בס"ד

מטלה 3 - מבני נתונים:

① צריך למצוא מהו זמן הריצה:

a. $T(n)=7T\left(\frac{n}{2}\right)+n^2$

$T(1)=1$

נשתמש בשיטת המאסטר:

נציב לפי הנתונים: $a=7$

$b=2$

$f=n^2$

כמות העלים:

$n^{\log_b a}=n^{\log_2 7}=n^{2.807} > n^2=f$

נוכל להבחין שאיברי הסכום הולכים וגדלים, כלומר משקל העץ מתפזר בצורה ששורש העץ הוא הקטן ורוב המשקל נמצא בעלים. כלומר: מקרה ① של שיטת המאסטר מתקיים.

לכן: אם קיים $\varepsilon>0$ עבורו $f(n)=O(n^{\log_b a-\varepsilon})$ אז $T(n)=\Theta(n^{\log_b a})$

ראינו שהעלים: $n^{2.807}$

השורש: n^2

נבחר $\varepsilon>0$ מתאים: $2.807-2=0.807$

לכן יתקיים: $f(n)=O(n^{\log_2 7-\varepsilon})$ כאשר $\boxed{0<\varepsilon\leq 0.807}$

אז $\boxed{T(n)=\Theta(n^{2.807})}$ הוא זמן ריצה.

b. $T(n)=4T\left(\frac{n}{2}\right)+n^2$

נציב את הנתונים לפי שיטת המאסטר: $a=4$, $b=2$, $f=n^2$

נחשב את העלים: $n^{\log_b a}=n^{\log_2 4}=n^2$

נשים לב שמקרה ② של השיטה מתקיים, כלומר האיזון נשמר, איברי הסכום שומרים על סדר גודל אחיד לאורך רמות העץ. גודל השורש שווה לגודל העלה, n^2.

לכן: אם $f(n)=\Theta(n^{\log_b a})$ אז $T(n)=\Theta(n^{\log_b a}\log n)$.

לפי הנתונים שלנו: $f(n)=\Theta(n^2)$ אז $\boxed{T(n)=\Theta(n^2\log n)}$ הוא זמן הריצה

בס"ד

המשך שאלה 1:

c. $T(n)= 2T(\frac{n}{3})+n^3$

שוב נשתמש בשיטת המאסטר ונציב את הנתונים:

$a=2$, $b=3$, $f=n^3$

כמות העלים: $n^{\log_b a}=n^{\log_3 2}=n^{0.63}$

נשים לב שמקרה 3 מתקיים. כלומר איברי הסכום הולכים ונעשים קטנים יותר ומשקל העץ נמצא בשורש.

לכן: אם קיים $\varepsilon>0$ עבורו $f(n)=\Omega(n^{\log_b a+\varepsilon})$ וגם קיים $c<1$ עבורו $af(n/b)\le cf(n)$ כמעט לכל n, אזי $T(n)=\Theta(f(n))$.

במקרה שלנו: נבחר ε כך: $3-0.63=2.37$ לכן נבחר $\boxed{0<\varepsilon\le 2.37}$ עבורו $\boxed{f(n)=\Omega(n^{\log_3 2+\varepsilon})}$

ניתן לבחור את ה-c.

$$2f(\frac{n}{3})=2(\frac{n}{3})^3=\frac{2}{27}n^3\le cn^3$$

ולכן נבחר: $\boxed{\frac{2}{27}\le c<1}$ וככה מתקיים שזמן הריצה הוא: $\boxed{T(n)=\Theta(n^3)}$

בס"ד

② נתון טבלת גיבוב בגודל 31.

נתון 2 פונקציות גיבוב:

1) $h_1(key) = key \bmod m$

2) $h_2(key) = \{key \bmod (m-1)\} + 1$

עלינו להכניס את האיברים הבאים לטבלה: משמאל לימין: $\overrightarrow{\{22,1,13,11,24,33,18,42,31\}}$

א) לפי מיעון מחושב $h(k)=h_1(k)$

פונקציית הגיבוב: $h(k) = k \bmod m$, גודל הטבלה: $m=31$. לכן: $\boxed{h(k) = k \bmod 31}$

נכניס את האיברים לפי הסדר שביקשו:

1) 22: $h(22) = 22 \bmod 31 = 22$

2) 1: $h(1) = 1 \bmod 31 = 1$

3) 13: $h(13) = 13 \bmod 31 = 13$

4) 11: $h(11) = 11 \bmod 31 = 11$

5) 24: $h(24) = 24 \bmod 31 = 24$

6) 33: $h(33) = 33 \bmod 31 = 2$

7) 18: $h(18) = 18 \bmod 31 = 18$

8) 42: $h(42) = 42 \bmod 31 = 11$ ⇐ תפוס! לכן נעשה רשימה בתא.

9) 31: $h(31) = 31 \bmod 31 = 0$

0	31	
1	1	
2	33	
3		
4		
5		
6		
7		
8		
9		
10		
11	11	→ $\boxed{42}$
12		
13	13	
14		
15		
16		
17		
18	18	
19		
20		
21		
22	22	
23		
24	24	
25		
26		
27		
28		
29		
30		

א) נוכל לראות בטבלה את הנתונים, הייתה רק התנגשות אחת בתא 11.

ב) אם גודל הטבלה הוא 32, 42 יעבור לתא 10, אומנם 33 יצטרך לעבור לתא 1 (שבו נמצא המספר 1) ולכן גם במקרה זה תהיה לנו התנגשות אחת.

ג) ניתן להסיק שכדאי לבחור את גודל הטבלה כך שהמספרים שנכניס אליה יהיו כולם קטנים ממנה / כך שלכל אחד תהיה שארית חלוקה שונה על מנת שלא יהיו התנגשויות.

בס"ד

המשך שאלה 2:

ב) לפי מיעון פתוח בסריקה לינארית $h(k,i)=(h_1(k)+i) \bmod m$ ← $h(k,i)=(k \bmod 31+i) \bmod 31$

עלינו להכניס את אותם האיברים לטבלה.

$\{22,1,13,11,24,33,18,42,31\}$

1) 22: $h(22,0)=(22 \bmod 31+0) \bmod 31 = 22 \bmod 31 = 22$

2) 1: $h(1,0)=(1 \bmod 31+0) \bmod 31 = 1 \bmod 31 = 1$

3) 13: $h(13,0)=(13 \bmod 31+0) \bmod 31 = 13 \bmod 31 = 13$

4) 11: $h(11,0)=(11 \bmod 31) \bmod 31 = 11$

5) 24: $h(24,0)=(24 \bmod 31) \bmod 31 = 24$

6) 33: $h(33,0)=(33 \bmod 31) \bmod 31 = 2 \bmod 31 = 2$

7) 18: $h(18,0)=(18 \bmod 31) \bmod 31 = 18$

8) 42: $h(42,0)=(42 \bmod 31) \bmod 31 = 11 \bmod 31 = 11$

נשים לב שיש לנו עכשיו התנגשות בתא 11. ולכן נעלה את i ב-1:

$h(42,1)=(42 \bmod 31+1) \bmod 31 = 12 \bmod 31 = 12$

9) 31: $h(31,0)=(31 \bmod 31) \bmod 31 = 0$

0	31
1	1
2	33
3	
4	
5	
6	
7	
8	
9	
10	
11	11
12	42
13	13
14	
15	
16	
17	
18	18
19	
20	
21	
22	22
23	
24	
25	
26	
27	
28	
29	
30	

א) סיכמנו בטבלה בצד, היה לנו התנגשות אחת במספר 42.

ב) אם הטבלה הייתה בגודל 32, 42 היה נכנס ל-10 שהוא ריק, אמנם המספר 33 יגיע לתא 1 ולכן מספר ההתנגשויות ישאר אותו דבר.

ג)

המשך שאלה ג:

ב) לפי מיעון פתוח בגיבוב כפול $h(k,i)=(h_1(k)+ih_2(k)) \bmod m$

נקבל את המערך: $\{22,1,13,11,24,33,18,42,31\}$

0	31
1	1
2	33
3	
4	
5	
6	42
7	
8	
9	
10	
11	11
12	
13	13
14	
15	
16	
17	
18	18
19	
20	
21	
22	22
23	
24	24
26	
27	
28	
29	
30	

1) 22: $h(22,0)=22 \bmod 31=22$

2) 1: $h(1,0)=1 \bmod 31=1$

3) 13: $h(13,0)=13 \bmod 31=13$

4) 11: $h(11,0)=11 \bmod 31=11$

5) 24: $h(24,0)=24 \bmod 31=24$

6) 33: $h(33,0)=33 \bmod 31=2$

7) 18: $h(18,0)=18 \bmod 31=18$

8) 42: $h(42,0)=42 \bmod 31=11$

⚠ הגענו להתנגשות! לכן נעלה את i ב-1:

$h(42,1)=((42 \bmod 31+42 \bmod 30)+1) \bmod 31=24$

⚠ שוב יש התנגשות, i=2:

$h(42,2)=((42 \bmod 31+2\cdot(12+1)) \bmod 31=$

$(11+26) \bmod 31=37 \bmod 31=\boxed{6}$

תא ריק לכן יש מקום.

9) 31: $h(31,0)=31$

ג) התנגשויות בטבלה. היתה לנו התנגשות אחת במספר 42.

ד) אם הטבלה בגודל 32, 42 ילך לתא 10. אומנם המספר 33 יתנגש עם 1 ולכן גם כאן תהיה לנו התנגשות אחת.

ה)

בס"ד

③ נתון עץ אדום שחור, כתבו אלגוריתם המקבל מספר מינ׳ ומספר מקס׳,

ומחזר כמה צמתים אדומים בעץ נמצאים בטווח הרצוי.

```
Public static int q3(int max, int min, tree(t)) {

  if(t==null) return 0;

  if(t.color = red){

    if(t.data ≥ min && t.data ≤ max){

      return 1+ q3(max, min, t.left) + q3(t.right);
    }

  }

  if(t.data > max){

    return q3(max, min, t.left);
  }
  if(t.data < min){

    return q3(max, Min, t.right);

  }

  return q3(max, min, t.left)+q3(max min, t.right);

}
```

בס"ד

4 (א) כתבו אלגוריתם יעיל המקבל כקלט מערך בגודל N ובודק האם הוא ערימת מינימום.

```
public static boolean SourA(int[] arr){
  int N = arr.length;
   for(int i = 1; i <= N/2; i++){
    if(arr[i] ≥ arr[2*i] || arr[i] ≥ arr[2*i+1])
        return false;
    }
   return true;
```

(ב) במקרה הגרוע זה $\frac{N}{2}$ כלומר שהלולאה רצה עד הסוף וזה באמת ערימת מינימום: הלולאה רצה $\frac{N}{2}$ פעמים, כלומר זמן הריצה יהיה תלוי ב-n ולכן: $\boxed{O(n)}$

בס"ד

(4) (ג) נכתוב את האלגוריתם:

```java
public static int[] qhelp(int[] arr, int i, int m) {
    int N = arr.length;
    if (i > N-1)
        return arr;
    arr[i] = arr[i] + m;
    qhelp(arr, 2i, m);
    qhelp(arr, 2i+1, m);
    return arr;
}
```

פונקציה זו היא פונקציית העזר. מרגע שסיימנו את כל האינדקסים והגענו לסוף המערך - עוצרים. (פונקציה זו היא רקורסיבית) מוסיפים למיקום ה-i: m, ואח"כ קוראים שוב לפונקציה עם הבנים של i, כלומר: 2i, 2i+1. ובסוף מחזירים את המערך.

נכתוב את הפונקציה הראשית:

```java
public static int[] q(int[] arr, int i, int m) {
    qhelp(arr, i, m);
    BuildMaxHeap(arr);
    return arr;
}
```

בפונקציה זו אנו קוראים לעזר. מפעילים אותה. ואח"כ קוראים לפונקציה Build-Max-Heap אשר מסדרת את העץ לערימה כמו שצריך

מדוע זמן הריצה הוא $O(n)$:

המקרה הכי גרוע הוא ש- $i=0$ ואז בעצם צריך לעבור על כל המערך.

כלומר נצטרך להוסיף m, לכל האיברים והפונקציה תרוץ n פעמים. (פונקציית העזר) - $\underline{\underline{O(n)}}$

לאחר שהעלנו את כל ערכי העץ ב-m אנו צריכים למיין מחדש לפי על הערימה

לפי הפונקציה - BuildMaxheap לכן גם הפונקציה הראשית היא $O(n)$.

סה"כ ביחד $T(n)=O(n)+O(n)=2\cdot O(n)$, 2 הוא קבוע ולכן זניח ולכן זמן

הריצה יהיה $\boxed{O(n)}$