



מבני נתונים ואלגוריתמים

תור, מחסנית וערימה
פולינה לוצקר

מנהלות

- מרצה הקורס: פרופסור יורם לוזון
- מתרגלת: פולינה לוצקר
- אימייל: polinalutbiu@gmail.com
- שעות קבלה: יום שני 13:00–15:00 , בתיאום מראש.
- אתר הקורס: [math-wiki](#)
- דרישות קדם: לינארית 1, אינפי 1, **ידע בתכנות**.

תרגילי בית

- מדי שבוע יתפרסם תרגיל בית חדש.
- שבוע אחד תרגיל תכנותי ושבוע אחד תרגיל תאורטי.
- תרגילי תכנות הם להגשה –חובה!
 - ההגשה דרך `submit`
 - <https://submit.cs.biu.ac.il/cgi-bin/welcome.cgi>
 - יש לכתוב בשפת Python 3
 - פידבק ישלח לכם למייל אחרי ההגשה – הפידבק אינו הציון שלכם!
 - אין בדיקה ידנית
 - תהיה בדיקת העתקה
- התרגילים התאורטיים אינם להגשה אלא לאימון עצמי בלבד.

פסאודו קוד

ויקפדיה: תיאור מצומצם ולא רשמי לאלגוריתם של תוכנית מחשב. פסאודו קוד משתמש בקונבנציות של שפות תכנות, אך מיועד לקריאה של בני אדם ולא לקריאה על ידי מחשב. הביטויים שנכתבים בפסאודו קוד אינם ניתנים ל**הידור** (עיבוד הטקסט על ידי מחשב) בפני עצמם, אך הם ניתנים לתכנות כקוד אמיתי שכן ניתן להידור בכל שפת תכנות שהיא.

מחסנית

LIFO=LAST IN FIRST OUT •

תומך בפעולות:

init •

Push •

isEmpty •

top •

Pop •

(Destack) •

ניתן ליישום באמצעות מערך או באמצעות רשימה מקושרת.

היעילות- $O(1)$ (פרט לריקון מחסנית המיושמת בעזרת רשימה- $O(n)$)

תור

FIFO=FIRST IN FIRST OUT •

תומך בפעולות:

init •

insert •

isEmpty •

head •

top •

(DeQueue) •

ניתן ליישום באמצעות מערך או באמצעות רשימה מקושרת.

היעילות- $O(1)$ (פרט לריקון מחסנית המיושמת בעזרת רשימה- $O(n)$) •

תרגיל 1

יש שתי מחסניות, אחת מלאה ואחת ריקה.
כתבו אלגוריתם המעביר את תוכן המחסנית הראשונה למחסנית השנייה ומרוקן את השנייה.

תרגיל 1

יש שתי מחסניות, אחת מלאה ואחת ריקה.
כתבו אלגוריתם המעביר את תוכן המחסנית הראשונה למחסנית השנייה ומרוקן את השנייה.

פתרון:

קלט: s_1 מלאה, s_2 ריקה.
-ניצור מחסנית ריקה t .

```
while S1.empty() == False:  
    T.push(S1.pop())  
while T.empty() == False:  
    S2.Push(T.pop())
```

מה היעילות?

תרגיל 2 – בדיקת תקינות סוגריים

תיאור הבעיה:

- ביטוי חשבוני תקין מבחינת סוגריים:
- יכול להכיל מספר לא מוגבל של סוגריים מסוגים שונים, ובלבד שיהיו מאוזנים.
- איזון הסוגריים מחייב שמספר הפותחים והסוגרים יהיה שווה בדיוק.
- לכל פותח יימצא סוגר מאותו סוג במקום המתאים.

• דוגמה לביטוי תקין:

• ((5))

• (7-[5-6])

• דוגמה לביטוי לא תקין:

• 5)

• (5-7 }

תרגיל 2 - בדיקת תקינות סוגריים-המשך

נניח ויש לנו סוגריים מהסוג $[\{ \}]$.

כתבו אלגוריתם המקבל כקלט ביטוי חשבוני s , ומחזיר `true` אם הביטוי תקין מבחינת סוגריים ו `false` אם לא.

תרגיל 2 - בדיקת תקינות סוגריים-פתרון

Create an empty stack T.

For ch in S:

 if ch is one of (,[,{:

 T.push(ch)

 else if ch in one of)],},:}

 if T.empty == True:

 return False

 if T.pop() doesn't match ch:

 return False

If T.empty == False:

 return False

Return True

תרגיל 2 - בדיקת תקינות סוגריים-פתרון

Create an empty stack T.

For ch in S:

if ch is one of (, [, {:

T.push(ch)

else if ch in one of)] }:

if T.empty == True:  בודק האם יש יותר סוגריים סוגרים מאשר פתוחים

return False

if T.pop() doesn't match ch:

return False

If T.empty == False:  בודק האם יש יותר סוגריים פותחים מאשר סוגרים

return False

Return True



תרגיל 3

ממשו תור בעזרת מחסניות.

רמז: יש להשתמש בשתי מחסניות – s_1 ו- s_2 .

תרגיל 3 – פתרון

פתרון:

הכנסה: הכנסה של איבר x תתבצע על ידי `s1.push(x)`.

הוצאה:

```
if s2.empty == False:
```

```
    return s2.pop()
```

```
Else If s1.empty == False:
```

```
    while s1.empty == False:
```

```
        s2.push(s1.pop())
```

```
    return s2.pop()
```

```
Else:
```

```
Return "empty"
```

מציאת אורך המסלול הקצר ביותר במבוך בינארי

תיאור הבעיה: בהינתן מטריצה $M \times N$ בינארית (כל איבר הינו 0 או 1) נרצה למצוא את המסלול הקצר ביותר בין נקודת התחלה לנקודת סיום. המסלול יכול להיווצר רק דרך תאים שערכם הינו 1. ניתן ללכת למעלה, למטה, ולצדדים (אין ללכת באסלון).

דרישה: היעילות $O(MN)$

מציאת אורך המסלול הקצר ביותר במבוך בינארי

דוגמה:

עבור המטריצה

1 1 1

1 0 1

1 1 1

נרצה למצוא מסלול מהנקודה $(0,0)$ עד לנקודה $(2,0)$

תשובה: 2

מציאת אורך המסלול הקצר ביותר במבוך בינארי

דוגמה:

עבור המטריצה

1 1 0

1 1 0

0 0 1

נרצה למצוא מסלול מהנקודה $(0,0)$ עד לנקודה $(2,2)$

תשובה: 1 - אין מסלול.

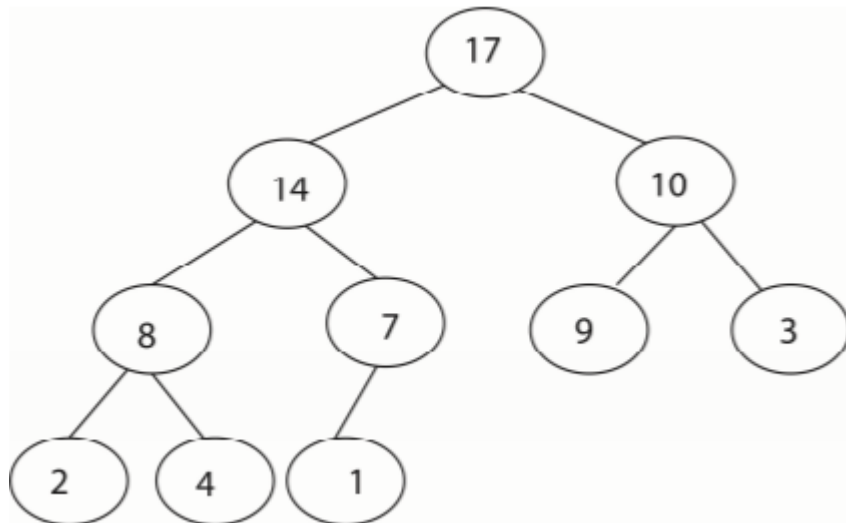
פתרון בעזרת תור

1. ניצור תור ריק ומטריצת עזר בגודל המסלול $(M \times N)$. תחילה זו תהיה מטריצת אפסים.
2. התור יכיל את הקואורדינטות של המטריצה (Point)
3. לכל נקודה ניצור סימון- האם ביקרנו בנקודה הזו או לא- נאתחל עבור כולם שלא ביקרנו.
4. נכניס את איבר ההתחלה לתור ונסמן שביקרנו.
5. כל עוד לא התור לא ריק:
הוצא איבר מהתור.
אם זו נקודת הסיום- החזר את אורך המסלול.
אחרת: עבור כל אחד מארבעת הצעדים האפשריים (התחשבו בקצוות) אם הצעד הינו 1
ולא ביקרנו בו- הכנס לתור ועדכן את אורך המסלול לנקודה זו להיות "הורה" $+1$,
וסמן ביקרנו בו.
6. אם הגענו לסוף התור ולא הגענו לנקודת היעד- החזר שאין מסלול.

ערימה

- מערך שניתן לראות כעץ בינארי כמעט מלא.
- עץ בינארי כמעט מלא = עץ בינארי שבו כל הרמות מלאות, פרט אולי לרמה האחרונה שמלאה מצד שמאל עד נקודה מסוימת.

- מציאת ערכים: אבא - $\lfloor \frac{i}{2} \rfloor$, בן שמאלי - $2i$, בן ימני - $2i + 1$.



גובה העץ = $\log_2(n)$

דוגמה: ערימת מקסימום : $A[\text{parent}[i]] \geq A[i]$

$A = [17, 14, 10, 8, 7, 9, 3, 2, 4, 1]$

הערה לגבי המימוש: יותר נוח לממש מאינדקס 1.

ערימה – המשך

פעולות ערימה:

צור ערימה $O(1)$

האם ריקה $O(1)$

הוצא ראש ערימה $O(\log n)$

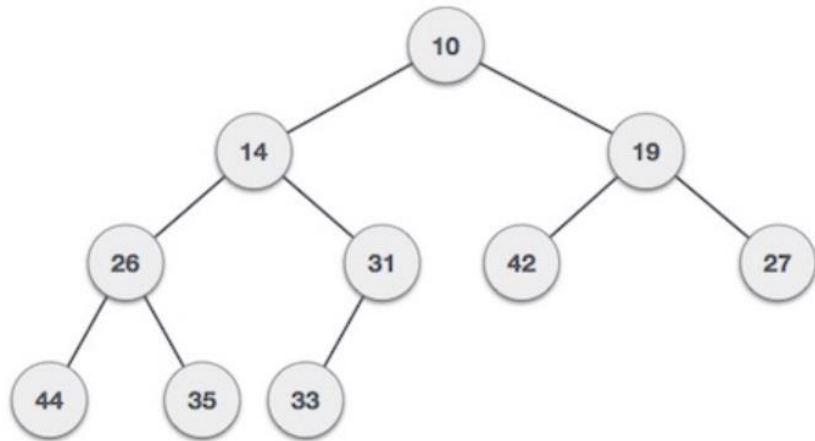
הכנס לערימה $O(\log n)$

(רוקן) $O(1)$

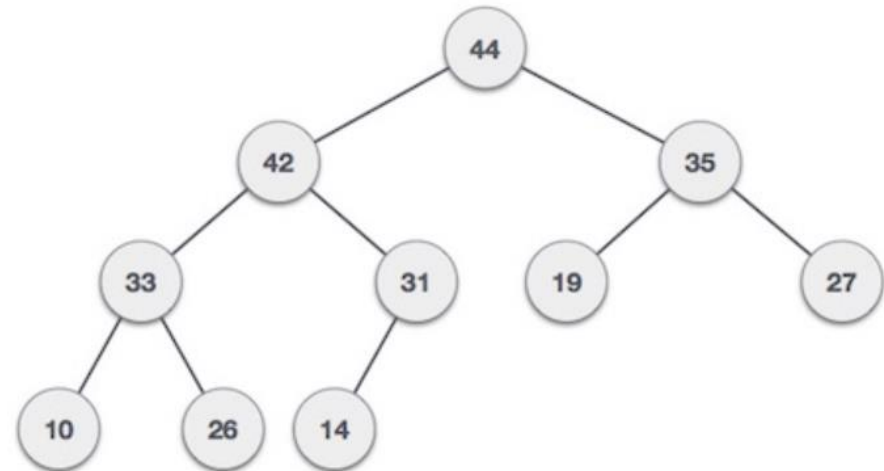
ערימות מינימום vs מקסימום

For input -> 35 33 42 10 14 19 27 44 26 31

Min-Heap – Where the value of the root node is less than or equal to either of its children.



Max-Heap – Where the value of the root node is greater than or equal to either of its children.



הוספת ערך לערימה

Push (x):

$A[end++] = x;$

$i = end;$

while $i > 1$ and $A[i] > A[Parent(i)]$

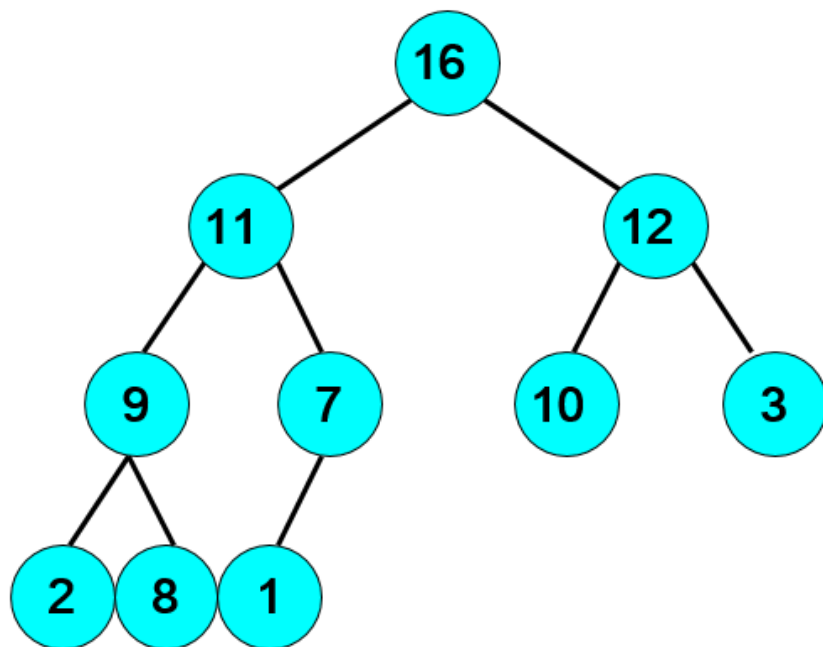
$k = Parent(i);$

$switch(A[i], A[k]);$

$i = k;$

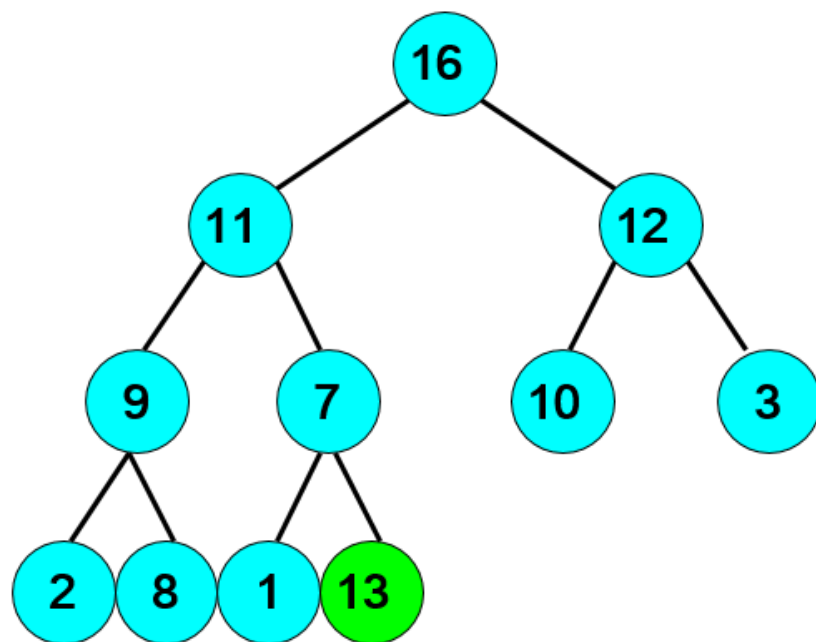
הוספת ערך לערימה-דוגמה

Push(x, Q)



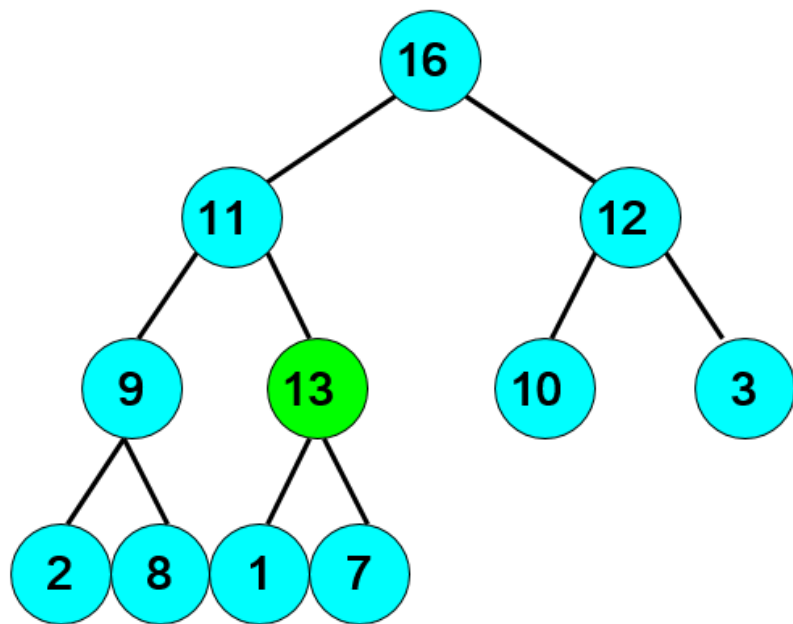
נכניס את הערך 13

הוספת ערך לערימה-דוגמה



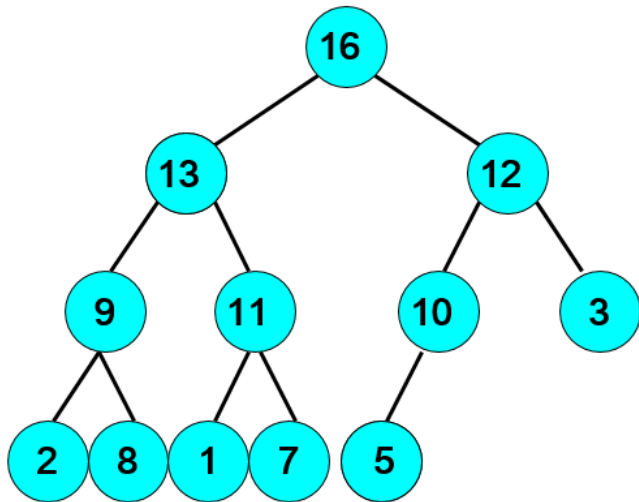
push(13, Q)

הוספת ערך לערימה-דוגמה



$\text{push}(13, Q)$

הוספת ערך לערימה-דוגמה



$\text{push}(13, Q)$

סבוכיות זמן

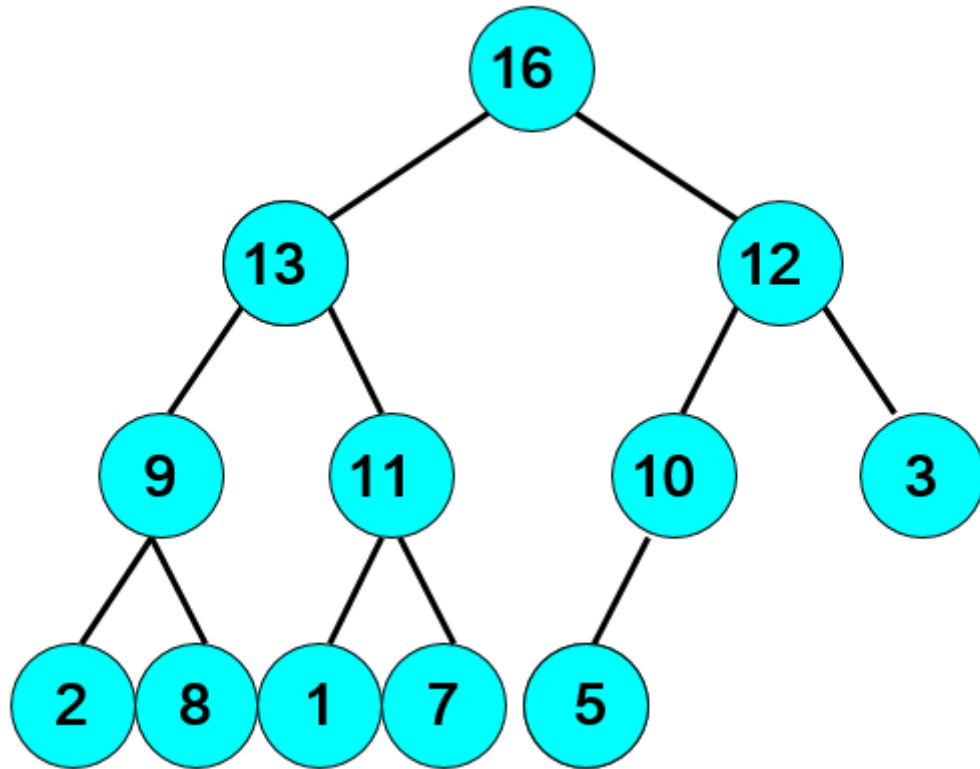
$O(\log(n))$

הורדת ערך מינימום/מקסימום בערימה

```
x = Pop():  
if end==0  
    error(empty)  
else  
    x=A[1];  
    A[1]=A[end];  
    end--;  
    Heapify(1);  
    return x;
```

```
Heapify(i): (שמירה על תכונת הסדר בתת עץ)  
l = Left(i);  
r = Right(i);  
if l≤heap-size and A[l] >A[i]  
    max = l;  
if r≤heap-size and A[r] >A[max]  
    max = r;  
if max ≠ i  
    switch( A[i],A[max])  
    Heapify(max);
```

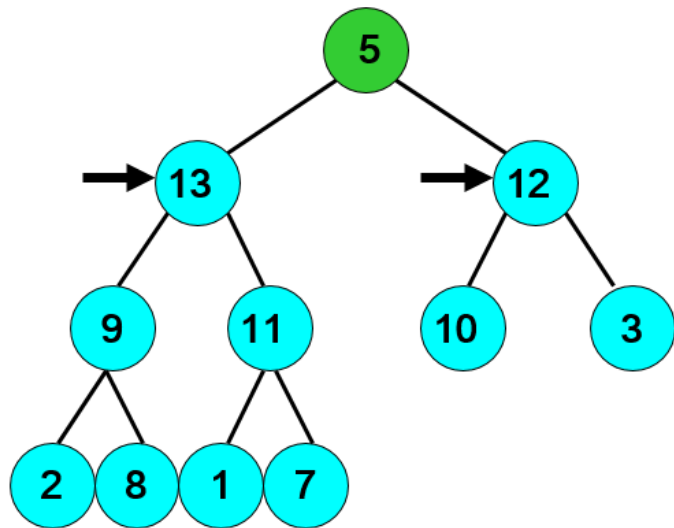
הורדת ערך מינימום/מקסימום בערימה דוגמה



ערימה התחלתית =>

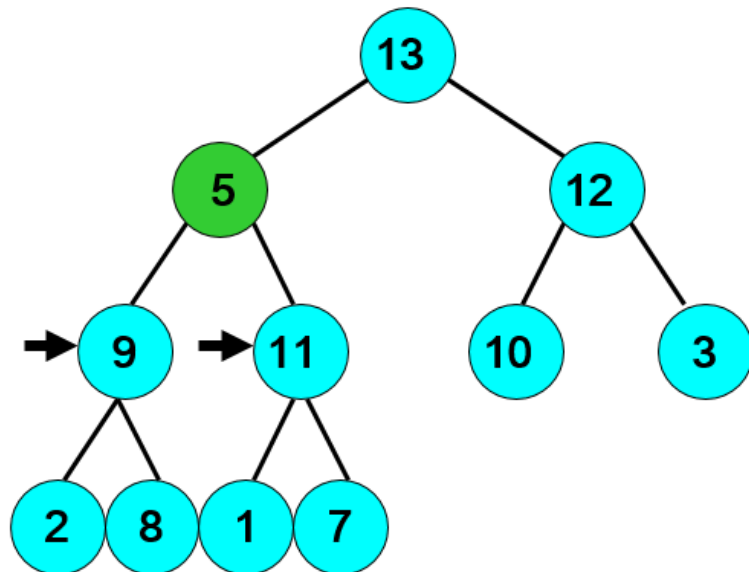
הורדת ערך מינימום/מקסימום בערימה דוגמה

Pop(Q)



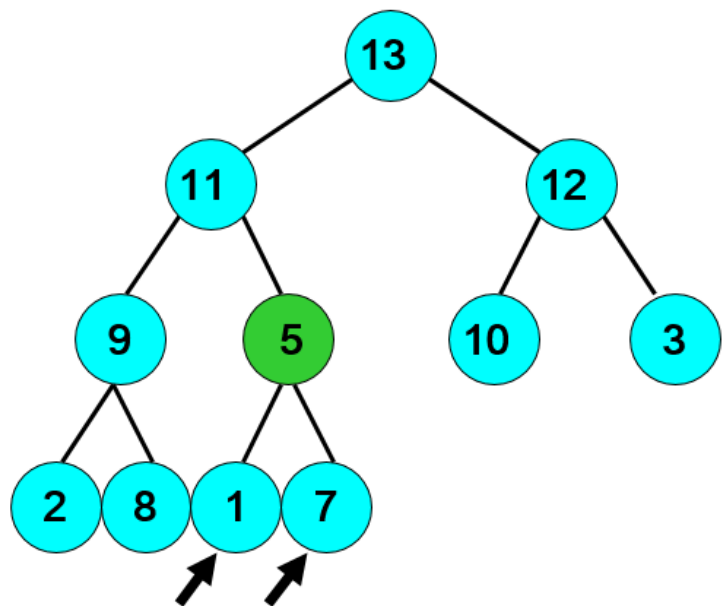
הורדת ערך מינימום/מקסימום בערימה דוגמה

Pop(Q)



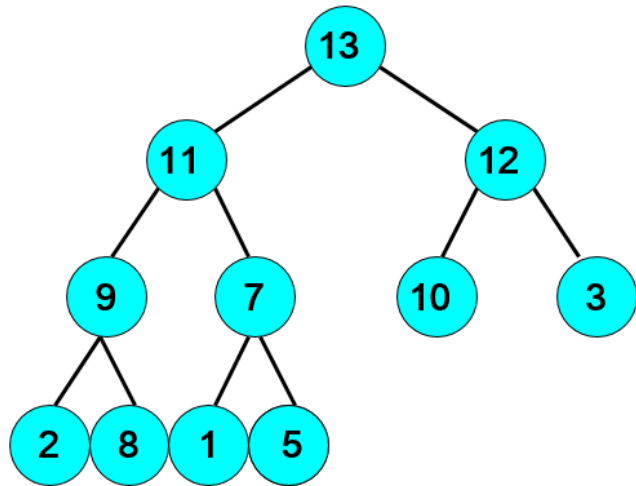
הורדת ערך מינימום/מקסימום בערימה דוגמה

Pop(Q)



הורדת ערך מינימום/מקסימום בערימה דוגמה

Pop(Q)



סבוכיות זמן

$O(\log(n))$

תרגיל 4

נתונות k רשימות ממוינות באורך m , מזגו את הרשימות לרשימה אחת ממוינת.

תרגיל 4 - פתרון

H- heap of pairs of values and indexes

L= array -size $m \cdot k$

$i_1 = i_2 = \dots = i_k = 0$

$s = 0$

For $r = 1$ to k

 H.push($L_r[i_r], i$)

While H.isEmpty == False:

$(a, j) = \text{H.pop}()$

$L[s] = a$

$s++$

$i_j ++$

 if $i_j < m$:

 H.push($L_j[i_j], j$)

תרגיל 4 - זמן ריצה

H- heap of pairs of values and indexes

L= array -size $m \cdot k$

$i_1 = i_2 = \dots = i_k = 0$

$s = 0$

For $r = 1$ to k

H.push($L_r[i_r], j$) $\longrightarrow O(k \log k)$

While H.isEmpty == False:

(a, j) = H.pop()

$L[s] = a$

$s++$

$i_j ++$

if $i_j < m$:

H.push($L_j[i_j], j$)

$\longrightarrow O(km \log k)$