Øving 2, teori: Datastrukturer

Question 1: Kø

Oppgaven ble godkjent!

Anta at du har en kø

$Q = \langle 4, 7, 32, 72, 3 \rangle$

hvor det første elementet representerer hodet til køen som definert i kapittel 10.1 i læreboken.

De neste tre deloppgavene er uavhengige av hverandre; dvs. alle refererer til denne køen.

Hvordan vil køen se ut etter å ha kjørt `ENQUEUE(Q,3)` ?

- ○ $\langle 4, 7, 32, 72, 3 \rangle$
- ● $\langle 4, 7, 32, 72, 3, 3 \rangle$
- ○ $\langle 3, 4, 7, 32, 72, 3 \rangle$
- ○ $\langle 4, 7, 32, 72 \rangle$

Question 2: Kø

Oppgaven ble godkjent!

$Q = \langle 4, 7, 32, 72, 3 \rangle$

Hvordan vil køen se ut etter å ha kjørt `DEQUEUE(Q)` ?

- ○ $\langle 4, 7, 32, 72, 3, 3 \rangle$
- ● $\langle 7, 32, 72, 3 \rangle$
- ○ $\langle 4, 7, 32, 72, 3 \rangle$
- ○ $\langle 4, 7, 32, 72 \rangle$
- ○ $\langle 3, 4, 7, 32, 72, 3 \rangle$

Question 3: Kø

Oppgaven ble godkjent!

Riktig, her må man kalle `DEQUEUE` 4 ganger (slik at køen kun inneholder 3) og deretter kalle `ENQUEUE` ganger 5 med de resterende elementene

$Q = \langle 4, 7, 32, 72, 3 \rangle$

Hva er det minste antallet `ENQUEUE` -/ `DEQUEUE` -operasjoner du trenger for at køen Q skal endres til $\langle 3, 4, 7, 32, 72, 3 \rangle$?

- 2
- 9
- 6
- 1
- 5
- 11

Question 4: Stakk

Oppgaven ble godkjent! ×

Anta du har en stakk

$S = \langle 4, 7, 32, 72, 3 \rangle$

hvor det bakerste elementet representerer toppen av stakken slik som de definerer i kapittel 10.1 i læreboken.

Hvordan vil stakken se ut etter å ha kjørt `PUSH(S,3)` ?

- $\langle 4, 7, 32, 72 \rangle$
- $\langle 4, 7, 32, 72, 3 \rangle$
- $\langle 4, 7, 32, 72, 3, 3 \rangle$
- $\langle 7, 32, 72, 3 \rangle$
- $\langle 3, 4, 7, 32, 72, 3 \rangle$

Question 5: Stakk

Oppgaven ble godkjent! ×

$S = \langle 4, 7, 32, 72, 3 \rangle$

Hvordan vil stakken se ut etter å ha kjørt `POP(S)` ?

- $\langle 3, 4, 7, 32, 72, 3 \rangle$
- $\langle 7, 32, 72, 3 \rangle$
- $\langle 4, 7, 32, 72, 3 \rangle$
- $\langle 4, 7, 32, 72 \rangle$
- $\langle 4, 7, 32, 72, 3, 3 \rangle$

Question 6: Stakk

Oppgaven ble godkjent! ×

$S = \langle 4, 7, 32, 72, 3 \rangle$

Hva er det minste antallet `PUSH` -/ `POP` -operasjoner du trenger for at stakken S skal endres til $\langle 3, 4, 7, 32, 72, 3 \rangle$?

- (•) 11
- ( ) 1
- ( ) 2
- ( ) 5
- ( ) 0
- ( ) 6

Question 7: Sirkulær dobbel-lenket liste

Oppgaven ble godkjent! ×

Anta at du har en sirkulær dobbel-lenket liste

$L = \langle 4, 7, 32, 72, 3 \rangle$

hvor hodet peker på 4-tallet.

De neste fem deloppgavene er uavhengige av hverandre; dvs. alle refererer til denne listen.

Hvordan vil listen se ut etter `LIST-SEARCH(L,4)` som definert i kapittel 10.2 i læreboken?

- ( ) $\langle 4, 7, 32, 72, 3, 3 \rangle$
- (•) $\langle 4, 7, 32, 72, 3 \rangle$
- ( ) $\langle 4, 7, 32, 72 \rangle$
- ( ) $\langle 3, 4, 7, 32, 72, 3 \rangle$
- ( ) $\langle 7, 32, 72, 3 \rangle$

Question 8: Sirkulær dobbel-lenket liste

Oppgaven ble godkjent! ×

$L = \langle 4, 7, 32, 72, 3 \rangle$

Hvordan vil listen se ut etter `LIST-INSERT(L,x)` for en node x med $x.key = 3$?

- $\langle 7, 32, 72, 3 \rangle$
- $\langle 3, 4, 7, 32, 72, 3 \rangle$
- $\langle 4, 7, 32, 72, 3, 3 \rangle$
- $\langle 4, 7, 32, 72, 3 \rangle$
- $\langle 4, 7, 32, 72 \rangle$

Question 9: Sirkulær dobbel-lenket liste

Oppgaven ble godkjent! ×

$L = \langle 4, 7, 32, 72, 3 \rangle$

Hvordan vil listen se ut etter `LIST-DELETE(L,x)` for noden x der $x.key = 4$?

- $\langle 7, 32, 72, 3 \rangle$
- $\langle 3, 4, 7, 32, 72, 3 \rangle$
- $\langle 4, 7, 32, 72, 3 \rangle$
- $\langle 4, 7, 32, 72, 3, 3 \rangle$
- $\langle 4, 7, 32, 72 \rangle$

Question 10: Implementering av pekere og objekter

Oppgaven ble godkjent! ×

$L = \langle 4, 7, 32, 72, 3 \rangle$

Vi ønsker å implementere L som en tabell av objekter av objekter tilsvarende som i Cormen (figur 10.5 s.242). Hvilket av alternativene under for startvariabel I og arrayene $N = next$, $K = key$ og $P = prev$ er korrekt implementert?

- $I = 2,\ N = \langle /, 6, 1, 3, 0, 4 \rangle,\quad K = \langle 3, 4, 72, 32, 0, 7 \rangle,\quad P = \langle 3, /, 4, 1, 0, 2 \rangle$
- $I = 4,\ N = \langle 8, 6, /, 1, 0, 0, 0, 2 \rangle,\ K = \langle 7, 32, 72, 4, 0, 0, 0, 3 \rangle,\ P = \langle 4, 8, 2, /, 0, 0, 0, 1 \rangle$
- $I = 1,\ N = \langle 2, 3, 4, 5, / \rangle,\quad K = \langle 4, 7, 32, 72, 3 \rangle,\quad P = \langle 1, 2, 3, 4, / \rangle$
- $I = 7,\ N = \langle 3, 0, 4, /, 0, 1, 6, 0 \rangle,\ K = \langle 32, 0, 72, 3, 0, 7, 4, 0 \rangle,\ P = \langle 6, 0, 1, 3, 0, 7, /, 0 \rangle$

Question 11: Implementering av pekere og objekter

Oppgaven ble godkjent! ×

$L = \langle 4, 7, 32, 72, 3 \rangle$

Vi ønsker å implementere L som en tabell av objekter tilsvarende som i Cormen (figur 10.6 s.243). Hvilket av alternativene under for startvariabel I og array A er korrekt implementert?

- $I = 13,\ A = \langle 0, 0, 0, 7, 1, 13, 32, 4, 22, 0, 0, 0, 4, 13, /, 3, /, 4, 0, 0, 0, 72, 16, 1 \rangle$
- $I = 4,\ \ A = \langle 72, 16, 13, 4, 10, /, 32, 13, 10, 7, 7, 4, 0, 0, 0, 3, /, 1 \rangle$
- **$I = 4,\ \ A = \langle 72, 13, 10, 4, 7, /, 7, 10, 4, 32, 1, 7, 3, /, 1 \rangle$**
- $I = 19,\ A = \langle 32, 7, 13, 0, 0, 0, 72, 16, 1, 0, 0, 0, 7, 1, 19, 3, /, 9, 4, 13, /, 0, 0, 0 \rangle$

Question 12: Hashfunksjon

Oppgaven ble godkjent! ×

Du får oppgitt at $x.key = m$ og $h(m) = j$ der h er en hashfunksjon. Da er...

- x elementet, j nøkkelen og m hashen.
- ingen av de andre alternativene korrekt
- j elementet, m nøkkelen og x hashen.
- **x elementet, m nøkkelen og j hashen.**

Question 13: Hash-funksjon

Oppgaven ble godkjent! ×

Hva betyr kollisjon (eng. collision) i forbindelse med hashtabeller?

- Begge alternativene.
- **Flere ulike faktiske nøkler gir samme hashverdi.**
- Flere ulike hashverdier gir samme faktiske verdi.
- Ingen av alternativene.

Question 14: Hashfunksjon

Oppgaven ble godkjent! ×

En god hashfunksjon vil, for en tabell av lengde n, kunne garantere at $k < n$ innsettinger ikke vil gi kollisjon?

- (•) Nei
- ( ) Ja

Question 15: Hashfunksjon

Oppgaven ble godkjent!

En hashfunksjon må være en matematisk funksjon og kan dermed ikke være randomisert (vi må ha egenskapen at `h(k)` gir samme verdi dersom vi regner den ut flere ganger).

Er `h(k) = (k * rand(1:k)) mod m` hvor `k` er nøkkelen og `m` er størrelsen på hashtabellen en god hashfunksjon?

- ( ) Ja
- (•) Nei

Question 16: Kjedet hashtabell

Oppgaven ble godkjent!

Hvis vi har en funksjon `DELETE(T,x)` der T er en kjedet hashtabell og x er en listenode, så er worst case kjøretid... (Legg merke til at x her er en faktisk listenode – ikke en nøkkel)

- ( ) $O(n)$ både for enkel- og dobbel-lenket liste.
- ( ) $O(1)$ for enkel-lenket liste og $O(n)$ for dobbel-lenket liste.
- ( ) $O(1)$ både for enkel- og dobbel-lenket liste.
- (•) $O(n)$ for enkel-lenket liste og $O(1)$ for dobbel-lenket liste.

Question 17: Kjedet hashtabell

Oppgaven ble godkjent!

I worst-case har alle elementene samme hashverdi og vi har derfor en lenket liste med n elementer. Det tar O(n) tid å søke gjennom listen.

Hva er worst-case-kjøretiden for innsetting i en hashtabell om man bruker kjeding ved kollisjoner? Anta at innsettingen også må sjekke om elementet allerede finnes i tabellen.

- ( ) $O(n \lg(n))$
- ( ) $O(1)$

- ○ $O(\lg(n))$
- ◉ $O(n)$

Question 18: Amorisert analyse

Oppgaven ble godkjent! ×

For å unngå at vi lager for stor initiell hashtabell ønsker vi å doble størrelsen på hashtabellen hver gang lastfaktoren blir ¼ (lastfaktor beregnes N/M hvor N er antall elementer i hashtabellen og M er størrelsen på hashtabellen). Hvis vi benytter amortisert analyse får vi at kjøretiden for innsetting er...

- ○ $O(\lg(n))$
- ○ $O(n \lg(n))$
- ○ $O(n)$
- ◉ $O(1)$

Question 19: Binærtre

Oppgaven ble godkjent! ×

Et tre har egenskapen at, for enhver to noder, finnes det nøyaktig én sti mellom dem. Dermed vil det, om vi legger til en ny kant mellom to noder, finnes to stier mellom disse nodene. Da har det oppstått en sykel.

Anta du har binærtre G og legger til én ny kant i treet. Du vil nå ha...

- ○ ingen av de andre alternativene.
- ○ et tre med høyere grad.
- ◉ en syklisk graf.
- ○ et binærtre med én kant mer.

Question 20: Amortisert analyse

Oppgaven ble godkjent! ×

Hvorfor er amortisert analyse bedre enn vanlig worst-case-beregning i mange tilfeller?

- ○ Amortisert analyse gir også en nedre grense.
- ◉ Worst case kan være altfor pessimistisk.
- ○ Amortisert analyse er raskere å beregne.
- ○ Worst case kan gi ugyldig svar.