מבני נתונים

מטרת הקורס:

1. הכרות עם מבני נתונים ומימושיהם היעילים
2. פיתוח כלים לניתוח יעילות
3. בחירת מבני נתונים לפתרון בעיות

מבני נתונים: מחסנית, תור, מילון, תור עדיפויות, טבלת ערבול...

שימושים: מיון, מימוש שפות תכנות, ארגון קבצים, ועוד ועוד.

ספר הלימוד העיקרי (קיים גם תרגום):

Cormen, Leiserson, Rivest, Introduction to Algorithms.

©Geiger&Itai,2001

תוכנית הקורס

1. מבני נתונים בסיסיים וסימונים אסימפטוטיים
2. מערכים ורשימות מקושרות
3. עצים ועצי חיפוש
4. עצי AVL
5. עצי 3-2
6. רשימות דילוגים
7. טבלאות ערבול
8. אחזקת קבוצות זרות
9. מיון
10. מיון
11. טיפול במחרוזות
12. גרפים
13. איסוף אשפה

©cs,Technion

חומר קריאה לשיעור זה

Chapter 2 – Growth of functions (23 - 41)

Chapter 4 – Recurrences (53 – 60)

Chapter 11.1 – Stacks and Queues (200 – 204)

הגדרות בסיסיות

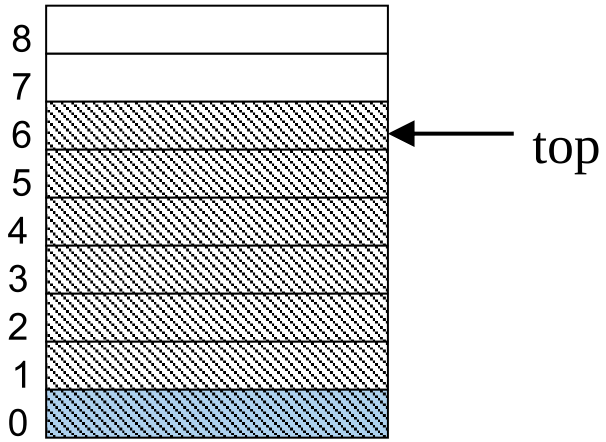
מבנה נתונים הוא אוסף של פעולות על קבוצת נתונים.

מימוש של מבנה נתונים הוא אוסף פרוצדורות, אחת לכל פעולה, המממשות את הפעולות של מבנה הנתונים.

מהי מחסנית?

האם היא מערך עם מציין ל- top ?

אתחול:	top = 0
הוספת איבר:	top = top + 1; A[top] = x;
ראש המחסנית:	A[top]
הוצאת איבר:	top = top - 1



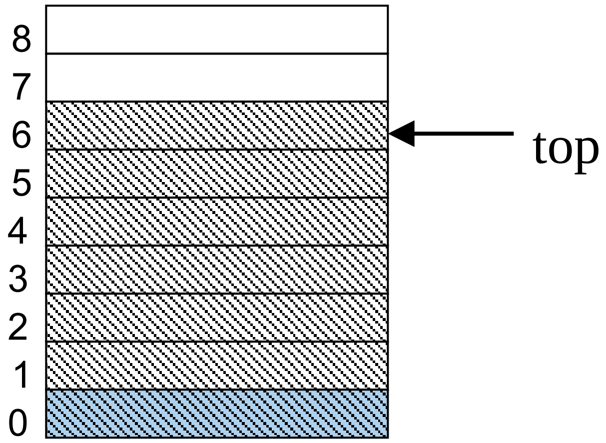
הגדרות בסיסיות

<u>מבנה נתונים</u> הוא אוסף של פעולות על קבוצת נתונים.

<u>מימוש של מבנה נתונים</u> הוא אוסף פרוצדורות, אחת לכל פעולה, הממממשות את הפעולות של מבנה הנתונים.

<u>מהי מחסנית?</u>

האם היא מערך עם מציין ל- top ?



אתחול: top = 0

הוספת איבר: top = top + 1; A[top] = x;

ראש המחסנית: A[top]

הוצאת איבר: top = top - 1

לא ! זהו מימוש של מחסנית באמצעות מערך ומצביע.

מחסנית כמבנה נתונים

אחרון נכנס - ראשון יוצא   Last In -- First Out : LIFO

<u>מחסנית מוגדרת ע"י הפעולות הבאות</u>:

create(S) - מחזיר מחסנית S ריקה חדשה.

push(S,x) - מכניס איבר בעל ערך x למחסנית S.

top(S) - מחזיר את האיבר שבראש המחסנית S (המחסנית אינה משתנה).

pop(S) - מוציא את האיבר שבראש המחסנית S.

is-empty(S) - מחזיר true אם המחסנית S ריקה ו-false אחרת.

©cs,Technion

מחסנית כמבנה נתונים (המשך)

פעולות המחסנית מקיימות את הכללים הבאים:

אפשר לבצע top ,pop רק על מחסנית לא ריקה.

מיד לאחר (create(S, ‏(is-empty(S מחזיר ערך true.

לאחר ביצוע push, ואחריו pop, המחסנית לא משתנה.

כל מימוש חייב לאפשר את הפעולות ולקיים את הכללים.

יש טענות הנובעות רק מהכללים, בלי תלות במימוש.

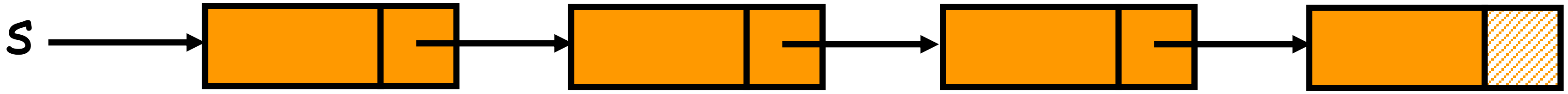
לדוגמא:

```
create(S);
push(S,17);
pop(S);
print is_empty(S);
```

מה יודפס ?

מימוש מחסנית

1. מימוש בעזרת מערך (כפי שהוסבר).
2. מימוש בעזרת רשימה מקושרת.



הגדרת צומת:

```
#define NULL 0
typedef struct node {
        DATA_TYPE info;
        struct node *next;
}  NODE;
typedef   NODE   *STACK;
```

יצירת מחסנית ריקה:

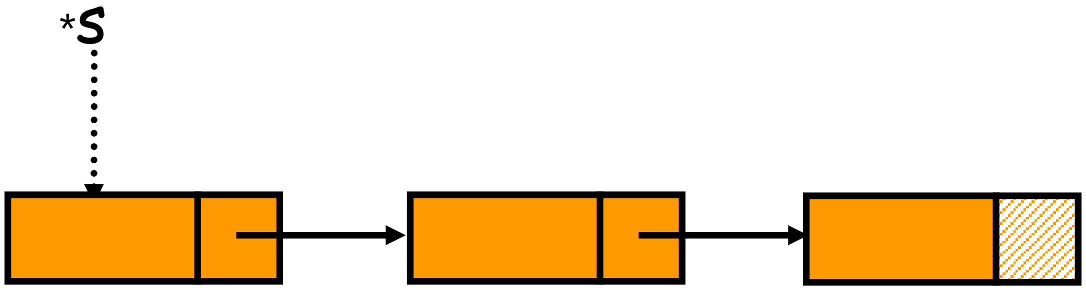
```
void create (STACK  *s) {
        (*s) = NULL ;
}
```

©cs,Technion

הכנסת איבר

פעולת push(s,x):

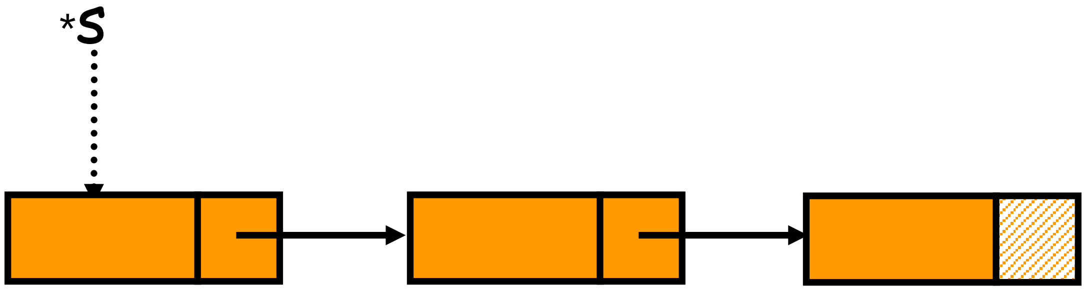
```
void  push ( STACK  *s,  DATA_TYPE x){
            NODE *P;
      P = malloc (sizeof (NODE));
      P -> info = x;
      P -> next = (*s) ;
      (*s) = p ;
}
```



הכנסת איבר

פעולת push(s,x):

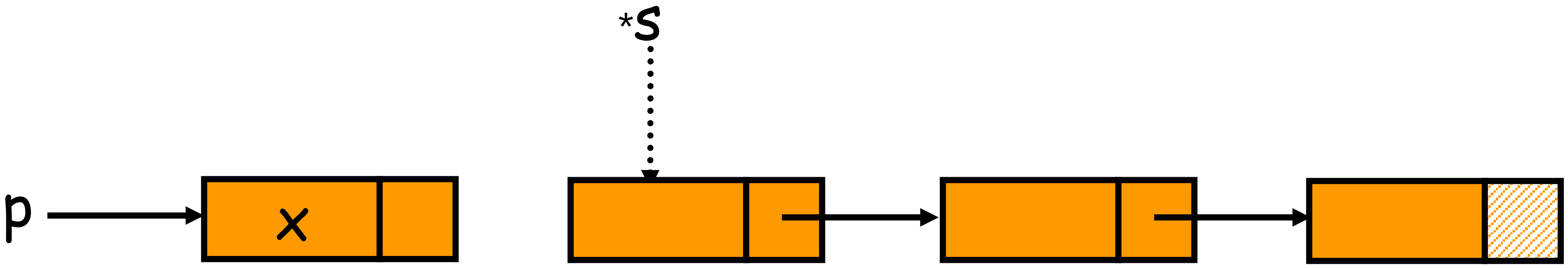
```
void  push ( STACK  *s,  DATA_TYPE x){
            NODE *P;
      P = malloc (sizeof (NODE));
      P -> info = x;
      P -> next = (*s) ;
      (*s) = p ;
}
```



הכנסת איבר

פעולת push(s,x):

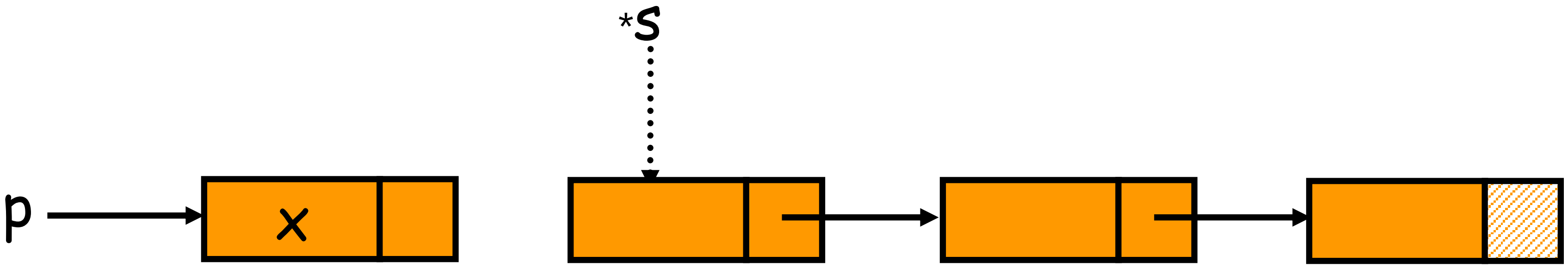
```
void  push ( STACK  *s,  DATA_TYPE x){
            NODE *P;
      P = malloc (sizeof (NODE));
      P -> info = x;
      P -> next = (*s) ;
      (*s) = p ;
}
```



הכנסת איבר

```
void  push ( STACK  *s,  DATA_TYPE x){
            NODE *P;
      P = malloc (sizeof (NODE));
      P -> info = x;
      P -> next = (*s) ;
      (*s) = p ;
}
```

פעולת push(s,x):

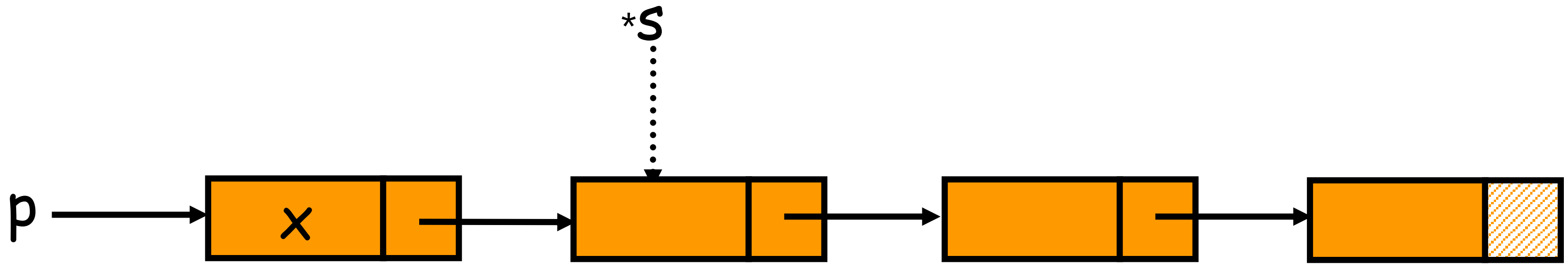


©cs,Technion

הכנסת איבר

פעולת push(s,x):

```
void  push ( STACK  *s,  DATA_TYPE x){
            NODE *P;
      P = malloc (sizeof (NODE));
      P -> info = x;
      P -> next = (*s) ;
      (*s) = p ;
}
```

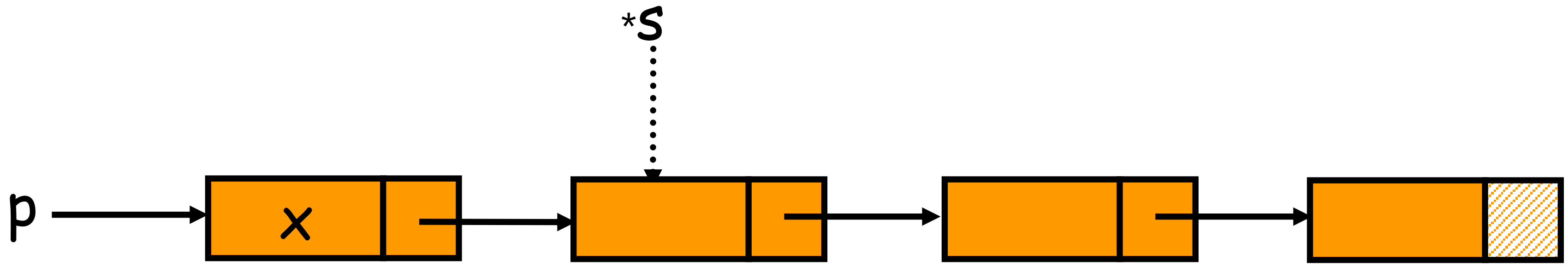


©cs,Technion

הכנסת איבר

פעולת push(s,x):

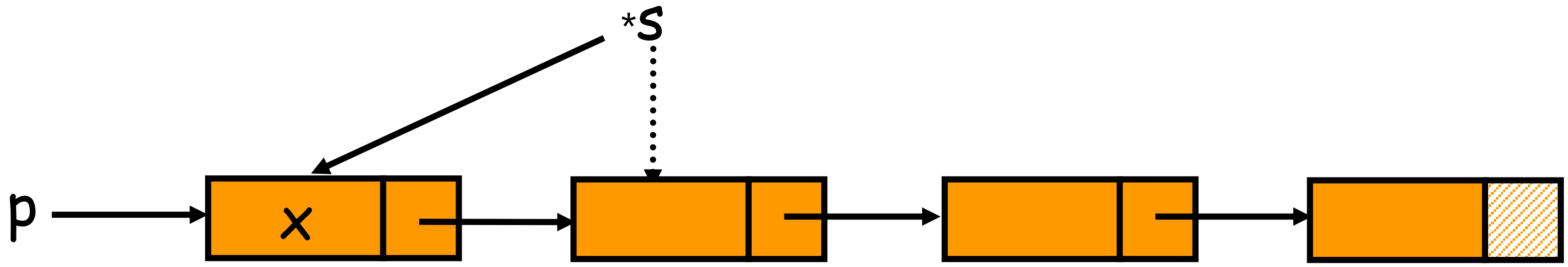
```
void  push ( STACK  *s,  DATA_TYPE x){
            NODE *P;
      P = malloc (sizeof (NODE));
      P -> info = x;
      P -> next = (*s) ;
      (*s) = p ;
}
```



הכנסת איבר

פעולת push(s,x):

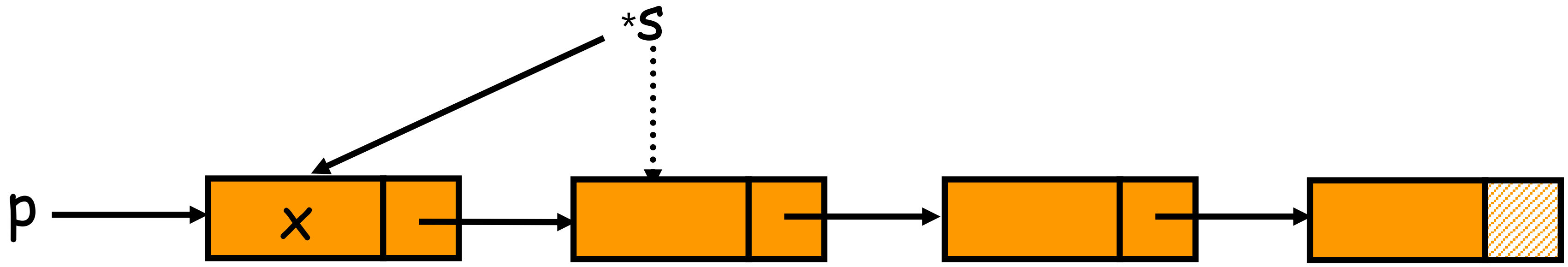
```
void  push ( STACK  *s,  DATA_TYPE x){
            NODE *P;
      P = malloc (sizeof (NODE));
      P -> info = x;
      P -> next = (*s) ;
      (*s) = p ;
}
```



הכנסת איבר

פעולת push(s,x):

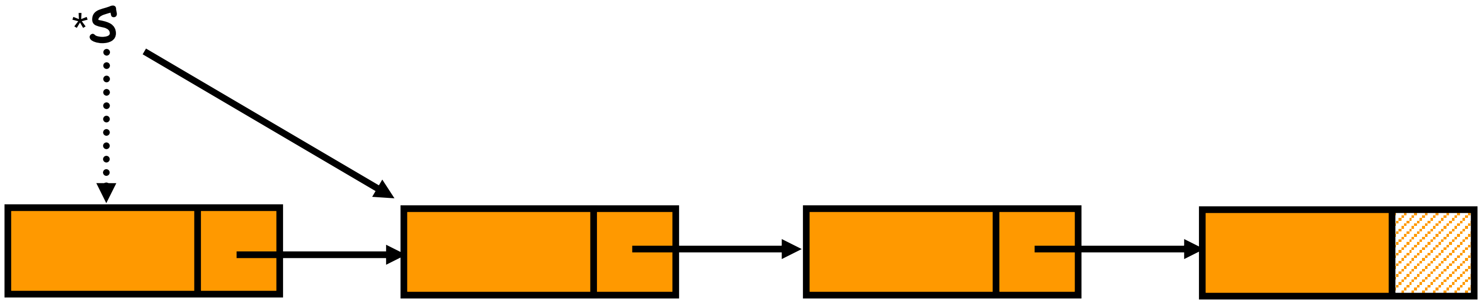
```
void  push ( STACK  *s,  DATA_TYPE x){
            NODE *P;
      P = malloc (sizeof (NODE));
      P -> info = x;
      P -> next = (*s) ;
      (*s) = p ;
}
```



אחזור והוצאת איבר

פעולת pop(s):

```
void pop ( STACK *s){
        STACK t ;
        t = (*s) → next;
        free (*s) ;
        (*s) = t ;
}
```



פעולת top(s):

```
DATA_TYPE top ( STACK *s){
return (*s) → info ;
}
```

©cs,Technion

תור כמבנה נתונים

ראשון נכנס - ראשון יוצא First In -- First Out : FIFO

<u>תור מוגדר ע"י הפעולות הבאות</u>:

create(Q) - מחזיר תור ריק.

head(Q)- מחזיר את ערך האיבר שבראש התור Q (התור אינו משתנה).

enqueue(Q,x) - מכניס איבר עם ערך x לסוף התור Q.

dequeue(Q) - מוציא את האיבר שבראש התור Q.

is-empty(Q) - מחזיר true אם התור Q ריק ו-false אחרת.

©cs,Technion

מימוש של תור בעזרת מערך

head(Q): Q[f];

enqueue(Q,x): Q[r] = x;

r = (r+1) %n;

dequeue(Q) f = (f+1) % n;

is_empty(Q): f = = r;

create(Q): f = r = 0;

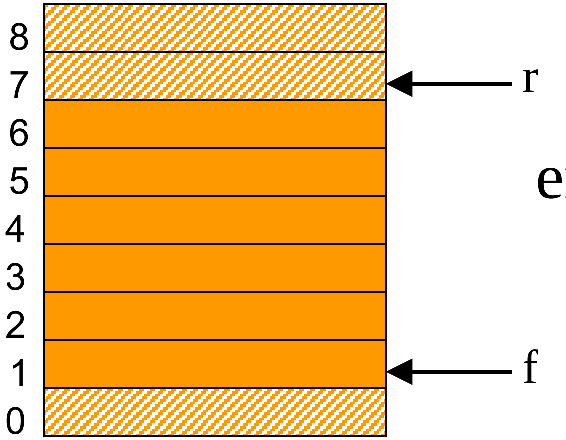
מערך Q בן n איברים עם שני מציינים

r = מציין את מקום האיבר שאחרי סוף התור

f = מציין את מקום האיבר שבראש התור

הפעולות האריתמטיות נעשות mod n.

המימוש מלא אם f = = (r +1)% n .



יש לבדוק "לא מלא" לפני enqueue

ו"לא ריק" לפני dequeue.

אנימציה

©cs,Technion

מימוש של תור בעזרת מערך

מערך Q בן n איברים עם שני מציינים

r = מציין את מקום האיבר שאחרי סוף התור

f = מציין את מקום האיבר שבראש התור

head(Q): Q[f];

enqueue(Q,x): Q[r] = x;
r = (r+1) %n;

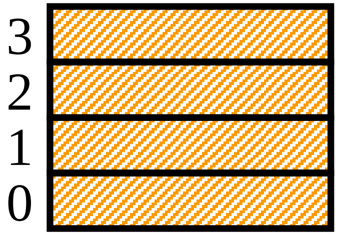
dequeue(Q) f = (f+1) % n;

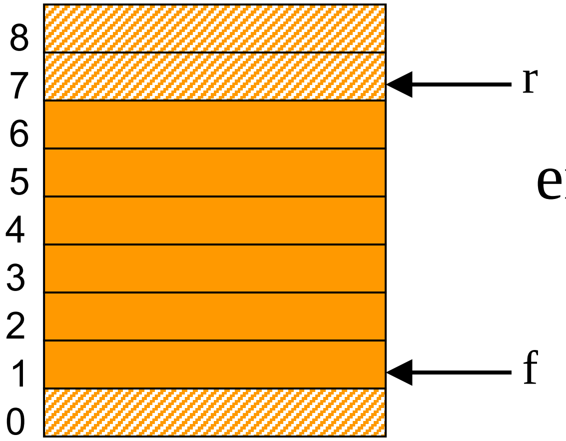
is_empty(Q): f = = r;

create(Q): f = r = 0;

הפעולות האריתמטיות נעשות mod n.

המימוש מלא אם f = = (r +1)% n .





יש לבדוק "לא מלא" לפני enqueue

ו"לא ריק" לפני dequeue.

©cs,Technion

מימוש של תור בעזרת מערך

<u>מערך Q בן n איברים עם שני מציינים</u>

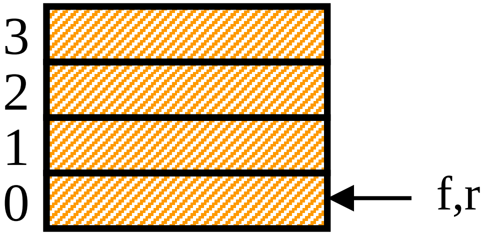
r = מציין את מקום האיבר שאחרי סוף התור

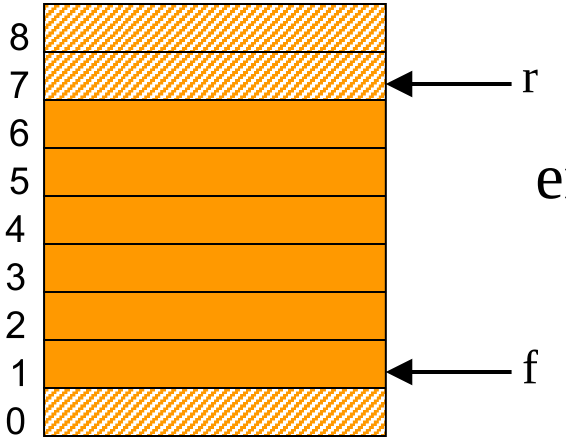
f = מציין את מקום האיבר שבראש התור

```
head(Q):        Q[f];
enqueue(Q,x):   Q[r] = x;
                r = (r+1) %n;
dequeue(Q)      f = (f+1) % n;
is_empty(Q):    f = = r;
create(Q):      f = r = 0;
```

הפעולות האריתמטיות נעשות mod n.

המימוש מלא אם f = = (r +1)% n .





יש לבדוק "לא מלא" לפני enqueue

ו"לא ריק" לפני dequeue.

מימוש של תור בעזרת מערך

מערך Q בן n איברים עם שני מציינים

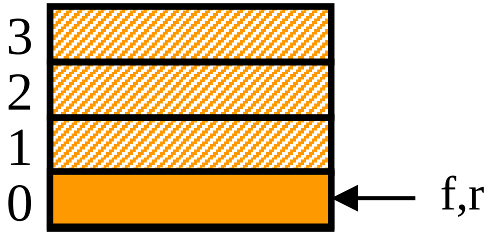
r = מציין את מקום האיבר שאחרי סוף התור

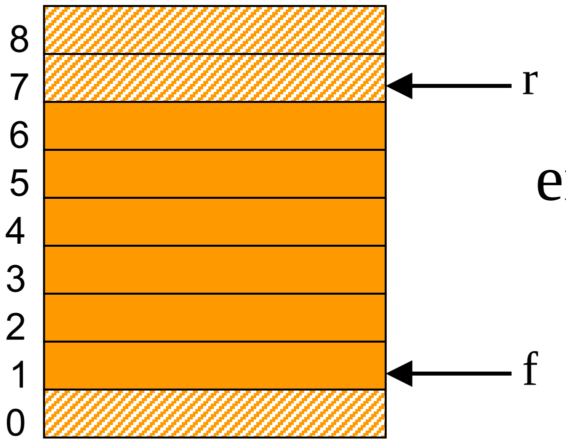
f = מציין את מקום האיבר שבראש התור

```
head(Q):        Q[f];
enqueue(Q,x):   Q[r] = x;
                r = (r+1) %n;
dequeue(Q)      f = (f+1) % n;
is_empty(Q):    f = = r;
create(Q):      f = r = 0;
```

הפעולות האריתמטיות נעשות mod n.

המימוש מלא אם f = = (r +1)% n .





יש לבדוק "לא מלא" לפני enqueue

ו"לא ריק" לפני dequeue.

©cs,Technion

מימוש של תור בעזרת מערך

מערך Q בן n איברים עם שני מציינים

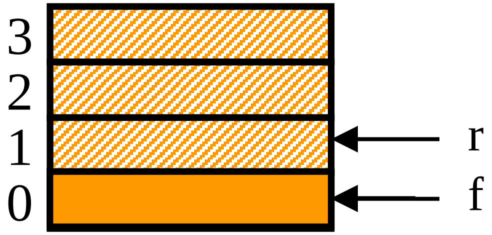
r = מציין את מקום האיבר שאחרי סוף התור

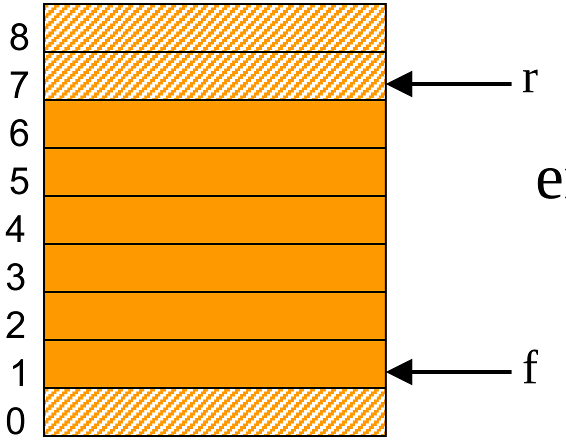
f = מציין את מקום האיבר שבראש התור

הפעולות האריתמטיות נעשות mod n.

המימוש מלא אם f = = (r +1)% n .

```
head(Q):        Q[f];
enqueue(Q,x):   Q[r] = x;
                r = (r+1) %n;
dequeue(Q)      f = (f+1) % n;
is_empty(Q):    f = = r;
create(Q):      f = r = 0;
```





יש לבדוק "לא מלא" לפני enqueue

ו"לא ריק" לפני dequeue.

מימוש של תור בעזרת מערך

מערך Q בן n איברים עם שני מציינים

r = מציין את מקום האיבר שאחרי סוף התור

f = מציין את מקום האיבר שבראש התור

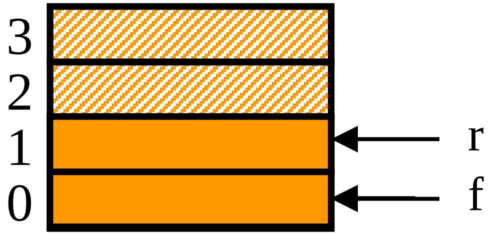
הפעולות האריתמטיות נעשות mod n.

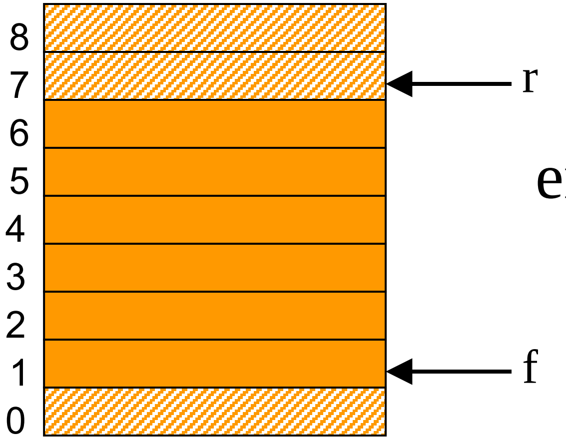
המימוש מלא אם f = = (r +1)% n .

יש לבדוק "לא מלא" לפני enqueue

ו"לא ריק" לפני dequeue.

```
head(Q):          Q[f];
enqueue(Q,x):     Q[r] = x;
                  r = (r+1) %n;
dequeue(Q)        f = (f+1) % n;
is_empty(Q):      f = = r;
create(Q):        f = r = 0;
```





מימוש של תור בעזרת מערך

מערך Q בן n איברים עם שני מציינים

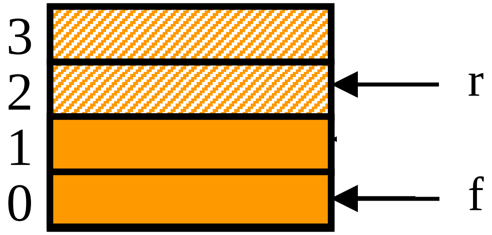
r = מציין את מקום האיבר שאחרי סוף התור

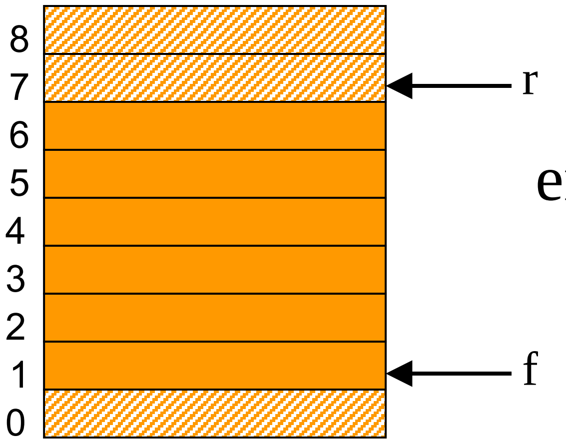
f = מציין את מקום האיבר שבראש התור

```
head(Q):          Q[f];
enqueue(Q,x):     Q[r] = x;
                  r = (r+1) %n;
dequeue(Q)        f = (f+1) % n;
is_empty(Q):      f = = r;
create(Q):        f = r = 0;
```

הפעולות האריתמטיות נעשות mod n.

המימוש מלא אם f = = (r +1)% n .





יש לבדוק "לא מלא" לפני enqueue

ו"לא ריק" לפני dequeue.

©cs,Technion

מימוש של תור בעזרת מערך

head(Q): Q[f];

enqueue(Q,x): Q[r] = x;

r = (r+1) %n;

dequeue(Q) f = (f+1) % n;

is_empty(Q): f = = r;

create(Q): f = r = 0;

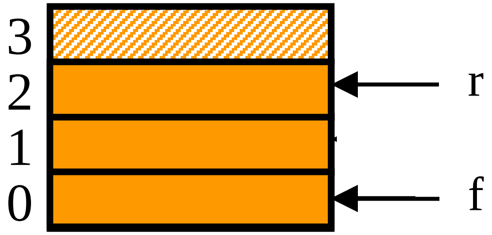
מערך Q בן n איברים עם שני מציינים

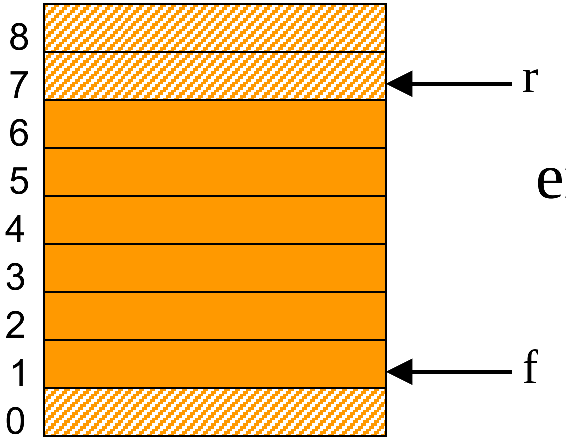
r = מציין את מקום האיבר שאחרי סוף התור

f = מציין את מקום האיבר שבראש התור

הפעולות האריתמטיות נעשות mod n.

המימוש מלא אם f = = (r +1)% n .





יש לבדוק "לא מלא" לפני enqueue

ו"לא ריק" לפני dequeue.

©cs,Technion

מימוש של תור בעזרת מערך

מערך Q בן n איברים עם שני מציינים

r = מציין את מקום האיבר שאחרי סוף התור

f = מציין את מקום האיבר שבראש התור

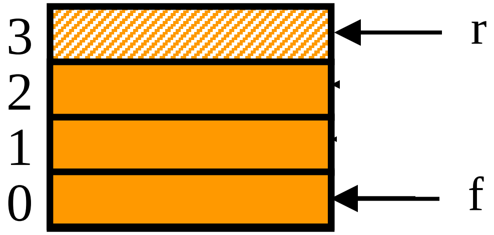
הפעולות האריתמטיות נעשות mod n.

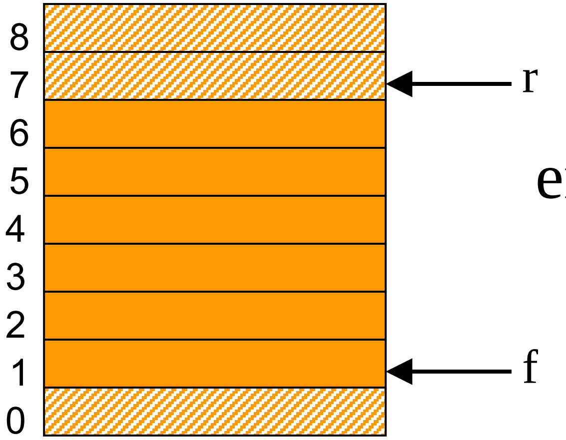
המימוש מלא אם f = = (r +1)% n .

יש לבדוק "לא מלא" לפני enqueue

ו"לא ריק" לפני dequeue.

```
head(Q):          Q[f];
enqueue(Q,x):     Q[r] = x;
                  r = (r+1) %n;
dequeue(Q)        f = (f+1) % n;
is_empty(Q):      f = = r;
create(Q):        f = r = 0;
```





מימוש של תור בעזרת מערך

<u>מערך Q בן n איברים עם שני מציינים</u>

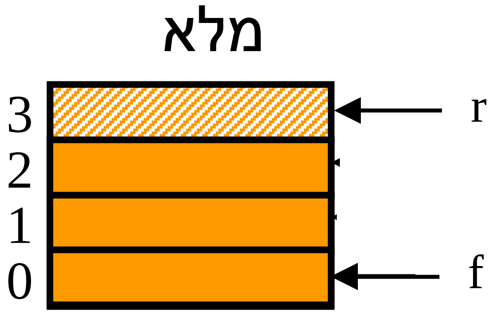
r = מציין את מקום האיבר שאחרי סוף התור

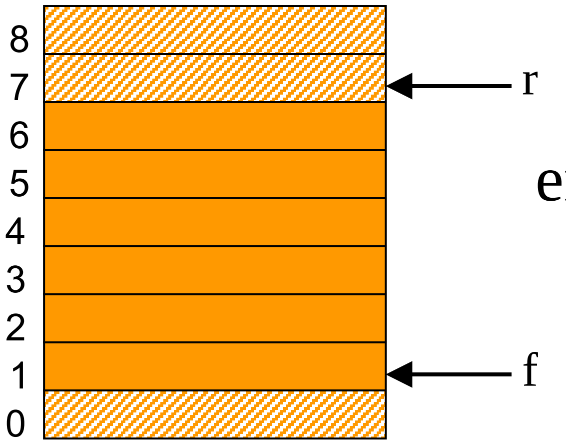
f = מציין את מקום האיבר שבראש התור

הפעולות האריתמטיות נעשות mod n.

המימוש מלא אם f = = (r +1)% n .

```
head(Q):        Q[f];
enqueue(Q,x):   Q[r] = x;
                r = (r+1) %n;
dequeue(Q)      f = (f+1) % n;
is_empty(Q):    f = = r;
create(Q):      f = r = 0;
```





יש לבדוק "לא מלא" לפני enqueue

ו"לא ריק" לפני dequeue.

©cs,Technion

מימוש של תור בעזרת מערך

מערך Q בן n איברים עם שני מציינים

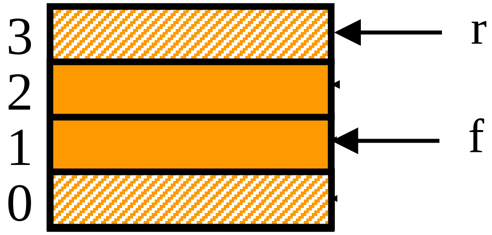
r = מציין את מקום האיבר שאחרי סוף התור

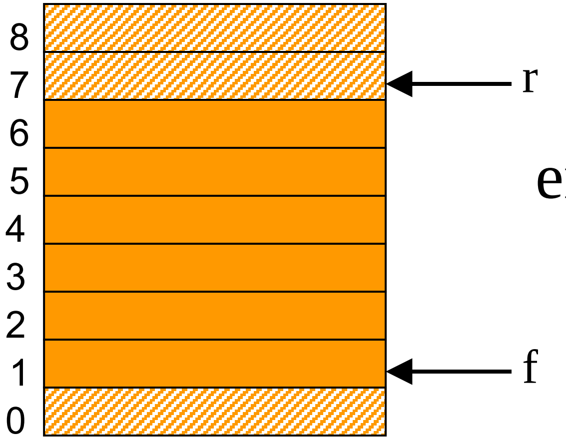
f = מציין את מקום האיבר שבראש התור

```
head(Q):        Q[f];
enqueue(Q,x):   Q[r] = x;
                r = (r+1) %n;
dequeue(Q)      f = (f+1) % n;
is_empty(Q):    f = = r;
create(Q):      f = r = 0;
```

הפעולות האריתמטיות נעשות mod n.

המימוש מלא אם f = = (r +1)% n .





יש לבדוק "לא מלא" לפני enqueue

ו"לא ריק" לפני dequeue.

מימוש של תור בעזרת מערך

<u>מערך Q בן n איברים עם שני מציינים</u>

r = מציין את מקום האיבר שאחרי סוף התור

f = מציין את מקום האיבר שבראש התור

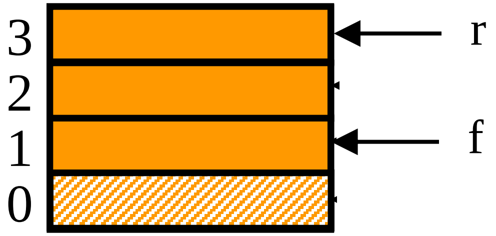
הפעולות האריתמטיות נעשות mod n.

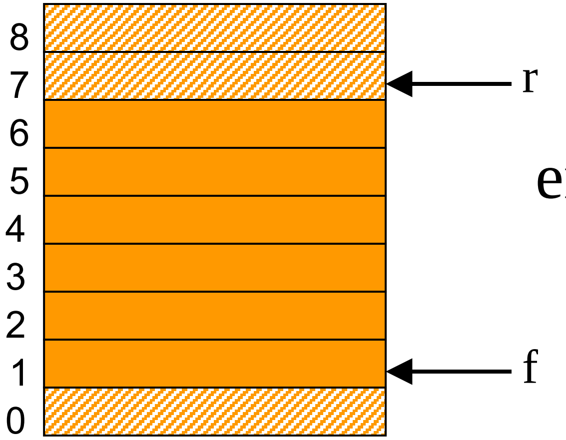
המימוש מלא אם f = = (r +1)% n .

יש לבדוק "לא מלא" לפני enqueue

ו"לא ריק" לפני dequeue.

```
head(Q):        Q[f];
enqueue(Q,x):   Q[r] = x;
                r = (r+1) %n;
dequeue(Q)      f = (f+1) % n;
is_empty(Q):    f = = r;
create(Q):      f = r = 0;
```





מימוש של תור בעזרת מערך

<u>מערך Q בן n איברים עם שני מציינים</u>

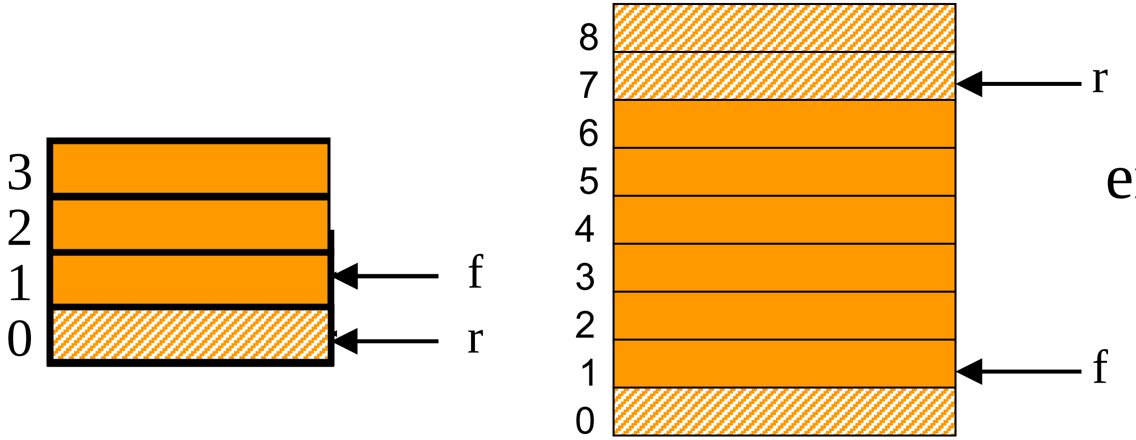
r = מציין את מקום האיבר שאחרי סוף התור

f = מציין את מקום האיבר שבראש התור

```
head(Q):        Q[f];
enqueue(Q,x):   Q[r] = x;
                r = (r+1) %n;
dequeue(Q)      f = (f+1) % n;
is_empty(Q):    f = = r;
create(Q):      f = r = 0;
```

הפעולות האריתמטיות נעשות mod n.

המימוש מלא אם f = = (r +1)% n .



יש לבדוק "לא מלא" לפני enqueue

ו"לא ריק" לפני dequeue.

©cs,Technion

מימוש של תור בעזרת מערך

head(Q): Q[f];

enqueue(Q,x): Q[r] = x;

r = (r+1) %n;

dequeue(Q) f = (f+1) % n;

is_empty(Q): f = = r;

create(Q): f = r = 0;

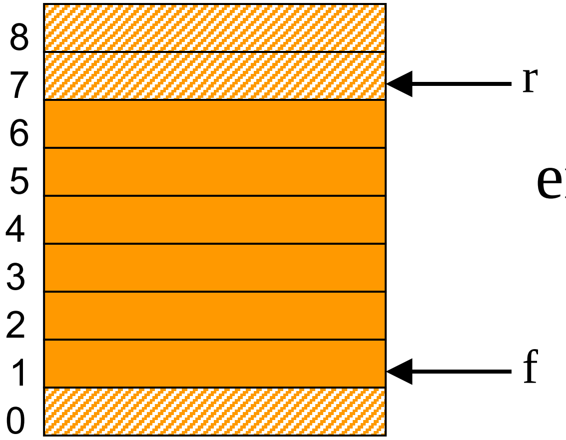
מערך Q בן n איברים עם שני מציינים

r = מציין את מקום האיבר שאחרי סוף התור

f = מציין את מקום האיבר שבראש התור

הפעולות האריתמטיות נעשות mod n.

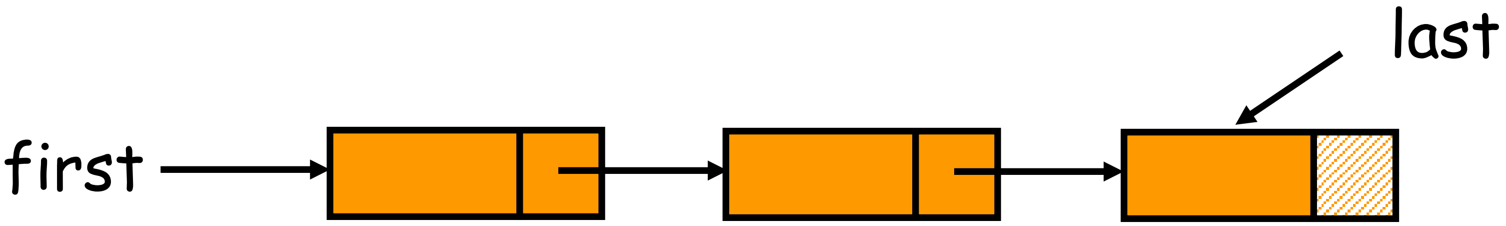
המימוש מלא אם f = = (r +1)% n .



יש לבדוק "לא מלא" לפני enqueue

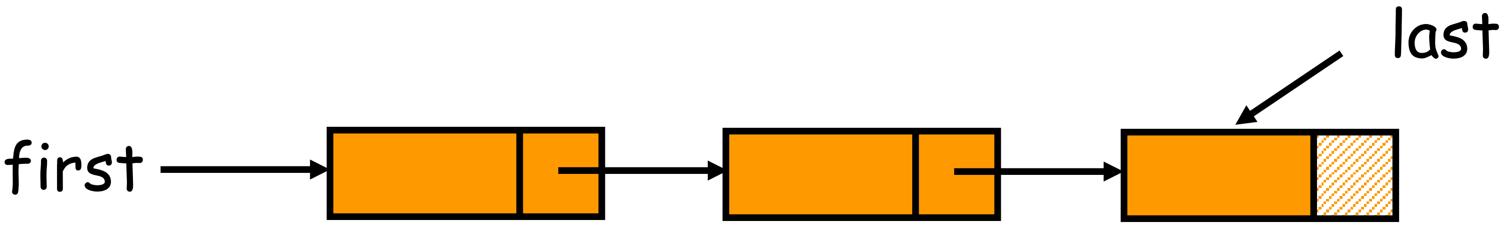
ו"לא ריק" לפני dequeue.

מימוש תור ע"י רשימה מקושרת

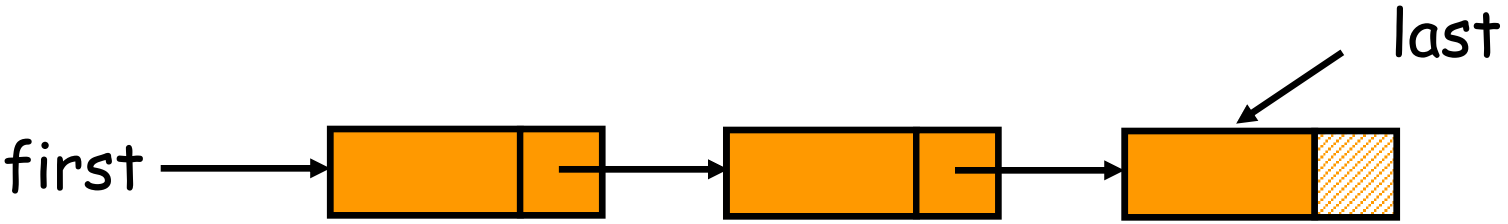


הכנסה בסוף התור (last) :

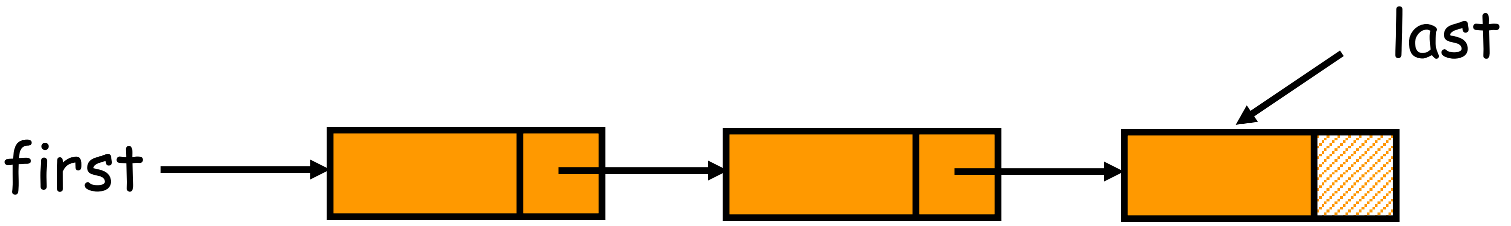
מימוש תור ע"י רשימה מקושרת



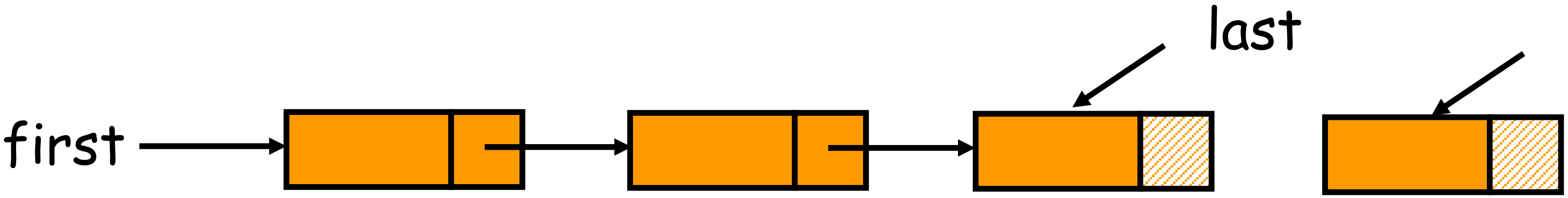
הכנסה בסוף התור (last) :



מימוש תור ע"י רשימה מקושרת

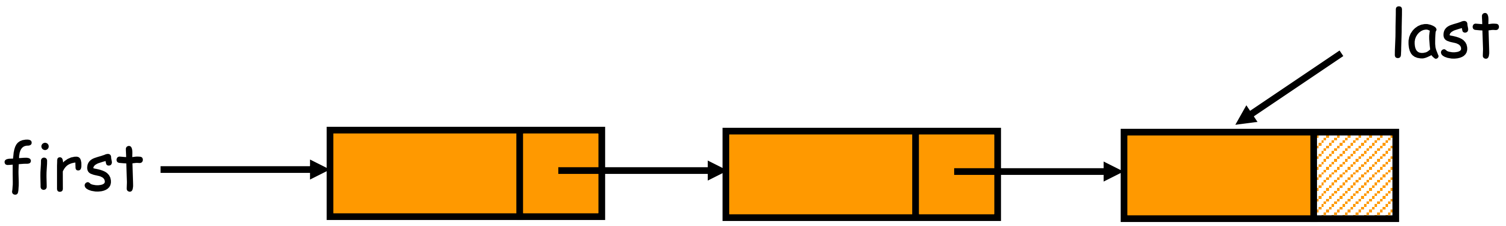


הכנסה בסוף התור (last) :

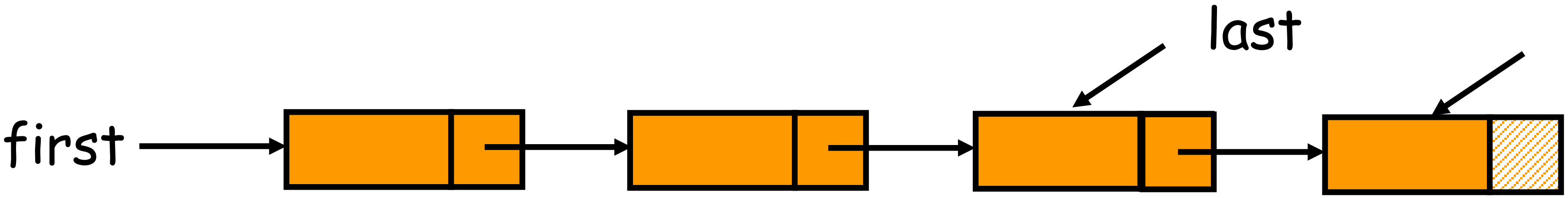


©cs,Technion

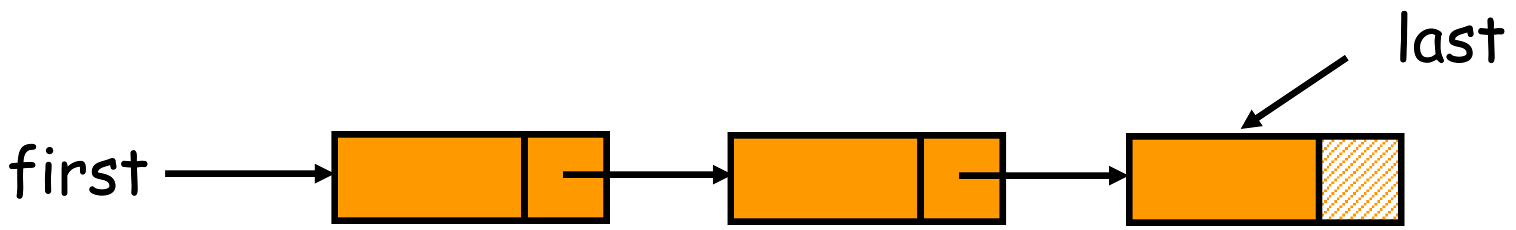
מימוש תור ע"י רשימה מקושרת



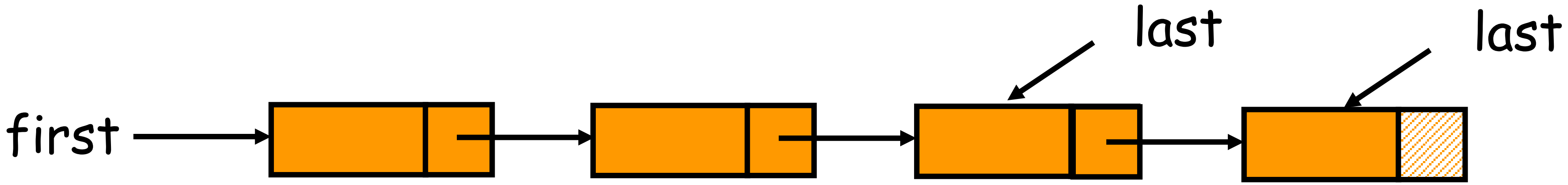
הכנסה בסוף התור (last) :



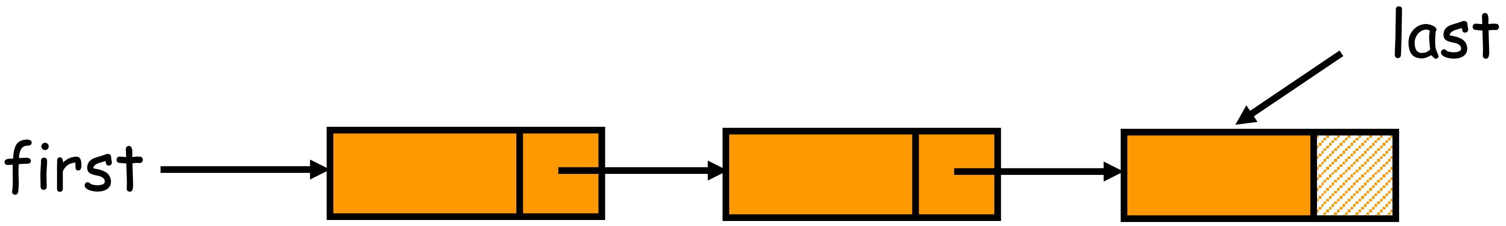
מימוש תור ע"י רשימה מקושרת



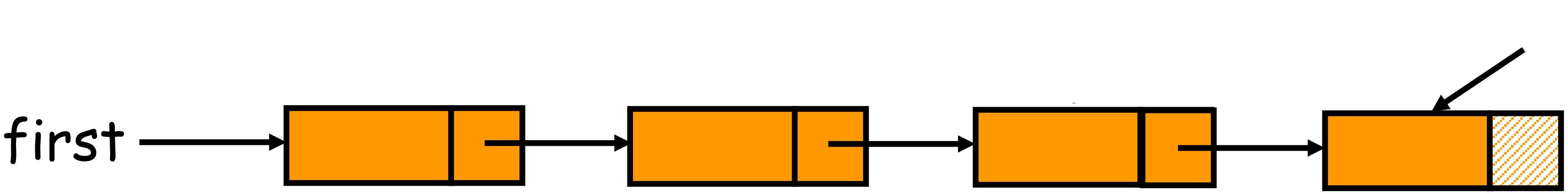
הכנסה בסוף התור (last) :



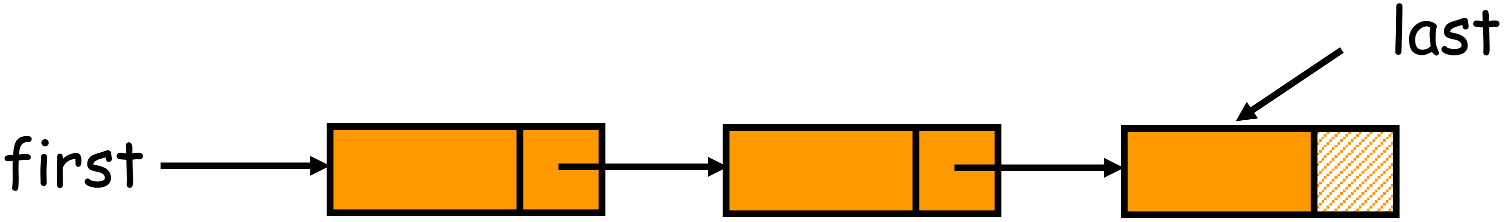
מימוש תור ע"י רשימה מקושרת



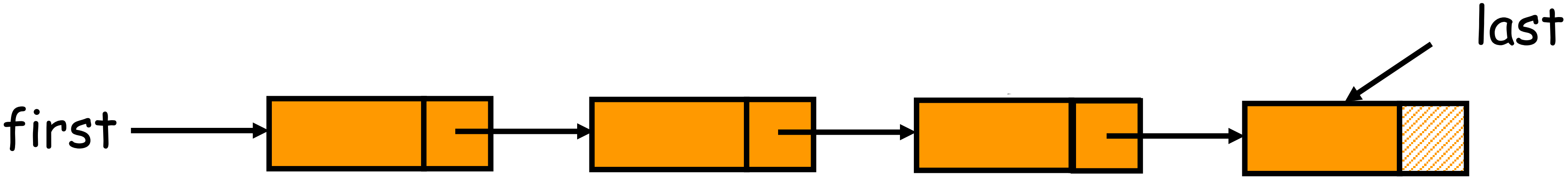
הכנסה בסוף התור (last) :



מימוש תור ע"י רשימה מקושרת



הכנסה בסוף התור (last) :



הוצאה מראש התור (first).

שימו לב שלא ניתן להוציא איבר באמצעות המצביע last.

©cs,Technion

הערות לגבי מבני נתונים

המשתמש במבנה:

מכיר את הפעולות והשפעתן על הנתונים.

אינו נדרש להכיר את פרטי המימוש.

בזמן פיתוח: ניתן להחליף את מימוש מבני הנתונים מבלי לפגוע בשימושים.
מאפשר לתכנת מימושים פשוטים ואחר כך להחליפם.

איכות המימוש נקבעת ע"י:

- ניתוח יעילות:
  - זמן – מספר צעדי החישוב הנדרשים, לכל פעולה.
  - מקום – כמות הזיכרון הנדרשת.
- פשטות התכנות (מאפשר אחזקה יעילה).

זמן ריצה של אלגוריתם

זמן ריצה של אלגוריתם A עבור קלט x יוסומן ב- $t_A(x)$. זמן הריצה נמדד ע"י מספר פקודות מכונה שהאלגוריתם מבצע על קלט נתון. מדד זה מתעלם מהבדלי המהירות בין הפקודות. (למשל, חבור לעומת כפל).

הגודל של קלט x יסומן ב- $|x|$. לדוגמא, בתוכנית המסכמת איברי מערך x, גודל הקלט הוא מספר איברי המערך.

זמן הריצה הגרוע ביותר (worst case) של אלגוריתם A עבור קלט שגודלו n מוגדר ע"י $t_A(n) = \max\{t_A(x) \mid |x| = n\}$.

זמן ריצה של אלגוריתם

זמן ריצה של אלגוריתם A עבור קלט x יוסומן ב- $t_A(x)$. זמן הריצה נמדד ע"י מספר פקודות מכונה שהאלגוריתם מבצע על קלט נתון. מדד זה מתעלם מהבדלי המהירות בין הפקודות. (למשל, חבור לעומת כפל).

הגודל של קלט x יסומן ב- $|x|$. לדוגמא, בתוכנית המסכמת איברי מערך x, גודל הקלט הוא מספר איברי המערך.

זמן הריצה הגרוע ביותר (worst case) של אלגוריתם A עבור קלט שגודלו n מוגדר ע"י $t_A(n)= \max \{t_A(x) \mid |x|=n\}$.

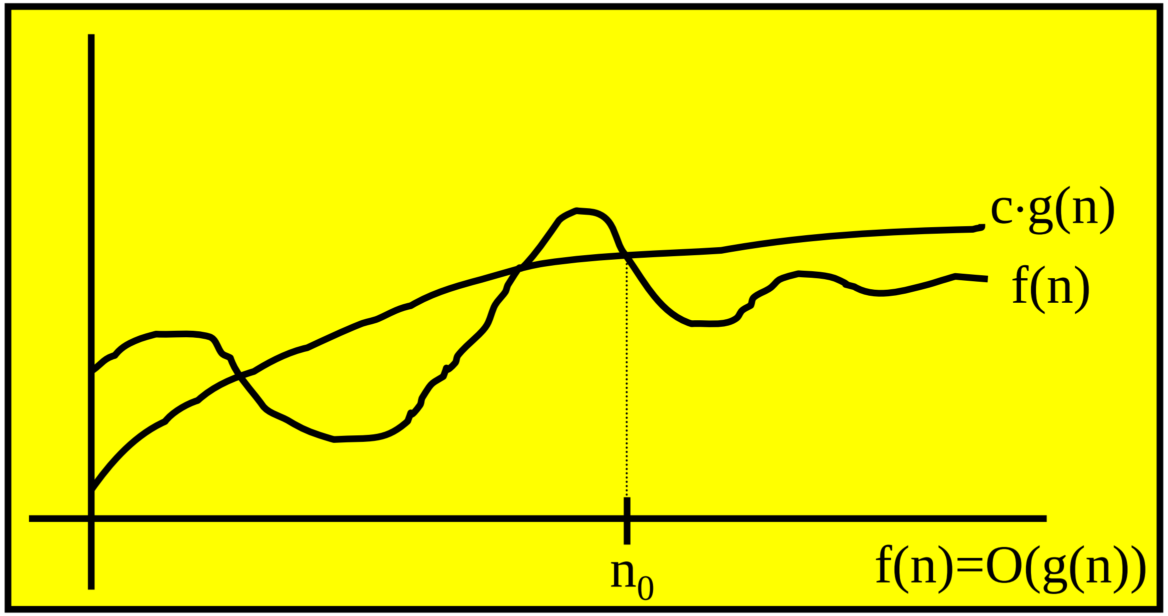
דוגמא:

```
sum = 0
for (i = 0; i < n; i++)
        sum = sum + a[i];
```

זמן הריצה הגרוע ביותר של אלגוריתם זה מקיים: $n + 1 \le t_A(n) \le c_1 \cdot n + c_2$ כאשר c_1,c_2 הם קבועים התלויים במימוש הפקודות בשפת מכונה.

©cs,Technion

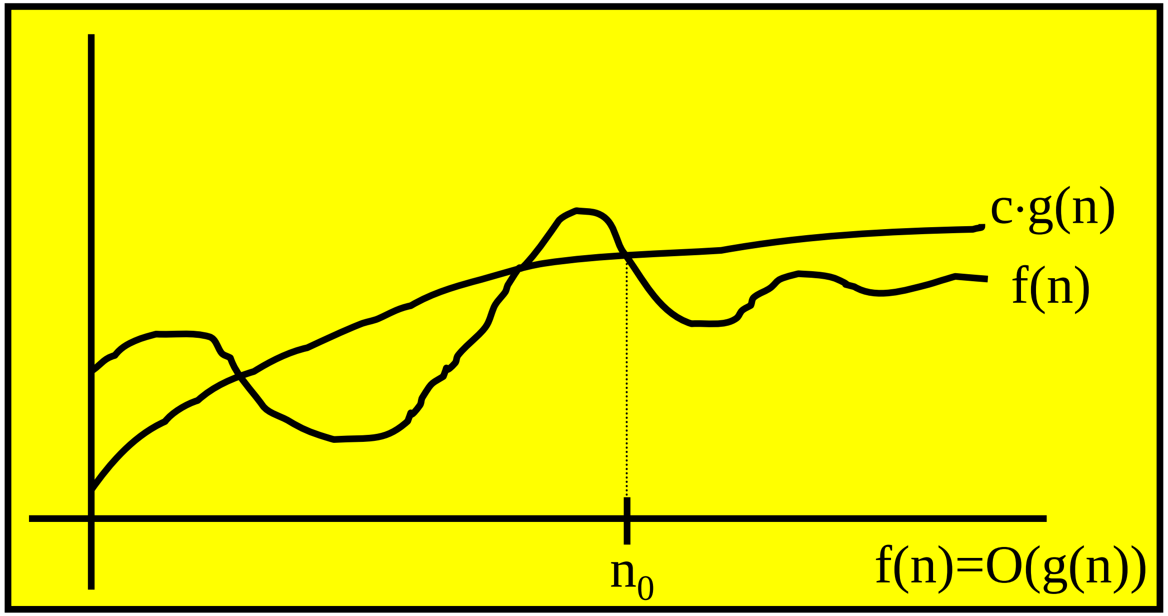
סיבוכיות והסימון O



סיבוכיות והסימון O

<u>הגדרה</u>: יהיו $f(n), g(n)$ פונקציות חיוביות. נאמר שהפונקציה $f(n)$ נמצאת בקבוצת הפונקציות $O(g(n))$ אם קיימים קבועים n_0,c כך שלכל $n_0 \le n$ מתקיים :

$$f(n) \le c \cdot g(n)$$

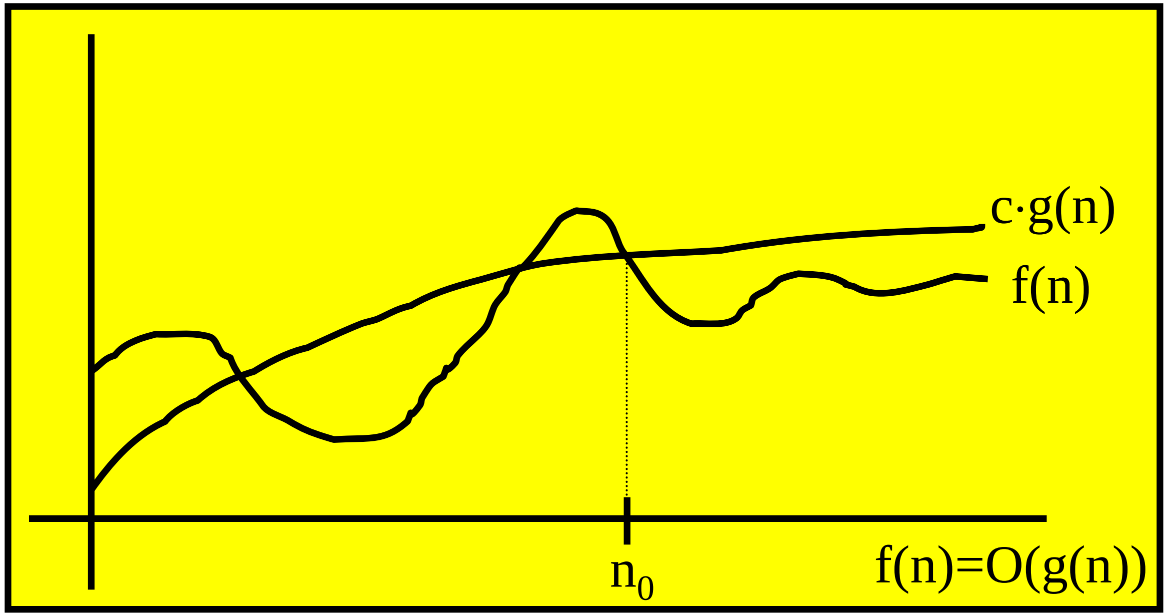


סיבוכיות והסימון O

<u>הגדרה</u>: יהיו $f(n), g(n)$ פונקציות חיוביות. נאמר שהפונקציה $f(n)$ נמצאת בקבוצת הפונקציות $O(g(n))$ אם קיימים קבועים n_0, c כך שלכל $n \geq n_0$ מתקיים :

$$f(n) \leq c \cdot g(n)$$

כמו כן נאמר ש- $g(n)$ מהווה <u>חסם עליון אסימפטוטי</u> לפונקציה $f(n)$ ונסמן זאת ע"י $f(n) = O(g(n))$ במקום הסימון הרגיל $f(n) \in O(g(n))$.

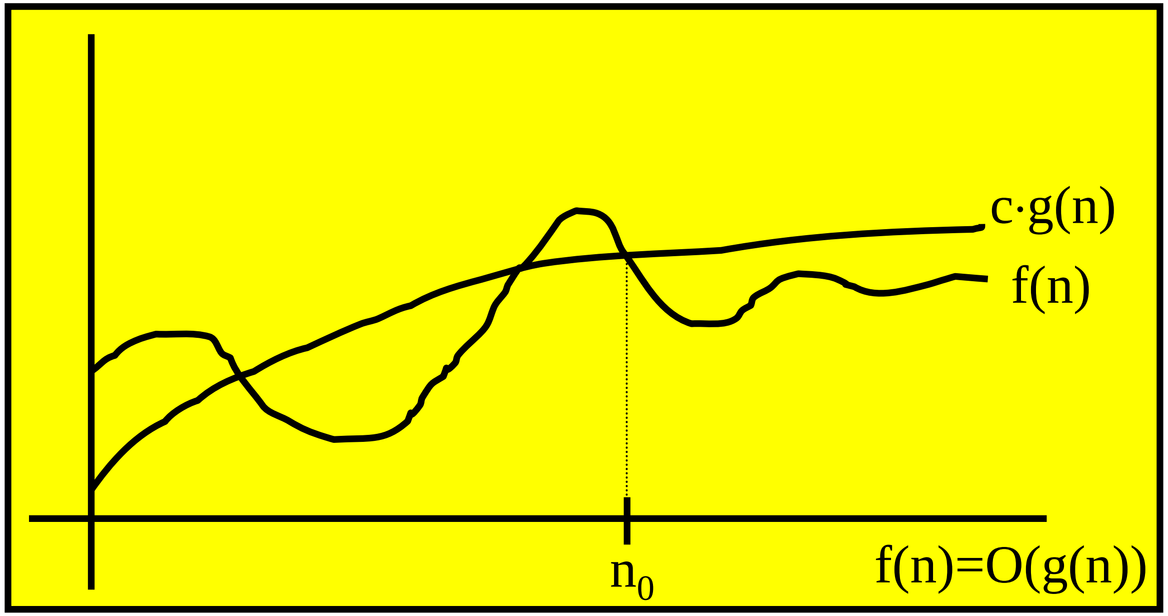


סיבוכיות והסימון O

<u>הגדרה</u>: יהיו $f(n), g(n)$ פונקציות חיוביות. נאמר שהפונקציה $f(n)$ נמצאת בקבוצת הפונקציות $O(g(n))$ אם קיימים קבועים n_0, c כך שלכל $n \geq n_0$ מתקיים :

$$f(n) \leq c \cdot g(n)$$

כמו כן נאמר ש- $g(n)$ מהווה <u>חסם עליון אסימפטוטי</u> לפונקציה $f(n)$ ונסמן זאת ע"י $f(n) = O(g(n))$ במקום הסימון הרגיל $f(n) \in O(g(n))$.



נשתמש בסימון זה כאשר הפונקציה $f(n)$ היא פונקציה שקשה לתאר במדויק, למשל זמן הריצה של אלגוריתם, בעוד $g(n)$ פשוטה יותר לתיאור.

דוגמאות פולינומיאליות

טענה: עבור k קבוע מתקיים $f(n) = a_0 + a_1 n + a_2 n^2 + \ldots + a_k n^k = O(n^k)$

הוכחה: נמצא קבועים c, n_0 כך כך שלכל $n \geq n_0$ יתקיים $f(n) \leq c \cdot n^k$.

יהי $a_{max} = \max\{0, a_0, \ldots, a_k\}$.

דוגמאות פולינומיאליות

<u>טענה</u>: עבור k קבוע מתקיים $f(n) = a_0 + a_1 n + a_2 n^2 + \ldots + a_k n^k = O(n^k)$

<u>הוכחה</u>: נמצא קבועים c, n_0 כך כך שלכל $n \geq n_0$ יתקיים $f(n) \leq c \cdot n^k$.

יהי $a_{max} = \max\{0, a_0, \ldots, a_k\}$.

$$f(n) \leq a_{max}(1 + n + n^2 + \ldots + n^k)$$

©cs,Technion

דוגמאות פולינומיאליות

טענה: עבור k קבוע מתקיים $f(n) = a_0 + a_1 n + a_2 n^2 + ... + a_k n^k = O(n^k)$

הוכחה: נמצא קבועים c, n_0 כך כך שלכל $n \geq n_0$ יתקיים $f(n) \leq c \cdot n^k$.

יהי $a_{max} = \max\{0, a_0, .., a_k\}$.

$$f(n) \leq a_{\max}(1 + n + n^2 + ... + n^k) = a_{\max} \frac{n^{k+1} - 1}{n - 1}$$

דוגמאות פולינומיאליות

טענה: עבור k קבוע מתקיים $f(n)=a_0+a_1n+a_2n^2+\ldots+a_kn^k=O(n^k)$

הוכחה: נמצא קבועים c, n_0 כך כך שלכל $n\geq n_0$ יתקיים $f(n)\leq c\cdot n^k$.

יהי $a_{max}=\max\{0,a_0,\ldots,a_k\}$.

$$f(n)\leq a_{\max}(1+n+n^2+\ldots+n^k)=a_{\max}\frac{n^{k+1}-1}{n-1}\leq a_{\max}\frac{n^{k+1}}{n/2}$$

דוגמאות פולינומיאליות

<u>טענה</u>: עבור k קבוע מתקיים $f(n) = a_0 + a_1 n + a_2 n^2 + \ldots + a_k n^k = O(n^k)$

<u>הוכחה</u>: נמצא קבועים c, n_0 כך כך שלכל $n \geq n_0$ יתקיים $f(n) \leq c \cdot n^k$.

יהי $a_{max} = \max\{0, a_0, \ldots, a_k\}$.

$$f(n) \leq a_{\max}(1 + n + n^2 + \ldots + n^k) \quad = a_{\max}\frac{n^{k+1}-1}{n-1} \quad \leq a_{\max}\frac{n^{k+1}}{n/2} \quad = 2a_{\max}n^k$$

©cs,Technion

דוגמאות פולינומיאליות

טענה: עבור k קבוע מתקיים $f(n) = a_0 + a_1 n + a_2 n^2 + ... + a_k n^k = O(n^k)$

הוכחה: נמצא קבועים c, n_0, כך כך שלכל $n_0 \le n$ יתקיים $f(n) \le c \cdot n^k$.

יהי $a_{max} = \max\{0, a_0, .., a_k\}$.

$$f(n) \le a_{max}(1 + n + n^2 + ... + n^k) = a_{max}\frac{n^{k+1} - 1}{n - 1} \le a_{max}\frac{n^{k+1}}{n/2} = 2a_{max}n^k$$

לפיכך נבחר $n_0 = 2$ וכן $c = 2a_{max}$.

דוגמאות פולינומיאליות

טענה: עבור k קבוע מתקיים $f(n) = a_0 + a_1 n + a_2 n^2 + \ldots + a_k n^k = O(n^k)$

הוכחה: נמצא קבועים c, n_0 כך כך שלכל $n_0 \le n$ יתקיים $f(n) \le c \cdot n^k$.

יהי $a_{max} = \max\{0, a_0, \ldots, a_k\}$.

$$f(n) \le a_{\max}(1 + n + n^2 + \ldots + n^k) = a_{\max} \frac{n^{k+1} - 1}{n - 1} \le a_{\max} \frac{n^{k+1}}{n/2} = 2 a_{\max} n^k$$

לפיכך נבחר $n_0 = 2$ וכן $c = 2a_{max}$.

דוגמאות: קבוע $f(n) = 10006 = O(1)$

ליניארי $f(n) = 4n + 27 = O(n)$

ריבועי $f(n) = n \log_2 n + 3n^2 = O(n^2)$

הדוגמא האחרונה נובעת מאי השוויון $\log_2 n < n$.

©cs,Technion

דוגמאות נוספות

לוגריתמי $f(n) = \log_a n = O(\log_b n)$

מכיוון שמתקיים $\log_a n = \log_a b \cdot \log_b n$.

לכל ε חיובי (שברים ושלמים) מתקיים: $f(n) = \log_a n = O(n^{\varepsilon})$

©cs,Technion

דוגמאות נוספות

לוגריתמי $f(n) = \log_a n = O(\log_b n)$

מכיוון שמתקיים $\log_a n = \log_a b \cdot \log_b n$.

לכל ε חיובי (שברים ושלמים) מתקיים: $f(n) = \log_a n = O(n^{\varepsilon})$

בין ליניארי וריבועי $f(n) = 8 + 15\,n + 9\,n \log_2 n = O(n \log_2 n)$

כיוון שמתקיים $f(n) < 32\,n \log_2 n$.

©cs,Technion

דוגמאות נוספות

לוגריתמי

$f(n) = \log_a n = O(\log_b n)$

מכיוון שמתקיים $\log_a n = \log_a b \cdot \log_b n$.

לכל ε חיובי (שברים ושלמים) מתקיים: $f(n) = \log_a n = O(n^{\varepsilon})$

בין ליניארי וריבועי

$f(n) = 8 + 15\, n + 9\, n \log_2 n = O(n \log_2 n)$

כיוון שמתקיים $f(n) < 32\, n \log_2 n$.

אקספוננציאלי

$f(n) = n^k + a^n = O(a^n) \qquad (a > 1)$

$f(n) = a^n = O(b^n) \qquad (b \geq a)$

©cs,Technion

דוגמאות שליליות

האם מתקיים ?

$f(n) = 3^n = O(2^n)$

$f(n) = e^n = O(n^k)$ (קבוע k)

$f(n) = \log n = O(\log\log n)$

©cs,Technion

דוגמאות שליליות

האם מתקיים ?

$f(n) = 3^n = O(2^n)$

$f(n) = e^n = O(n^k)$ (קבוע k)

$f(n) = \log n = O(\log \log n)$

התשובה שלילית בשלושת המקרים. נוכיח את המקרה הראשון.

נניח בשלילה שקיימים קבועים c,n_0 כך שעבור $n > n_0$ יתקיים: $3^n < c \cdot 2^n$.

אבל אי שוויון זה אינו מתקיים עבור $n > \log_{1.5} c$. סתירה.

סיבוכיות זמן

דוגמא ראשונה: סכום איברי מערך. ראינו שמתקיים $T(n) < c_1 \cdot n + c_2$

ולפיכך $T(n) = O(n)$.

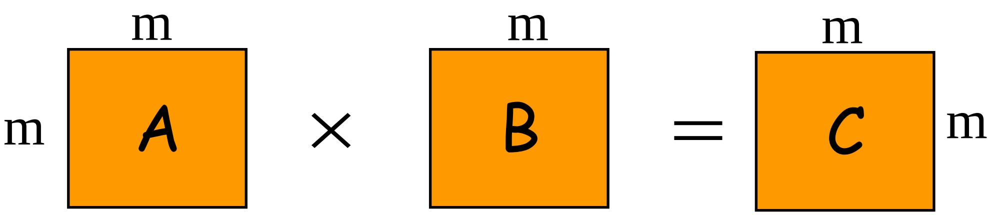
סיבוכיות זמן

דוגמא ראשונה: סכום איברי מערך. ראינו שמתקיים $T(n) < c_1 \cdot n + c_2$

ולפיכך $T(n) = O(n)$.

דוגמא שניה: כפל מטריצות ריבועיות בגודל $m \cdot m$.

גודל הקלט $n = 2m^2$.



מחשבים m^2 איברים במטריצת התוצאה C. כאשר כל איבר מחושב ע"י:

$$C[i,j] = \sum_{k=1}^{m} A[i,k] \cdot B[k,j]$$

סיבוכיות הזמן כפונקציה של m היא $O(m^3)$.

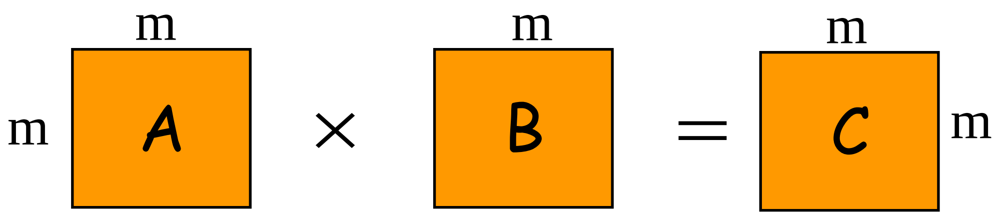
סיבוכיות זמן

דוגמא ראשונה: סכום איברי מערך. ראינו שמתקיים $T(n) < c_1 \cdot n + c_2$

ולפיכך $T(n) = O(n)$.

דוגמא שנייה: כפל מטריצות ריבועיות בגודל $m \cdot m$.

גודל הקלט $n = 2m^2$.



מחשבים m^2 איברים במטריצת התוצאה C. כאשר כל איבר מחושב ע"י:

$$C[i,j] = \sum_{k=1}^{m} A[i,k] \cdot B[k,j]$$

סיבוכיות הזמן כפונקציה של m היא $O(m^3)$.

סיבוכיות הזמן כפונקציה של גודל הקלט n היא $O(n^{3/2})$ כיוון שמתקיים:

$$T(n) = O(m^3) = O\left(\frac{n^{3/2}}{2^{3/2}}\right) = O(n^{3/2})$$

קבוע

סיבוכיות זמן (המשך)

דוגמא שלישית: חיפוש בינרי של x במערך ממוין בן n איברים.

בכל צעד משווים את x לאיבר האמצעי במערך העכשווי.

- אם האיבר האמצעי שווה ל- x החיפוש נגמר.
- אם האיבר האמצעי גדול מ- x ממשיכים עם חלק המערך המכיל את המספרים הקטנים.
- אם האיבר האמצעי קטן מ- x ממשיכים עם חלק המערך המכיל את המספרים הגדולים.

סיבוכיות זמן (המשך)

<u>דוגמא שלישית</u>: חיפוש בינרי של x במערך ממוין בן n איברים.

בכל צעד משווים את x לאיבר האמצעי במערך העכשווי.

- אם האיבר האמצעי שווה ל- x החיפוש נגמר.
- אם האיבר האמצעי גדול מ- x ממשיכים עם חלק המערך המכיל את המספרים הקטנים.
- אם האיבר האמצעי קטן מ- x ממשיכים עם חלק המערך המכיל את המספרים הגדולים.

נסמן ב-T את סיבוכיות הזמן. מתקיימת <u>משוואת הנסיגה</u> הבאה:

$$T(1) = c_1 \qquad T(n) = c + T(\lceil n/2 \rceil)$$

סיבוכיות זמן (המשך)

<u>דוגמא שלישית</u>: חיפוש בינרי של x במערך ממוין בן n איברים.

בכל צעד משווים את x לאיבר האמצעי במערך העכשווי.

- אם האיבר האמצעי שווה ל- x החיפוש נגמר.
- אם האיבר האמצעי גדול מ- x ממשיכים עם חלק המערך המכיל את המספרים הקטנים.
- אם האיבר האמצעי קטן מ- x ממשיכים עם חלק המערך המכיל את המספרים הגדולים.

נסמן ב-T את סיבוכיות הזמן. מתקיימת <u>משוואת הנסיגה</u> הבאה:

$$T(1) = c_1 \qquad T(n) = c + T(\lceil n/2 \rceil)$$

בהנחה ש- n חזקה של 2 נקבל:

$$T(n) = c + c + T(n/4)$$

סיבוכיות זמן (המשך)

דוגמא שלישית: חיפוש בינרי של x במערך ממוין בן n איברים.

בכל צעד משווים את x לאיבר האמצעי במערך העכשווי.

- אם האיבר האמצעי שווה ל- x החיפוש נגמר.
- אם האיבר האמצעי גדול מ- x ממשיכים עם חלק המערך המכיל את המספרים הקטנים.
- אם האיבר האמצעי קטן מ- x ממשיכים עם חלק המערך המכיל את המספרים הגדולים.

נסמן ב-T את סיבוכיות הזמן. מתקיימת משוואת הנסיגה הבאה:

$$T(1) = c_1 \qquad T(n) = c + T(\lceil n/2 \rceil)$$

בהנחה ש- n חזקה של 2 נקבל:

$$T(n) = c + c + T(n/4) = c \underbrace{+ c + \ldots +}_{i} c + T(n/2^i)$$

סיבוכיות זמן (המשך)

דוגמא שלישית: חיפוש בינרי של x במערך ממוין בן n איברים.

בכל צעד משווים את x לאיבר האמצעי במערך העכשווי.

- אם האיבר האמצעי שווה ל- x החיפוש נגמר.
- אם האיבר האמצעי גדול מ- x ממשיכים עם חלק המערך המכיל את המספרים הקטנים.
- אם האיבר האמצעי קטן מ- x ממשיכים עם חלק המערך המכיל את המספרים הגדולים.

נסמן ב-T את סיבוכיות הזמן. מתקיימת משוואת הנסיגה הבאה:

$$T(1) = c_1 \qquad T(n) = c + T(\lceil n/2 \rceil)$$

בהנחה ש- n חזקה של 2 נקבל:

$$T(n) = c + c + T(n/4) = \overbrace{c + c + \ldots + c}^{i} + T(n/2^i)$$

$$= c + c + \ldots + c + T(n/2^{\log_2 n})$$

סיבוכיות זמן (המשך)

דוגמא שלישית: חיפוש בינרי של x במערך ממוין בן n איברים.

בכל צעד משווים את x לאיבר האמצעי במערך העכשווי.

- אם האיבר האמצעי שווה ל- x החיפוש נגמר.
- אם האיבר האמצעי גדול מ- x ממשיכים עם חלק המערך המכיל את המספרים הקטנים.
- אם האיבר האמצעי קטן מ- x ממשיכים עם חלק המערך המכיל את המספרים הגדולים.

נסמן ב-T את סיבוכיות הזמן. מתקיימת משוואת הנסיגה הבאה:

$$T(1) = c_1 \qquad T(n) = c + T(\lceil n/2 \rceil)$$

בהנחה ש- n חזקה של 2 נקבל:

$$T(n) = c + c + T(n/4) = \overbrace{c + c + \ldots + c}^{i} + T(n/2^i)$$

$$= c + c + \ldots + c + T(n/2^{\log_2 n}) = c\log_2 n + T(1) = O(\log n)$$

סיבוכיות זמן (המשך)

דוגמא שלישית: חיפוש בינרי של x במערך ממוין בן n איברים.

בכל צעד משווים את x לאיבר האמצעי במערך העכשווי.

- אם האיבר האמצעי שווה ל- x החיפוש נגמר.
- אם האיבר האמצעי גדול מ- x ממשיכים עם חלק המערך המכיל את המספרים הקטנים.
- אם האיבר האמצעי קטן מ- x ממשיכים עם חלק המערך המכיל את המספרים הגדולים.

נסמן ב-T את סיבוכיות הזמן. מתקיימת משוואת הנסיגה הבאה:

$$T(1) = c_1 \qquad T(n) = c + T(\lceil n/2 \rceil)$$

בהנחה ש- n חזקה של 2 נקבל:

$$T(n) = c + c + T(n/4) = c + \overbrace{c + \ldots + c}^{i} + T(n/2^i)$$

$$= c + c + \ldots + c + T(n/2^{\log_2 n}) \quad = c\log_2 n + T(1) = O(\log n)$$

הטענה נכונה גם ללא ההנחה על n. פרטים על שיטות פתרון למשוואות נסיגה ניתן למצוא

בספר הלימוד (פרק 4): Cormen, Leiserson, Rivest, Introduction to Algorithms.

סיבוכיות זמן (המשך)

דוגמא רביעית:

```
S = 0;
for ( i = 1;  i < n;  i ++)
          for ( j = 0;  j< n;  j + = i )
                    S++ ;
```

אנליזה גסה: שתי לולאות מקוננות. $T(n) \le n\,(n-1)$.

סיבוכיות זמן (המשך)

דוגמא רביעית:

```
S = 0;
for ( i = 1;  i < n;  i ++)
        for ( j = 0;  j< n;  j + = i )
                S++ ;
```

אנליזה גסה: שתי לולאות מקוננות. $T(n) \le n\,(n-1)$.

אנליזה עדינה:

כאשר $i = 1$, מתבצעות n פעולות.

$i = 2$, $\lfloor n/2 \rfloor$

$i = 3$, $\lfloor n/3 \rfloor$

•••

$i = n$, $n/n = 1$

סה"כ $T(n) \le n + \frac{n}{2} + \frac{n}{3} + \ldots + \frac{n}{n} = n\left(1 + \frac{1}{2} + \frac{1}{3} + \ldots + \frac{1}{n}\right) = nH_n$

סיבוכיות זמן (המשך)

דוגמא רביעית:

```
S = 0;
for ( i = 1;  i < n;  i ++)
        for ( j = 0;  j< n;  j + = i )
                S++ ;
```

אנליזה גסה: שתי לולאות מקוננות. $T(n) \le n\ (n\text{-}1)$.

אנליזה עדינה:

כאשר $i = 1$, מתבצעות n פעולות.

$i = 2$, $\lfloor n/2 \rfloor$

$i = 3$, $\lfloor n/3 \rfloor$

•••

$i = n$, $n/n = 1$

סה"כ
$$T(n) \le n + \frac{n}{2} + \frac{n}{3} + \ldots + \frac{n}{n} = n\left(1 + \frac{1}{2} + \frac{1}{3} + \ldots + \frac{1}{n}\right) = nH_n$$

H_n נקרא המספר ההרמוני ה-n

כיוון שמתקיים:
$$\ln(n+1) = \int_1^{n+1} \frac{dx}{x} \le 1 + \frac{1}{2} + \frac{1}{3} + \ldots + \frac{1}{n} \le 1 + \int_1^n \frac{dx}{x} = 1 + \ln(n)$$

סיבוכיות זמן (המשך)

דוגמא רביעית:

```
S = 0;
for ( i = 1;  i < n;  i ++)
        for ( j = 0;  j< n;  j + = i )
                S++ ;
```

אנליזה גסה: שתי לולאות מקוננות. $T(n) \le n\,(n-1)$.

אנליזה עדינה:

כאשר $i = 1$, מתבצעות n פעולות.

$i = 2$, $\lfloor n/2 \rfloor$

$i = 3$, $\lfloor n/3 \rfloor$

•••

$i = n$, $n/n = 1$

סה"כ $$T(n) \le n + \frac{n}{2} + \frac{n}{3} + \ldots + \frac{n}{n} = n\left(1 + \frac{1}{2} + \frac{1}{3} + \ldots + \frac{1}{n}\right) = nH_n$$

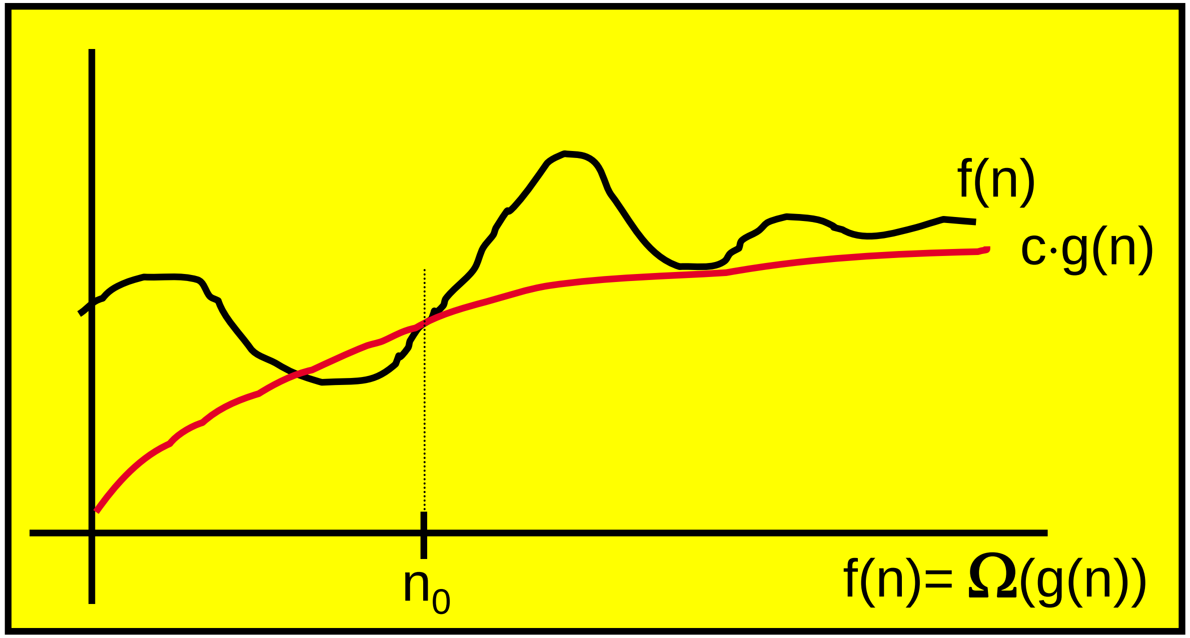
H_n נקרא המספר ההרמוני ה-n

כיוון שמתקיים: $$\ln(n+1) = \int_1^{n+1} \frac{dx}{x} \le 1 + \frac{1}{2} + \frac{1}{3} + \ldots + \frac{1}{n} \le 1 + \int_1^{n} \frac{dx}{x} = 1 + \ln(n)$$

הערה: לשם דיוק היה עלינו לכפול את כל המשוואות בקבוע ולפיכך: $T(n) = O(n \cdot \log n)$

חסם תחתון אסימפטוטי

הגדרה: יהיו $f(n), g(n)$ פונקציות חיוביות. נאמר שהפונקציה $f(n)$ נמצאת בקבוצת הפונקציות $\Omega(g(n))$ (אומגה) אם קיימים קבועים c, n_0 כך שלכל $n \geq n_0$ מתקיים :
$f(n) \geq c \cdot g(n)$

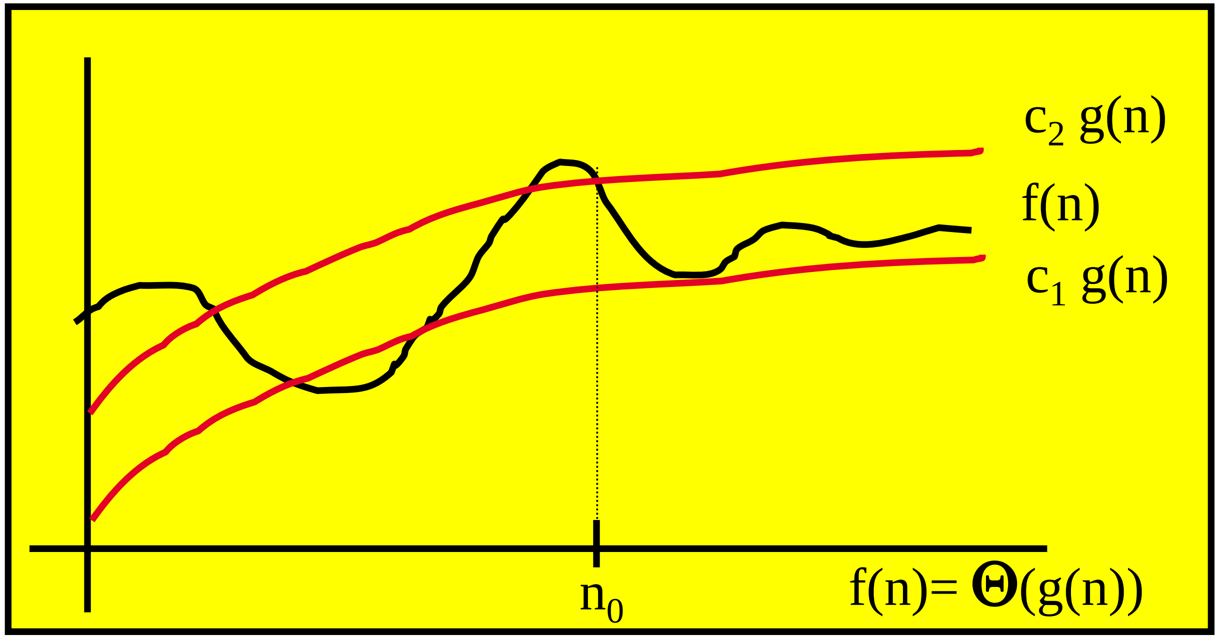


קיימות הגדרות נוספות לחסם תחתון אשר שקולות להגדרה לעיל עבור פונקציות מונוטוניות .

©cs,Technion

חסם הדוק אסימפטוטי

<u>הגדרה</u>: יהיו $f(n)$, $g(n)$ פונקציות חיוביות. נאמר שמתקיים $f(n)=\Theta(g(n))$ (תטה) אם $f(n)=O(g(n))$ וגם $f(n)=\Omega(g(n))$.



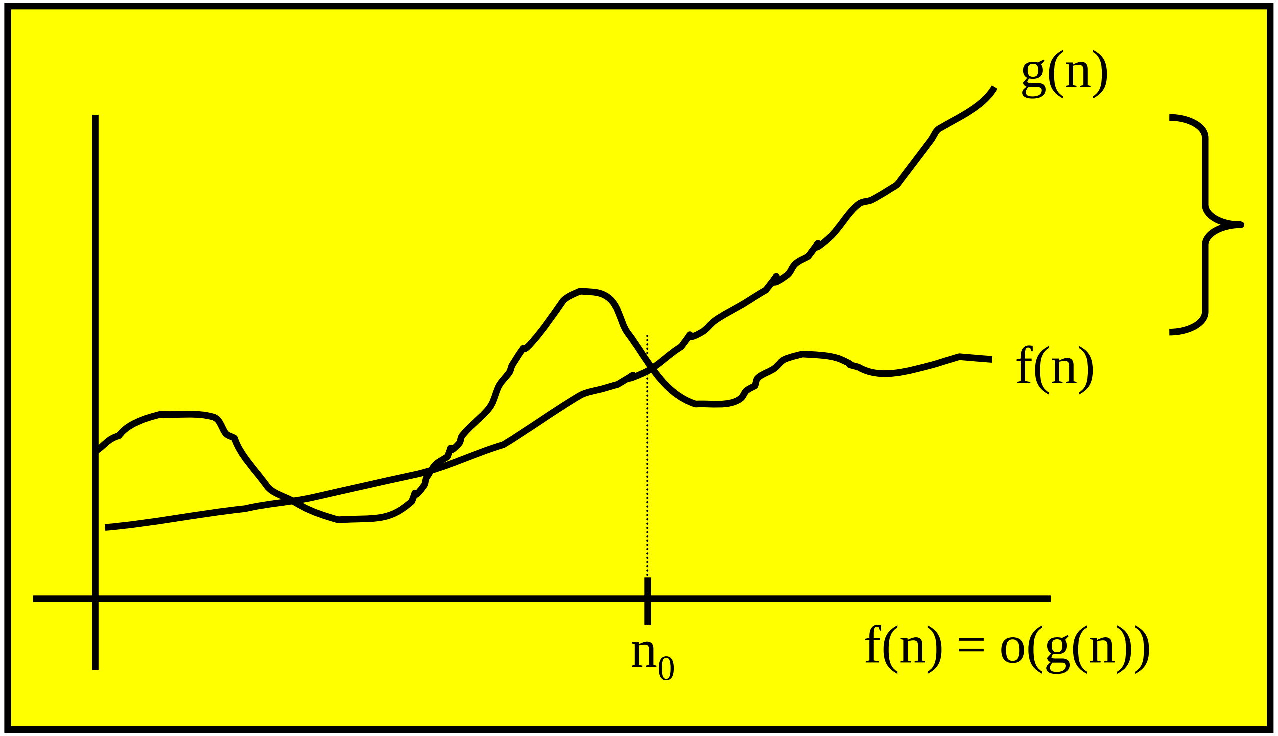
<u>הגדרה (אקויולנטית)</u>: נאמר שמתקיים $f(n)=\Theta(g(n))$ אם קיימים קבועים n_0,c_1,c_2 כך שלכל $n \geq n_0$ מתקיים $c_1 \cdot g(n) \leq f(n) \leq c_2 \cdot g(n)$.

©cs,Technion

הסימון o קטן

<u>הגדרה</u>: יהיו $f(n), g(n)$ פונקציות חיוביות. נאמר שהפונקציה $f(n)$ נמצאת בקבוצת הפונקציות $o(g(n))$ אם <u>לכל קבוע c</u> קיים קבוע n_0 כך שלכל $n \geq n_0$ מתקיים :

$$f(n) \leq c \cdot g(n)$$



$f(n)$ זניחה אסימפטוטית יחסית לפונקציה $g(n)$.

<u>הגדרה (אקויולנטית)</u>: נאמר שמתקיים $f(n)=o(g(n))$ אם $\lim_{n\to\infty} f(n)/g(n) = 0$.

דוגמאות: $n\text{-}100 \neq o(n)$, $\log n = o(n)$,

©cs,Technion

מגבלות הסימון האסימפטוטי

נוח להשתמש בסימונים אסימפטוטיים מפני שהסימון מתעלם מקבועים ומאפשר ניתוח זמנים פשוט יותר. אנו נשתמש בסימונים אלה לאורך הקורס.

אבל לסימון יש מגבלות מסוכנות...

ברור שנעדיף תוכנית הרצה בזמן $T(n) = n^2$ על פני תוכנית הרצה בזמן קבוע של 10^{80} (ממומשת למשל ע"י טבלה ענקית של תשובות לכל אפשרות) כיון שבתוכניות ממשיות אנו משתמשים בגודל קלט n סופי הקטן באופן משמעותי מ- 10^{40}.

<u>מסקנה</u>: צריך לוודא שהקבועים n_0,c המתחבאים בהגדרות אסימפטוטיות O, Θ, Ω אמנם "סבירים".

מגבלות הסימון האסימפטוטי

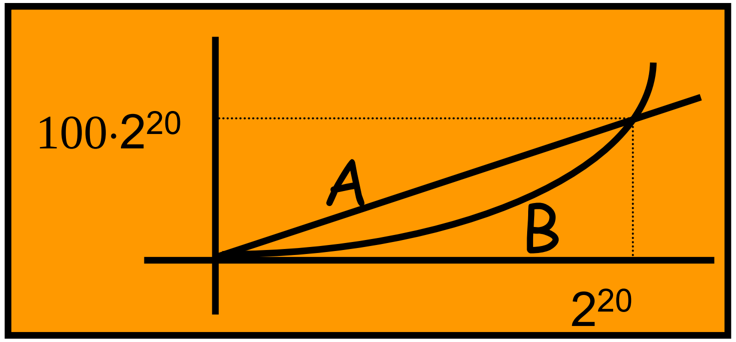
נוח להשתמש בסימונים אסימפטוטיים מפני שהסימון מתעלם מקבועים ומאפשר ניתוח זמנים פשוט יותר. אנו נשתמש בסימונים אלה לאורך הקורס.

אבל לסימון יש מגבלות מסוכנות...

ברור שנעדיף תוכנית הרצה בזמן $T(n) = n^2$ על פני תוכנית הרצה בזמן קבוע של 10^{80} (ממומשת למשל ע"י טבלה ענקית של תשובות לכל אפשרות) כיון שבתוכניות ממשיות אנו משתמשים בגודל קלט n סופי הקטן באופן משמעותי מ- 10^{40}.

<u>מסקנה</u>: צריך לוודא שהקבועים n_0, c המתחבאים בהגדרות אסימפטוטיות O, Θ, Ω אמנם "סבירים".

<u>דוגמא</u>. נניח שאלגוריתם A רץ בזמן $T_A(n)= 100\ n$ ואלגוריתם B רץ בזמן $T_B(n)= 5n\log_2 n$. בניתוח אסימפטוטי עדיף אלגוריתם A כיון שהאלגוריתם ליניארי, אבל עבור קלטים המקיימים $n < 2^{20}$ עדיף אלגוריתם B.



©cs,Technion