סיכום מבני נתונים ומבוא לאלגוריתמים

יחסים אסימפטוטים

$$= O(n^k) = O(c^k) = O(n!) = O(n^n)$$

סכום סדרה חשבונית: $S = \frac{(a_1 + a_n) \cdot n}{2}$

סכום סדרה הנדסית: $S = \frac{a1 \cdot (q^n - 1)}{q - 1}$

סכום טור הנדסי מתכנס: $S = \frac{a_1}{1 - q}$

פתרון נוסחאות נסיגה

שיטת ההצבה: שיטה שנועדה רק להוכיח סיבוכיות שניחשנו מראש. הוכחה בעזרת אינדוקציה.

שיטת האיטרציות: פותחים את הרקורסיה כסכום איברים התלויים בתנאי ההתחלה וב-n.

שיטת המאסטר:

- $T(n)$ מוגדרת על שלמים אי שליליים ע"י הרקורסיה: $T(n) = a \cdot T\left(\frac{n}{b}\right) + f(n)$
- $a \geq 1$ ו $b \geq 1$ קבועים.
- $f(n)$ פונקציה כלשהי.

ניתן לחסום את $T(n)$ אסימפטוטית באופן הבא:

1. אם $f(n) = O(n^{\log_b a - \epsilon})$ כאשר $\epsilon > 0$ קבוע כלשהו,
   אזי $T(n) = \Theta(n^{\log_b a})$

2. אם $f(n) = \Theta(n^{\log_b a})$
   אזי $T(n) = \Theta\left(n^{\log_b a} \cdot \log n\right)$

   "מקרה 2 מורחב":
   אם $f(n) = \Theta(n^{\log_b a} \log^k n)$, עבור $k \geq 0$
   אזי $T(n) = \Theta(n^{\log_b a} \log^{k+1} n)$

3. אם $f(n) = \Omega(n^{\log_b a + \epsilon})$ כאשר $\epsilon > 0$ קבוע כלשהו,
   <u>וגם</u> $a \cdot f\left(\frac{n}{b}\right) \leq c \cdot f(n)$ עבור קבוע $c < 1$ והחל מ-n מספיק גדול
   אזי $T(n) = \Theta(f(n))$

הגדרות

בעיית הבחירה – מקבלת קבוצה A של n מספרים שונים ומספר i ומחזירה את האיבר הגדול בדיוק מ i-1 האיברים האחרים ב-A. (ערך מיקום i)

<u>מציאת חציון</u> –

n אי זוגי – החציון הוא: $i = \frac{(n+1)}{2}$.

n זוגי – החציון התחתון הוא: $i = \left\lfloor\frac{(n+1)}{2}\right\rfloor$

או פשוט $i = \left\lfloor\frac{(n+1)}{2}\right\rfloor$ לשני המקרים.

מיונים

אלגוריתם	**קריאה**	**מקרה ממוצע**	**מקרה גרוע**	**הערה**
מיון הכנסה	INSERTION-SORT(A)	$O(n^2)$	$O(n^2)$	
מיון מיזוג		O(nlgn)	O(nlgn)	
מיון ערימה	HEAPSORT(A)	O(nlgn)	O(nlgn)	
מיון מהיר	QUICKSORT(A,p,r)	O(nlgn)	$O(n^2)$	

מיון מנייה	COUNTING-SORT(A,B,k)	O(n+k), אם k=o(n) אז O(n)	O(n+k)	הנחה: ערכי המערך בתחום מסויים
מיון בסיס	RADIX-SORT(A,d)	Θ (d*(n+k))	Θ (d*(n+k))	הנחה: לאיברים עד d ספרות. לכל ספרה k אפשרויות.
מיון דלי	BUCKET-SORT(A)	O(n)	(nlgn)	הנחה: האיברים מפוזרים באופן אחיד ובלתי תלוי בקטע [0,1).

<u>מערך לא ממויין</u>

אלגוריתם	מטרה	מקרה ממוצע	מקרה גרוע
MINIMUM(A)	מציאת מינימום	O(n)	O(n)
MAXIMUM(A)	מציאת מקסימום	O(n)	O(n)
PARTITION(A,p,r)	חלוקת המערך	O(n)	O(n)
SELECT	מציאת ערך מיקום (גם חציון)	O(n)	O(n)
RANDOMIZED-SELECT(A,p,r,i)	ערך מיקום (לא יעיל)	O(n)	$O(n^2)$

<u>מערך ממויין</u>

מטרה	שיטה	זמן ריצה
מציאת שני איברים שסכומם s	ריצה עם 2 מצביעים: מההתחלה ומהסוף (תקף גם לשני מערכים ממויינים שונים)	O(n)
מציאת שני איברים שהפרשם d	ריצה עם 2 מצביעים: שניהם מההתחלה	O(n)
מציאת איבר	חיפוש בינארי	O(lgn)
מציאת קצה של איבר חוזר	וריאציה של חיפוש בינארי	O(lgn)
מיזוג שני מערכים ממוינים	ביצוע אלגוריתם Merge	O(n)

<u>ערמה</u>

בערמת מינימום/מקסימום מתקיים כל צומת קטנה/גדולה מכל בניה. סדר הבנים הפנימי אינו משנה.

יתרון: בניה ב- O(n), הוצאת מינימום ב-O(lgn), מציאת מינימום

חיסרון: חיפוש מפתח ב- O(n)

פעולה	מטרה	זמן ריצה
LEFT(i)	החזרת הבן השמאלי של i	O(1)
RIGHT(i)	החזרת הבן הימני של i	O(1)
PARENT(i)	החזרת האב של i	O(1)
MAX-HEAPIFY(A,i)	מקבלת ערמה המניחה כי הבנים של i הם ערמות תקניות, ואולי A[i] מפר את תכונות הערימה, אז היא מחליקה את A[i] במורד הערימה עד שתת העץ המושרש באינדקס i יהפוך לערימה.	O(h)
BUILD-MAX-HEAPIFY(A)	מקבל מערך והופל אותו לערמת מקסימום	O(n)
תור קדימויות		
MAX-HEAP-INSERT(S,x)	מכניסה את האיבר הx לקבוצה S	O(lgn)
HEAP-MAXIMUM(S)	מחזירה את האיבר בעל הערך הגדול ביותר בS	O(1)
HEAP-EXTRACT-MAX(S)	מוציאה מ-S את האיבר בעל המפתח הגדול ביותר ומחזירה אותו.	O(lgn)
HEAP-INCREASE-KEY(S,x,k)	מעלה את ערך המפתח של האיבר x ל-k. מניחים שהערך החדש גדול או שווה לערך המפתח של x לפני הפעולה.	O(lgn)

מחסנית

פעולה	מטרה	זמן ריצה
STACK-EMPTY(S)	האם המחסנית ריקה?	O(1)
PUSH(S,x)	דחוף איבר למחסנית	O(1)
POP(S)	הוצא והחזר את ראש המחסנית	O(1)

תור

פעולה	מטרה	זמן ריצה
ENQUEUE(Q,x)	הכנסת איבר לסוף התור	O(1)
DEQUEUE(Q,x)	הוצאת איבר מהראש התור	O(1)

רשימה מקושרת

פעולה	מטרה	זמן ריצה
LIST-SEARCH(L,k)	חיפוש האיבר k ברשימה L.	O(n)
LIST-INSERT(L,x)	משחילה את x לראש הרשימה המקושרת	O(1)
LIST-DELETE(L,x)	מוחקת את האיבר x מהרשימה המקושרת	O(1)

טבלאות גיבוב

מיעון סגור:

פעולה	מטרה	מקרה ממוצע	מקרה גרוע
CHAINED-HASH-INSERT(T,x)	הכנס את x בראש הרשימה T[h(key[x])]	O(1)	O(1)
CHAINED-HASH-SEARCH(T,k)	חפש איבר בעל המפתח k ברשימה T[h(k)]	O(1)	O(n)
CHAINED-HASH-DELETE(T,x)	מחק את x מהרשימה T[h(key[x])]	O(1)	O(1)

מיעון פתוח:

פעולה	מטרה	זמן ריצה
HASH-INSERT(T,k)	הכנסה	
HASH-SERCH(T,k)	חיפוש	

עץ חיפוש בינארי

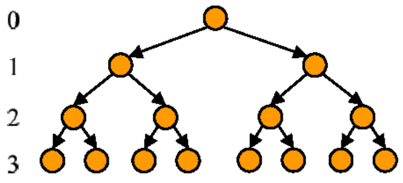
"עץ מלא" – עץ שבו לכל צומת פנימי 2 בנים.

"עץ שלם" – עץ בינארי מלא שבו כל העלים באותו עומק.

תכונות עצים בינריים שלמים

בעץ בינרי שלם בעל n צמתים, L עלים, וגובה h:

1. מספר הצמתים בעומק i: $n_i = 2^i$
2. מספר העלים: $L = n_h = 2^h$
3. מספר הצמתים: $n = \sum_{i=0}^{h} n_i = \sum_{i=0}^{h} 2^i = 2^{h+1} - 1$
4. הגובה: $h = \log_2(n+1) - 1$
5. מספר הצמתים הפנימיים: $n - L = 2^h - 1 = L - 1$



h – גובה העץ. (במקרה הכי גרוע, גובה העץ שווה למספר איבריו h=n)

פעולה	מטרה	זמן ריצה
TREE-SEARCH(x,k)	מחפש צומת בעץ x בעל המפתח k	O(h)
TREE-MINIMUM(x)	מחזיר את האיבר בעל המפתח הקטן ביותר בעץ. *איבר זה יהיה תמיד העלה בשמאלי ביותר.*	O(h)
TREE-MAXIMUM(x)	מחזיר את האיבר בעל המפתח הגדול ביותר בעץ. *איבר זה יהיה תמיד העלה הימני ביותר.*	O(h)
TREE-SUCCESOR(x) עוקב	מחזיר את הצומת המכילה את המפתח העוקב לx. *אם לx יש בן ימני יורדים ימינה פעם אחת ואז שמאלה עד הסוף.* *אם אין לx אין בן ימני עולים שמאלה כמה שאפשר ואז הולכים ימינה פעם אחת .*	O(h)
TREE-PREDECESSOR(x) קודם	מחזיר את הצומת המכילה את המפתח הקודם לx. *אם לx יש בן שמאלי יורדים שמאלה פעם אחת ואז ימינה עד הסוף.* *אם אין לx בן שמאלי עולים ימינה כמה שאפשר ואז הולכים שמאלה פעם אחת.*	O(h)
TREE-INSRET(T,z)	הכנסת הצומת z לעץ חיפוש בינארי T.	O(h)
TREE-DELETE(T,z)	מחיקת הצומת z מעץ חיפוש בינארי T.	O(h)
BFS(T)	הדפסת העץ בסדר של הרמות (סריקה לרוחב)	O(n)

<u>**עץ אדום שחור**</u> (מקרה פרטי של עץ חיפוש בינארי)

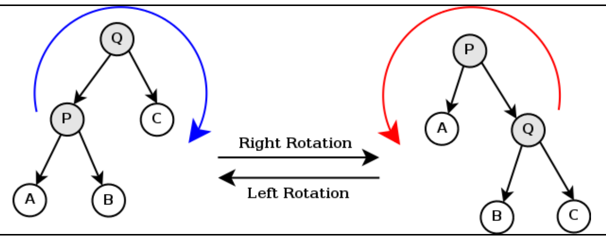
עץ חיפוש בינארי הוא עץ אדום שחור אם הוא מקיים את **תכונות האדום-שחור**:

1. כל צומת הוא אדום או שחור.
2. השורש הוא שחור.
3. כל עלה (NIL) הוא שחור.
4. אם צומת הוא אדום, אזי שני בניו שחורים.
5. כל המסלולים הפשוטים מצומת נתון כלשהו לצאצאים עלים מכילים אותו מספר של צמתים שחורים.

יתרון: חיפוש איבר ב-O(lgn) במקרה הגרוע.

חיסרון: בניה מרשימה (לא ממויינת) ב-O(nlgn)

<u>למה 13.1</u>: גובהו של עץ אדום שחור המכיל n צמתים פנימיים הוא לכל היותר 2lg(n+1).

פעולה	מטרה	זמן ריצה
LEFT-ROTATE(T,x)		O(1)
RIGHT-ROTATE(T,x)		O(1)
RB-INSERT(T,z)	מכניס לעץ א"ש T את הצומת z שמניחים ששדה המפתח שלו כבר מכיל ערך.	O(lgn)
RB-INSERT-FIXUP(T,z)	מאחר שצביעת z בצבע אדום עלולה להפר את אחת מתכונות האדום-שחור, קריאה למתודה משיבה את תכונות האדום-שחור.	O(lgn)
RB-DELETE(T,z)	מחיקת הצומת z מא"ש T.	O(lgn)
RB-DELETE-FIXUP(T,z)	קריאה למתודה משיבה את תכונות האדום-שחור לאחר הכנסה של z.	O(lgn)
TREE-SEARCH(T,k)	מחפשת את המפתח k בעץ א"ש T. *מתודה זו היא המצאה שלי.*	O(lgn)
בניית עץ אדום שחור	מקבלת מערך ובונה ממנו עץ עדום שחור. השגרה עוברת על איברי המערך ובכל פעם קוראת ל--RB INSERT עם האיבר. במקרה של מערך ממוין נבנה עץ חיפוש בינארי מאוזן	O(nlgn) – לא ממוין O(n) -ממוין

	בדרך הידועה, אז נהפוך אותו לא"ש.	

<u>עץ אדום שחור מורחב – עץ ערכי מיקום</u>

עבור כל צומת נשמור דירוג שהוא מספר הצאצאים האמיתיים שלו + 1 (הוא עצמו).

כלומר לכל עלה יהיה דירוג 1 כי אין לו כלל צאצאים.

פעולה	מטרה	זמן ריצה
OS-SELECT(x,i)	מחזירה מצביע לצומת המכיל את המפתח ה-i הקטן ביותר בתת-עץ המושרש ה-x.	O(lgn)
OS-RANK(T,x)	מחזירה את מיקומו של x בסדר הלינארי הנקבע ע"י סריקה תוכנית של T.	O(lgn)

<u>אלגוריתם BFS – סריקה לפי העומק של הצמתים</u>

```
BFS(T)
    ENQUEUE(Q,root [T])
    while not ISEMPTY(Q)
        do    p←DEQUEUE(Q)
              if left [ p] ≠ NIL
                  then ENQUEUE(Q,left [ p])
              if right [ p] ≠ NIL
                  then ENQUEUE(Q,right [ p])
              print key[p]
```

	$\frac{(n^2+n)}{2}$	$\sum_{i=0}^{n} i$
	$\frac{n(n+1)(2n+1)}{6}$	$\sum_{i=0}^{n} i^2$
	$\frac{n^4}{4}+\frac{n^3}{2}+\frac{n^2}{2}$	$\sum_{i=0}^{n} i^3$
	$\frac{n^5}{5}+\frac{n^4}{2}+\frac{n^3}{3}-\frac{n}{30}$	$\sum_{i=0}^{n} i^4$
	$(1/6)n^6 + (1/2)n^5 + (5/12)n^4 - (1/12)n^2$	$\sum_{i=0}^{n} i^5$
	$(1/7)n^7 + (1/2)n^6 + (1/2)n^5 - (1/6)n^3 + (1/42)n$	$\sum_{i=0}^{n} i^6$
	$(1/8)n^8 + (1/2)n^7 + (7/12)n^6 - (7/24)n^4 + (1/12)n^2$	$\sum_{i=0}^{n} i^7$

$\sum_{i=0}^{n} i^8$	(1/9)n^9 + (1/2)n^8 + (2/3)n^7 - (7/15)n^5 + (2/9)n^3 - (1/30)n	
$\sum_{i=0}^{n} i^9$	(1/10)n^10 + (1/2)n^9 + (3/4)n^8 - (7/10)n^6 + (1/2)n^4 - (3/20)n^2	
$\sum_{i=0}^{n} i^{10}$	(1/11)n^11 + (1/2)n^10 + (5/6)n^9 - n^7 + n^5 - (1/2)n^3 + (5/66)n	
$\boldsymbol{\sum_{i=0}^{n} 2^i}$	$2^{n+1} - 1$	
$\sum_{i=1}^{n} 2^i$	$2^{n+1} - 2$	
$\sum_{i=1}^{n} \frac{1}{i(i+1)}$	$\frac{n}{n+1}$	
$\sum_{i=1}^{n} i(i+1)(i+2)$	$\frac{n(n+1)(n+2)(n+3)}{4}$	
$\sum_{i=1}^{n} (2i-1)$	n^2	
$\sum_{i=1}^{n} (4i+1)$	n(2n+3)	
$\boldsymbol{\sum_{i=1}^{n} i \cdot \lg(i)}$	$\frac{\log(H(n))}{\log(10)}$	
$\sum_{i=1}^{n} \lg(i)$	$\frac{\log((1)_n)}{\log(10)}$	