סיכום מבני נתונים

מבנה נתונים	פעולות מוגדרות	סיבוכיות
מחסנית טוב כשצריכים את האיברים בצורת LIFO	Create(S) creates an empty stack	O(1)
	Push(S,x) insert x into the stack	O(1)
	Pop(S) extracts the last inserted item (returns void)	O(1)
	Is_Empty(S) boolean true if stack is empty	O(1)
	Top(S) returns the top item on the stack	O(1)
תור טוב כשצריכים את האיברים בצורת FIFO	Create(Q) Creates an empty Queue	O(1)
	Head(Q) Returns the first member in the Queue	O(1)
	Enqueue(Q,x) Insert x into the Queue	O(1)
	Dequeue(Q) Extract the first member in the Queue (returns void)	O(1)
	Is_empty(Q) Boolean returns true if Queue is empty	O(1)
רשימה מקושרת יש גם רשימה מעגלית ודו כיוונית	הכנסה והוצאה של איבר בהינתן מצביע למקום המתאים	O(1)
	חיפוש איבר	O(n)
עץ חיפוש AVL עץ חיפוש בינארי שבו הפרש גבהים בין בנים הוא לכל היותר 1	הכנסת איבר	O(log(n))
	הוצאת איבר	O(log(n))
	חיפוש איבר	O(log(n))
עץ 2-3 (עץ +B) עץ חיפוש שבו לכל צומת 2 או 3 בנים. זהו עץ +B מדרגה 3.	הכנסת איבר	O(log(n))
	הוצאת איבר	O(log(n))
	חיפוש איבר	O(log(n))
עץ דרגות - Rank tree תומך במציאת אינדקס של איבר בסיבוכיות לוגריתמית	הכנסת איבר	O(log(n))
	הוצאת איבר	O(log(n))
	חיפוש איבר	O(log(n))
	מציאת אינדקס של איבר (כמה איברים יש לפניו)	O(log(n))
רשימת דילוגים חסרון: סיבוכיות ממוצעת יתרון: מימוש פשוט	הכנסת איבר	~O(log(n))
	הוצאת איבר	~O(log(n))
	חיפוש איבר	~O(log(n))
Hash Table יתרון: זמן ממוצע O(1). חסרון: יש לדעת מראש סדר גודל מספר מפתחות	הכנסת איבר	~O(1)
	הוצאת איבר	~O(1)
	חיפוש איבר	~O(1)
Union/Find מימוש בעזרת עצים הפוכים	Makeset(i) Return new set with single element i	O(1)
	Find(i) Returns the set that i belongs to	O(log(n))
	Union(p,q) returns a new group containing all elements of groups p,q	O(1)
	סיבוכיות משוערכת (לא אמיתית!) ל m פעולות Union/Find	~O(m)
Heap טוב כאשר צריכים לקבל את האיבר הגדול או הקטן ביותר מהר. זהו לא עץ חיפוש. מציאת איבר ב O(n).	Make_heap(Q) Create an empty Heap	O(1)
	Make_heap(Q,A) Create a Heap from an array (עץ כמעט שלם)	O(n)
	Insert(x,Q) Insert x into the heap	O(log(n))
	Max(Q) Returns the maximal/minimal value in the heap	O(log(n))
	Del_max(Q) Deletes the maximal/minimal value in the heap	O(log(n))
Trie טוב כאשר עובדים עם מחרוזות של תווים או ספרות	הכנסת מחרוזת באורך s	O(s)
	הוצאת מחרוזת באורך s	O(s)
	חיפוש מחרוזת באורך s	O(s)
	מציאת מחרוזת מינימום לקסיקוגרפי באורך s	O(s)
עץ סיומות (Suffix Tree)	יצירת עץ סיומות מתוך מחרוזת באורך s	O(s)
	מציאת תת מחרוזת באורך k	O(k)
	מציאת תת מחרוזת משותפת למחרוזות באורך L_1, L_2	$O(L_1+L_2)$

סיכום סיבוכיות

הפונקציה f(n) נמצאת בקבוצת הפונקציות O(g(n)) אם קיימים קבועים c,n_0 כך שלכל $n_0 \le n$ מתקיים:
$f(n)=O(g(n)) \Leftrightarrow f(n) \le c*g(n)$
g(n) מהווה חסם עליון אסימפטוטי.
אותו הדבר רק הפוך לגבי חסם תחתון (אומגה).
$f(n)=\Omega(g(n)) \Leftrightarrow f(n) \ge c*g(n)$
$f(n)=\theta(g(n)) \Leftrightarrow f(n)=O(g(n)) \;\&\&\; f(n)=\Omega(g(n))$

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 רקורסיות:

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: פותרת רקורסיות מהצורה: $T(n)=aT(n/b)+f(n)$ כאשר $a\ge1$ ו $b>1$ קבועים ו f(n) פונקציה.
T(n) מוגדרת עבור שלמים אי שליליים.
שיטת הפתרון:
1. חשב $x=\log_b(a)$
2. אם $f(n)<c*n^{x-\varepsilon}$ אזי $T(n)=\theta(n^x)$
3. אם $f(n)=c*n^x$ אזי $T(n)=\theta(n^x\log(n))$
4. אם $f(n)>c*n^{x+\varepsilon}$ וגם $a*f(n/b)\le k*f(n)$ כאשר $k<1$ קבוע אזי $T(n)=\theta(f(n))$

מעבר על עצים:

Pre-Order: קודם על השורש, אז בן שמאלי ואז בן ימני
In-order: קודם בן שמאלי, אז שורש, ואז בן ימני
Post-Order: קודם בן שמאלי, אז בן ימני, ואז שורש.

מיונים:

Counting Sort (Bucket Sort) - מיון יציב (לא משנה סדר יחסי של איברים בעלי ערכים זהים) בו ניתן להשתמש כאשר תחום הערכים k סופי וידוע מראש (קבוע). סיבוכיות - O(n+k)
Radix-Sort - אם נתונים n מספרים בעלי d ספרות בבסיס b אזי ניתן למיינם בסיבוכיות O(d*(n+b)) וכאשר d,b קבועים O(n). משתמש ב Bucket-Sort ולכן הינו מיון יציב.
Quick-Sort - בבחירה אקראית של Pivot נותן סיבוכיות ממוצעת של $\theta(n*\log(n))$ ואם ניתן למצוא חציון ב O(1) אזי הסיבוכיות היא $\theta(n*\log(n))$ אמיתית.
Heap-Sort - בניית ערימת מינימום מהאיברים למיון, ואז הוצאתם אחד אחד מהקטן לגדול. סיבוכיות $O(n*\log(n))$

פונקציות ערבול יעילות:

1. שיטת המודולו
   $h(x)=x \bmod m$ כאשר m מספר ראשוני שאינו קרוב לחזקה של 2.

2. שיטת הכפל בקבוע בין 0 ל-1.
   $h(k)=\lfloor m\cdot(a\cdot k \bmod 1)\rfloor$
   א. הכפל את המפתח k ב- a.
   ב. קח רק את השבר העשרוני של התוצאה.
   ג. הכפל אותו ב-m (פקטור גודל טבלת ערבול) ועגל כלפי מטה.
   $a=\frac{\sqrt{5}-1}{2}$

3. פונקצית ערבול למחרוזות ארוכות:
   א. הכנס פרמוטציה אקראית מעל המספרים 0-255 למערך T[0..255]
   ב. בצע Bitwise xor בין האות הראשונה s_1 במחרוזת לבין T[0]. התוצאה a_1 נמצאת בתחום 0..255
   ג. בשלב ה-i בצע Bitwise xor בין האות Si במחרוזת לבין $T[a_{i-1}]$ כאשר a_{i-1} הוא תוצאת ה xor בשלב הקודם.
   ד. תוצאת פונקצית הערבול היא תוצאת ה xor עם האות האחרונה של מחרוזת המפתח כלומר: $hash(string)=S_n \text{ xor } a_n$

4. קבוצה אוניברסלית
   א. בחר גודל טבלת ערבול ראשוני m.
   ב. הכנס פרמוטציה אקראית מעל התחום 0..m-1 למערך T[0..r]
   ג. שבור את המפתח ל r+1 חלקים ואכסן אותו במערך key[0..r]
   ד. תוצאת פונקצית הערבול:
   $$h(x)=\left(\sum_{i=0}^{r}key[i]\cdot T[i]\right)\bmod m$$