]

h.put(s[i], s[0])

l 🡨 s.nextLine

v 🡨 “ ”

v 🡨 + kp

while h.contains(kp) then

kp 🡨 h.get(kp)

v 🡨 + “ ” + kp

1. Se fil Kattunge.java for kjørbar kode

Oppgave 4.

1. Se fil Balanserttre.java for kjørbar kode

x er en ArrayList<Integer>

prosedyre: balanser(x)

y 🡨 x.størrelse/2

hL 🡨 ArrayList

vL 🡨 ArrayList

if x.størrelse != 0 do

print(x.get(y))

for i = y+1, i < x.størrelse do

hL.add(x.get(i))

for i = 0, i < y do

vL.add(x.get(i))

balanser(hL)

balanser(vL)

1. Se fil BalanserttreHeap.java for kjørbar kode

x er en PriorityQueue<Integer>

Prosedyre: balanser(x)

bH 🡨 priorityQueue

y 🡨 x.størrelse / 2

hH 🡨 PriorityQueue

vH 🡨 PriorityQueue

if x.størrelse != 0 do

i 🡨 0

while x.størrelse > 0 do

if i < y do

vH.offer(x.poll())

else if i = y do

z 🡨 x.poll

bH.offer(z)

else

hH.offer(x.poll())

i++

balanser(hH)

balanser(vH)