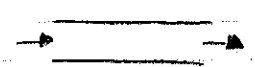


Queue



make-queue() : יצירת קופ"ח

(inject) enqueue(Q, x) : הכנסה

(eject) dequeue(Q) : יציאה מהקופ"ח
↓
קדימוניות

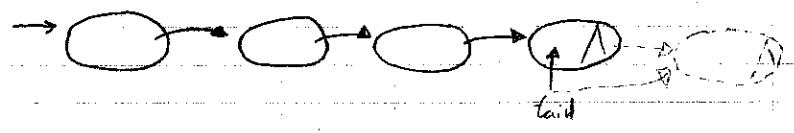
First in First Out

empty?(Q) : ריקנות

size(Q)

front(Q) : ערך בראש

מבנה נתון : I linked



make-queue()

```

head ← null
tail ← null
size ← 0

```

enqueue(Q, x)

```

Create a new node n
n.element ← x
n.next ← null
if size > 0 then tail.next ← n
                    tail ← n
else head ← n
      tail ← n

size ← size + 1

```

dequeue(Q)

```

if size = 0 then throw error
else temp ← head.element
  head ← head.next
  size ← size - 1
  if size = 0 then tail ← null
  return temp

```

מבנה נתונים (מבנה נתונים) - מבנה נתונים II

make-queue()

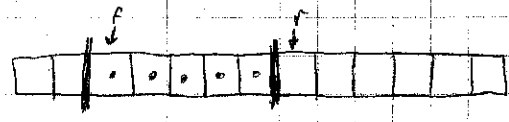
create an array A

f ← 0
r ← 0

enqueue(Q, x)

size(Q)

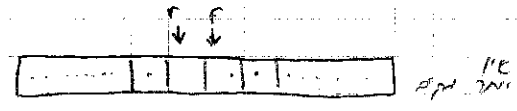
if $r \geq f$ then return $r - f$
else return $r - f + \text{length}(A)$



enqueue(Q, x)

if $\text{size}(Q) \geq \text{length}(A) - 1$ then throw error (מבנה נתונים מלא)

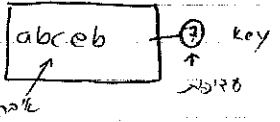
$A[r] \leftarrow x$
 $r \leftarrow (r+1) \bmod \text{length}(A)$



dequeue(Q)

if $\text{size}(Q) = 0$ then error
else $f \leftarrow (f+1) \bmod \text{length}(A)$
return $A[f-1]$

Priority queue



מבנה נתונים המיון - מבנה נתונים

make-priority-queue()

insert(Q, x)

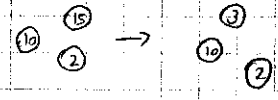
extract-min(Q)

minimum(Q)

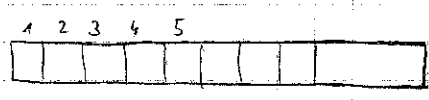
size

empty?

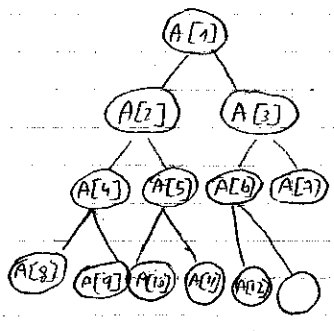
decrease-key(Q, x, key) - הקטן את המפתח של x במבנה נתונים



מחיצת קומוניקציה - ש"ר 6



מחיצת קומוניקציה - ש"ר 6



אם $A[i]$ הוא קטן מכל ילדיו, אז $A[i]$ הוא המינימום.
 אם $A[i]$ הוא גדול מכל ילדיו, אז $A[i]$ הוא המקסימום.
 $A[2i-1]$ הוא ילדו הימני של $A[i]$.
 $A[2i]$ הוא ילדו השמאלי של $A[i]$.

אם $A[i]$ הוא קטן מכל ילדיו, אז $A[i]$ הוא המינימום.
 אם $A[i]$ הוא גדול מכל ילדיו, אז $A[i]$ הוא המקסימום.
 $A[2i-1]$ הוא ילדו הימני של $A[i]$.
 $A[2i]$ הוא ילדו השמאלי של $A[i]$.

אם $A[i]$ הוא קטן מכל ילדיו, אז $A[i]$ הוא המינימום.
 אם $A[i]$ הוא גדול מכל ילדיו, אז $A[i]$ הוא המקסימום.
 $A[2i-1]$ הוא ילדו הימני של $A[i]$.
 $A[2i]$ הוא ילדו השמאלי של $A[i]$.

make-priority-queue()

create an array
 size \leftarrow 0

minimum(Q)

if size = 0 then throw error
 else return $A[1]$

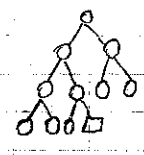
insert(Q, x)

size \leftarrow size + 1
 $i \leftarrow$ size

while $A[\lfloor \frac{i}{2} \rfloor] > x$ and $i > 1$
 do $A[i] \leftarrow A[\lfloor \frac{i}{2} \rfloor]$
 $i \leftarrow \lfloor \frac{i}{2} \rfloor$

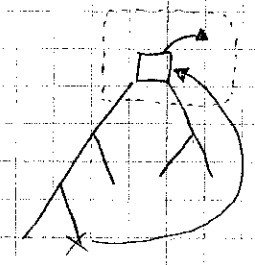
$A[i] \leftarrow x$

= decrease-key(Q, i, x)



extract-min(Q)

if size = 0 then throw error
 else min $\leftarrow A[1]$
 $A[1] \leftarrow A[\text{size}]$
 size \leftarrow size - 1
 heapify-down(A, 1)
 return min



heapify-down(A, i)

while $2i+1 < \text{size}$ and $\min\{A[2i], A[2i+1]\} < A[i]$
 do if $A[2i] < A[2i+1]$ then exchange $A[2i] \leftrightarrow A[i]$
 $i \leftarrow 2i$
 else exchange $A[2i+1] \leftrightarrow A[i]$
 $i \leftarrow 2i+1$

י"ד ב.
(אוב) 17
714

$i \neq z_i$ and $A[z_i] < A[i]$
then exchange $A[i] \leftrightarrow A[z_i]$

הערה: המינימום של המערך הוא $A[1]$ (במקרה של מערך מסוג $A[1..n]$)
המספר i הוא המיקום של $A[i]$ והמיקום z_i הוא המיקום של המינימום במערך $A[1..n]$.
המספר z_i הוא המיקום של המינימום במערך $A[1..n]$.
 $O(\log n)$ - א

המספר i הוא המיקום של $A[i]$

Heap-Sort ($A[1..n]$)

$Q \leftarrow \text{make-priority-queue}()$
for $i \leftarrow 1$ to n } $O(\log 1 + \log 2 + \dots + \log n) = O(n \log n)$
do $\text{insert}(Q, A[i])$
for $i \leftarrow 1$ to n } $O(\log 1 + \dots + \log n) = O(n \log n)$
do $B[i] \leftarrow \text{extract-min}(Q)$
 $O(n \log n) \rightarrow$ זמן הריצה