טריקים ושטיקים מבנ"ת

11 ביולי 2025

1. ערימה, היא עץ בינארי כמעט שלם. אם נסתכל עליה ללא הרמה האחרונה ־ היא עץ בינארי שלם. הרמה האחרונה היא משנה. זה טוב לתכונות שאנחנו יודעים על ערימה. כלומר אם הגובה שלה הוא $logn$, אזי הערימה של $logn - 1$ הרמות הראשונות, היא עץ בינארי שלם.

2. בשאלות בהם צריך להוכיח או להפריך האם ניתן לעשות מבנה נתונים שיוציא את המינימום או המקסימום ב$O(1)$, לרוב נפריך ־ נתאר תרחיש בו נניח בשלילה שניתן, נוציא n פעמים את איבר המינימום או המקסימום ונקבל מיון שהתבצע ב$o(nlogn)$ (o קטן!) בסתירה לחסם תחתון למיון מבוסס השוואות.

3. טוב לזכור ־ אם נסרוק $in-Order$ עץ כלשהו, נקבל מיון. זה עולה $O(n)$ לבצע את ההדפסה. ניתן לעשות גם בעץ 2־3 אך יש להסביר מדוע ניתן לעשות זאת (הרעיון יהיה דומה לסריקה המקורית רק שנעבור בין הילד השמאלי לאבא השמאלי, בן אמצעי, אבא ימני, ילד ימני וכו').

4. אלגוריתם סלקט די יעיל. ניתן באמצעותו למצוא את k האיברים הקטנים במערך. נפעיל אותו על האיבר הk ב$O(n)$, ואז נסרוק את המערך ונבדוק האם האיבר קטן מהאיבר שמצאנו, אם כן נדפיס אותו, כך מצאנו את k הקטנים. כמו כן ־ לאחר שמבצעים סלקט, מקבלים מהצד הימני לk את כל מי שגדול מk (לא באופן ממויין) ומשמאל את כל מי שקטן מk (שוב, לא באופן ממויין).

5. ניתן "למיין מטריצה". פשוט נשטח אותה להיות מערך n^2 (ממש שורה שורה לפי הסדר) ואז נבצע מיון שיעלה $n^2logn^2 = 2n^2logn = O(n^2logn)$.

6. נשתמש ב$UnionFind$ כאשר נהיה בבעיות אילוצים ותמיד בקורס הזה כאשר נראה דרישה לlog^*n. נזכור שמאפשר $find$ ב$O(log^*n)$ וכן $union$ ו$makeSet$ ב$O(1)$.

7. טריק יפה על מערך ממויין ־ ניתן לרוץ עם שני אינדקסים כך שאחד יהיה מאותחל לאינדקס הראשון, השני לאחרון. נניח ונרצה לחפש זוג איברים שההפרש ביניהם הוא מס' מסויים. נחשב את ההפרש על שני האינדקסים, אם ההפרש שיצא גדול ממה שנרצה, נרצה להקטין את הקלט לכן נקטין את האינדקס הגבוה. אם ההפרש יוצא קטן ממה שצריך נגדיל את האינדקס הנמוך. סה"כ יעלה $O(n)$ שכן מעבר לינארי על n איברים.

8. נניח ונתקל בדרישה לשאילתא־ בכל רגע נתון נרצה להחזיר איבר שיהיה קטן מלפחות $\frac{n}{4}$ מהאיברים ושיהיה גדול מ$\frac{n}{4}$ איברים, כלומר שיהיה באחוזון $75 - 25$. נוכל למצוא את החציון במערך בכל $\frac{n}{4}$ פעולות, במקרה הגרוע אם כל הפעולות היו מחיקה החציון יהפוך לאחוזון 25, אם כל הפעולות היו הכנסה החציון יהפוך לאחוזון 75. לשיעורין זה יתבצע ב$O(1)$. מדוע? עלות הפעולה של מציאת החציון היא n לפי סלקט. כל כמה פעמים זה קורה? $\frac{n}{4}$. לכן לשיעורין, $C_i = \frac{\hat{n}}{\frac{n}{4}} = 4$.

כלומר, לשיעורין זה יעלה 4 מטבעות. כלומר - $O(1)$.

9. כדאי לזכור את הטור הבא שפחות טריוואלי: $\sum_{i=1}^{n}\frac{1}{i}=ln(n)+O(1)$

10. ניתן לחפש במגוון דרכים. $O(n)$ רגיל, במערך ממויין ניתן לבצע חיפוש בינארי ב$O(logn)$ ע"י בדיקת הערך האמצעי בכל שלב וללכת לאזור המתאים, וניתן לבצע במערך ממויין חיפוש של הערך x במערך בצורה אקספוננציאלית בקפיצות של 2, עד שנגיע למצב $2^i<x<2^{i+1}$, שם נחפש ב$log(x)$, וסה"כ זה עלה $O(logx)$.

11. אם יש לי מערך ממויין, ניתן ליצור ממנו עץ AVL ב$O(n)$, נבחר בכל שלב את האמצעי, ונפעיל רקורסיבית את הבנייה על שני החלקים השונים של המערך וכך נבנה עץ AVL.

12. מציאת m האיברים הגדולים מתוך ערימת מקסימום בעלת n איברים, $O(mlog(m))$. ראשית נעביר את השורש לערימת עזר. בכל פעם נוציא את השורש של ערימת העזר, נוסיף אליה את שני הבנים של השורש מהערימה המקורית . שוב, נוציא את שורש ערימת העזר (אחד משני הבנים שזה הרגע נוספו), ונוסיף לערימת העזר את שני ה בנים שלו מהערימה המקורית. נעצור לאחר שהוצאנו m איברים. נשים לב כי בכל רגע נתון אין בערימת העזר יותר מ־m איברים לכן כל הוצאה תיקח $log(m)$ ואנו מבצעים סה"כ m הוצאות.

13. מיון שלא מבוסס השוואות. נזכור שבמיון בסיס, אם נתון איברים מהטווח $\{0,1,...R^d-1\}$, אזי R הוא הבסיס שעובדים בו, וכן d זה אורך המספרים הכי גדול. אם נבחר את הבסיס להיות n, נראה כי אם בהינתן שהמספרים בתחום $[0,n^4]$, אזי $d=log_n(n^4)=4$, כלומר אורך המספר הכי גדול יהיה באורך 4, ואז $O(d(n+R))=O(4(n+n))=O(8n)=O(n)$.

14. ראיתי בעבר במבחנים שממש שאלו מה זמן החיפוש של איבר בטבלת האש. אם הוא כבר קיים אזי $\Theta(\frac{1}{a}ln(\frac{1}{1-a}))$, אם לא נמצא (הזמן הממוצע להכנסת איבר יוניפורם האשינג) אזי $\Theta(\frac{1}{1-a})$.

15. מה אומרת דרישת סיבוכיות $log(logn)$? לרוב, מדובר על חיפוש בינארי ב$logn$ איברים. ניתן לנסות לשלב חיפוש אקספוננציאלי ובינארי יחד, ולהגיע לדרוש.

16. לשים לב, לפעמים כאשר עובדים עם איברים $a_1,...a_n$ גם אם הקלט אינו ממויין, אפשר להחליט שנבנה מבנה לפי האינדקסים $1,...n$ ואז ניתן לבנות מבנה כמו עץ חיפוש ב$O(n)$, ולחפש לפי i. יש לשים לב שהחיפוש לא יתבצע לפי הערכים אלא לפי האינדקסים. לפעמים זה שימושי.

17. כשיש לך איברים שכל אחד שייך לבדיוק קבוצה אחת, אתה יכול לעבור עליהם פעם אחת בלבד על ידי "מחיקה אחרי עיבוד".

18. **טריקים לתכנון דינמי:** יש פונקציות רגילות עם i,j ויש פונקציות שניתן להגדיר כמו כך -

$$f(i)=\ \vee_{1\leq j\leq i}\{f(j)\}\quad 15\leq S[i]-S[j]\leq 25$$

פונקציות כאלו הן לכאורה נשמרות במערך פשוט אך מילוי כל תא בהם הוא $O(n)$ שכן בודקים את כל הקודמים. זו טכניקה טובה.

טכניקה נוספת לתכנון דינמי היא הגדרת שלוש פונקציות למשל כמו בדוגמה הבאה:

$$\begin{aligned} f(0,j) = &\ max\{f(0,j-1), f(1,j-1), f(2,j-1)\} \\ f(1,j) = &\ A[1,j] + max\{f(0,j-1), f(2,j-1)\} \\ f(2,j) = &\ A[2,j] + max\{f(0,j-1), f(1,j-1)\} \end{aligned}$$

כאשר הפתרון יהיה $max\{f(0,n), f(1,n), f(2,n)$, כלומר היו לנו 3 מקרים שונים ולכן פיצלנו לשלוש פונקציות.

לרוב כאשר נדרש לתת סדרה נגדיר $f(i,j)$ כפתרון אופטימלי לתת הסדרה של האיברים $i,...j$. ואז הפתרון יהיה ב$f(1,n)$. כאשר נדבר על סדרה רציפה כן ניתן לפעמים להעזר ב$f(i)$ יחיד. כמו כן אפשר לשים הגדרה $f(i)$ יהיה האיברים $1,...i$ כאשר בוחרים את i כאיבר אחרון. ואז הפתרון יהיה $max_{1\leq j\leq n}\{f(j)\}$.

19. לשים לב! אם עושים עיבוד מקדים של מערך תתי הסכומים כלומר $B[i]=\sum_{k=1}^{i}A[k]$ אזי אם נרצה להחזיר את סכום תת המערך $A[i],...A[j]$ מחזירים את $B[j]-B[i-1]$.

20. **טריקים על** AVL: כבר אמרנו שניתן ומומלץ להוסיף שדות $size$ וsum לתת העץ המושרש. כעת, נרצה להסביר כיצד מחשבים שתי פעולות חשובות:

א. $rank$ - כמה איברים קטנים ממני בעץ? נניח שהשורש הוא 5. אני רוצה לחשב כמה איברים קטנים מ20. שמרתי שדה $sizeLeft$ ומתקיים $size(root)=10$. אזי בוודאות עשר קטנים מ20. נקח את המספר הזה ונלך רקורסיבית לחלק הימני. כך נמשיך עד שנמצא את 20 או איבר שגדול ממנו. זה יעלה כגובה העץ $O(logn)$

ב. $sum(k)$ - מה סכום האיברים של המפתחות שקטנים מk? באופן דומה ל$rank$, רק שהפעם נשמור שדה sum ואותו נוסיף. גם זה כגובה העץ $O(logn)$

21. מומלץ לעיתים במקום שדה בavl להשתמש במצביע. למשל - מצביע לאיבר המינימלי.

22. כדאי בפעולות של לשיוערין להבין שלא חובה למחוק איברים מהמחסנית

23. כאשר יש לנו בעיות של שני תתי מחרוזות T וS מומלץ להגדיר פונקציה $f(i,j)$ שתטפל ב$1,...i$ איברים של S ו$1,..j$ איברים של T.

24. **טריקים של עיבוד מקדים:** בשאלות מתורת המשחקים בהם המטרה היא "לדפוק את השחקן השני" ולא בהכרח להשיג מקסימום. יתכן שנקבל נוסחת נסיגה כזו:

$$f_1(i,j) := \left\{ \begin{array}{cc} v_i & i=j \\ \sum_{k=i}^{j} v_k - min\{f_2(i+1,j), f_2(i,j-1)\} & i>j \end{array} \right\}$$

$$f_2(i,j) := \left\{ \begin{array}{cc} v_i & i=j \\ \sum_{k=i}^{j} v_k - min\{f_1(i+1,j), f_1(i,j-1)\} & i>j \end{array} \right\}$$

לגיטימי ויפה. נראה כי ה$\sum$ הזה יקר מאוד לחישוב. מה שנרצה לעשות לרוב יהיה ליצור מערך סכומים B כך ש$B[i]=\sum_{k=1}^{i}A[k]$ ואז לחשב את הסיגמה שם יתבצע בקלות עם שליפה מהמערך. כלומר $B[j]-B[i-1]$. חישוב העיבוד המקדים יעלה $O(n)$ בלבד.

25. **מציאת עוקב בעץ** AVL- נוכל למצוא את העוקב בזמן $O(logn)$ של איבר מסויים. כלומר, האיבר הבא אחריו ביחס סדר. ראשית נחפש את הצומת x.
מקרה ראשון - יש לצומת x תת עץ ימני. במקרה זה, העוקב יהיה הקטן ביותר בתת העץ הימני.
מקרה שני - אין לצומת x תת עץ ימני, אזי אנחנו נטפס למעלה בעץ כל עוד אנחנו בן ימני, ונחזיר את ההורה הראשון שאנחנו נהיה בן שמאלי שלו.
יש לשים לב אם האיבר הוא הגדול ביותר בעץ אין לו עוקב.
מציאת קודם בעץ -
אם יש לצומת תת עץ שמאלי, הקודם יהיה הכי גדול בתת העץ השמאלי.
אם אין, טפס למעלה בעץ כל עוד אתה בן שמאלי ותחזיר הורה ראשון שאתה בן ימני שלו.

26. איחוד ערימות - לא יעיל אך אם נדרש, מעתיקים את איברי הערימות עם סריקה כלשהי לתוך מערך בגודל $|H_1|+|H_2|$, ואז מבצעים $heapify$ ויוצרים ערימה מהמערך החדש בעלות $O(|H_1|+|H_2|)$.

27. איחוד עצי AVL: מבצעים סריקת $in-order$ על העצים T_1, T_2 לקבלת שני מערכים ממויינים. זה עלה $n+m$. יוצרים מערך ממויין חדש בגודל $n+m$, בעלות $O(n+m)$ (בכל שלב בודקים מי הכי קטן מהראשון משמאל במערך ומקדמים את האינדקס במידת הצורך), ואז יוצרים עץ AVL חדש ברקורסיה מהאיבר האמצעי. סה"כ יעלה $O(n+m)$.

28. **שימו לב וזה חשוב מאוד -** ניתן לגרום להוצאת איבר מערימה וחיפוש בה ב$O(logn)$- אנחנו ניצור טבלת האש ובה פוינטרים דו כיוניים לאיברים בערימה. סה"כ בניית הערימה עם הטבלה עלתה $O(n)$ זמן ומקום. כעת, כאשר נרצה למחוק למשל איבר מהערימה נחפש אותו בתוך טבלת האש, עם הפוינטר "נשתגר" אל מיקומו בערימה, נחליף אותו עם האיבר הימני ביותר ברמה התחתונה ונבצע פעפועים כמו שמבצעים למחיקת מינימום. סה"כ קבועים +פעפוע כגובה העץ ולכן $O(logn)$.

29. האלגוריתם $YoungTableau$: נניח שנתון בשאלה שאיברי מטריצה ממויינים בשורות ועמודות - כלומר כל שורה ממוינת בעצמה, וכל עמודה ממוינת בעצמה. ונרצה למצוא את איבר k כאשר מימדי המטריצה $n \times m$. מה שנעשה - נסתכל על האיבר $A[1,m]$. כעת נבדוק אם $A[i,j]>k$ נרצה לזוז שמאלה תא אחד במערך (כי בהכרח הוא יהיה לפני. אם $A[i,j]<k$ אזי הוא עתיד להיות באחד מהשורות למטה באותה עמודה ולכן רד למטה, אחרת $A[i,j]=k$ מצאת את k. סה"כ מעבר לינארי על $n+m$ איברים במקרה הגרוע ולכן $O(n+m)$.

30. מס' העצים בינאריים מגודל n קודקודים - תופתעו או שלא, הוא מס' קטלן הn-י (קל לראות). שהוא כמובן $C_n = \frac{1}{n+1}\binom{2n}{n}$.

31. הוכח הפרך - נניח שבקוד הופמן כל האותיות בתדירות קטנה מ$\frac{1}{3}$ מאורך הטקסט, אזי לא ייתכן שיש מילה באורך אחד. פתרון: אם יש מילה באורך אחד, אזי היא חוברה בעץ לסכום כל התדירויות האחרות. זה כמובן לא ייתכן לאור הנתון על שליש ולכן הטענה נכונה.

נניח שבקלט לאלגוריתם הופמן קיים תו שתדירותו היא יותר מ$\frac{2}{5}$. אזי חייבת להיות מילת קוד באורך אחד.
הוכחה: יש לנו תו s עם תדירות$>\frac{2}{5}$. כל שאר התווים יחד, סכום התדירויות שלהם יהיה קטן מ$\frac{3}{5}$. נוכיח שs יקבל מילת קוד באורך 1.
נב"ש כי c יקבל מילת קוד באורך $c \geq 2$. באלגוריתם הופמן מיזוג קורה רק בין שני הצמתים עם התדירות הנמוכה ביותר. בכל שלב של האלגוריתם $f(s) > \frac{2}{5}$, וכן סכום התדירויות $\sum -f(s) < \frac{3}{5}$.

לכן כל צומת אחר (בודד או ממוזג) תהיה מקסימלית עם תדירות $>\frac{2}{5}$. אם $f(s) > \frac{2}{5}$ וכל צומת אחר קטנה מ$\frac{3}{5}$ אזי באף שלב דוכל חיבור אחר לא יוכלו להיות השניים הקטנים ביותר. לכן s לא יתמזג עד הסוף ולכן עומקו יהיה 1 כלומר תהיה מילת קוד באורך 1. הסבר: בשביל שs יתמזג עם מישהו, צריך שתהיה צומת אחרת שהתדירות שלו תהיה פחותה או שווה לתידורת של s אבל כל צומת אחר מורכב מערכים עם סך תדירויות קטן משלוש חמישיות ולכן אפילו אם נחבר את שני הצמתים הכבדים ביותר שאינם s התוצאה תהיה $\frac{3}{5}$. סה"כ התו לא יתמזג עם עוד איבר נוסף ולכן ישאר בודד כלומר מילה באורך 1.

32. לא"ב נתון ולשכיחויות מסוימות קיים יותר מקידוד אופטימלי אחד - **אם יש למשל שני תווים עם תדירות זהה.**

33. לשאלות של בנה מבנה נתונים ש - לזכור להשתמש לפעמים בעץ avl שמכיל מצביעים לעצי avl שונים.

34. כל פעם שרואים "הגדולים ביותר" / "הקטנים ביותר" - סלקט!!!

35. בשאלות עם מערכים כדאי לשחק עם פוינטרים - שניים להתחלה ומשווים, אחד להתחלה של אחד והשני לסוף של השני ומשחקים כאלו זה לרוב יתן $O(n)$ ריצה.

36. **זכור** - $T(n) = 2T(\frac{n}{2}) + O(n)$ **מקיים** $T(n) = O(nlogn)$.

37. כתוב אלגוריתם לינארי שבודק אם העץ הבינארי הנתון הוא עץ חיפוש - עבוד בהפרד ומשול רקורסיבית. עליך לדאוג כי במעבר שמאלה בעץ כל האיברים יהיו קטנים מהאבות הקדמונים שלהם ולכן עבור כל איבר נשמור מינימום ומקסימום טווח ערכים בהם הוא יכול להיות ככל שנתקדם בעץ נקבע את הערכים הללו. כאשר נלך שמאלה למשל נבדוק רקורסיבית את $(left, min, nodeVal)$ כי ה$nodeVal$ יהפוך למקסימום עבורו. כאשר נלך ימינה נבדוק רקורסיבית את $(right, nodeVal, max)$ שכאן הערך שהיינו בו יהפוך למינימום שהוא יכול להיות בו כרגע. סה"כ מעבר לינארי על n איברים ולכן $O(n)$

38. כתוב אלגוריתם לינארי שבודק אם העץ הבינארי הנתון הוא עץ AVL- נכתוב אלגוריתם לינארי רקורסיבי. נרצה לבדוק בכל ביקור בצומת שלנו האם $|H_L - H_R| \le 1$. כעת ניגש רקורסיבית , נבדוק בכל שלב שצד ימין וצד שמאל הם עץ AVL ונתקדם רקורסיבית עד שנגיע לעלים. בעלים נחשב $h = max\{leftH, rightH) + 1$ ומקדם האיזון יהיה $k = leftH - rightH$. סה"כ אם נקבל כי $|k| > 1$ אנחנו נחזיר שלא יתכן שזה עץ avl כלומר שקר. אחרת, רקורסיבית נפתור את הבעיה על העץ משמאל ומימין. סה"כ ביקור בכל העץ שיעלה $O(n)$.

39. אם אין חשיבות לסדר המילוי בתכנון דינמי - עמודות או שורות, וכן המטריצה מגודל $n \times m$ כאשר $n \neq m$ אזי בחר למלא בהתאם ל$O(min\{n, m\})$.

40. אתה רואה $logn$? זה לא בהכרח עצים או ערימה!!!!! זכור שיש יוניון פיינד - יש פתרון שמאפשר $uninon, find$ ב$O(logn)$ ו$makeset$ ב$O(1)$. **חשוב!!!!**

41. לשים לב - סימנו לך $A[1, ...n]$? זה סימון שאומר קיבלת מערך עם n איברים. אמרו לך יש לך מבנה A עם המפתחות $1, ...n$? זה משהו שונה לגמרי! אתה מחפש לפי המפתחות האלו. אם בחרת לייצג אותם במערך למשל לפי האינדקס שלהם הם כבר ממויינים!

42. ניתן לנתח לשיעורין הוצאה של איבר מעץ AVL בזמן אמורטייזד $O(1)$, אם נקצה לכל איבר

בהכנסה $2logn$ מטבעות. $logn$ לשימוש מיידי ו$logn$ לשימוש עתידי במחיקה.

43. בהינתן ערימת מינימום נרצה להחזיר את k האיברים הקטנים ביותר מבלי לפגוע במבנה - $O(klogk)$, ניצור ערימת עזר, בתחילה נקח את איבר המינימום מערימה מקורית ונכניס אותו. אח"כ נכניס לערימת עזר את בניו וכן הלאה בכל שלב בערימת העזר יהיו לכל היותר k איברים ולכן הוצאתו ממנה $logk$ כפול k הכנסות והוצאות ב$O(klogk)$.

44. אם נסמן $k = \frac{n}{logn}$ נקבל $O(klogk) = O(n)$.

45. נתונה סדרת איברים $a_1, ..., a_n$ שונים זה מזה. לכל i מתקיים a_i הוא חציון איברי הסדרה עד אליו. נניח שבמקרה וכמות האיברים זוגית אזי החציון יהיה החציון התחתון למשל החציון של קבוצה בגודל 4 יהיה ה2 בגודלו. כיצד נמיין את המערך הזה? בהכרח a_n הוא איבר החציון של המערך. כעת נעבור לa_{n-1} שהוא החציון של מי שנשארו, אם הוא גדול מa_n נשים אותו מימינו ואם קטן משמאלו לפי יחס הסדר בניהם וכן הלאה נמשיך עם האיברים עד שנגיע לאיבר הa_1. סה"כ מעבר לינארי על n איברים ולכן $O(n)$

46. נניח ונותנים לנו עץ בינארי כמעט שלם וצריך למצוא את האיבר ברמה התחתונה מימין שהוא האחרון ב$O(log^2n)$: אלגוריתם לא טריוואלי ולכן כדאי לזכור - נבצע חיפוש בינארי על רוחב תחתית הערימה. בכל שלב נתחיל מהשורש הנוכחי ונשווה את אורך שני המסלולים הבאים: כל הדרך פניה שמאלה, לבין צעד אחד ימינה ואז כל הדרך שמאלה. אם הם באותו אורך נמשיך בתת עץ ימני ואם לא אז נמשיך בשמאלי. סה"כ גובה הערימה $O(logn)$ ונבצע $O(logn)$ איטרציות עד למציאתו ולכן $O(log^2n)$. הסבר - אם שני מסלולים באותו אורך אזי רמה תחתונה היא מלאה ולכן צריך להמשיך בתת עץ ימני, אם מסלול ימין שמאל קצר יותר אזי הרמה התחתונה לא מלאה בתת העץ הימני ולכן תמשיך בתת העץ השמאלי.

47. **לא לשכוח - יש רשימה מקושרת!!**
פשוטה - הוספה בתחילת רשימה $O(1)$, הוספה בסוף $O(n)$, הוספה ומחיקה אחרי איבר ידוע $O(1)$, חיפוש $O(n)$, מחיקה לפי ערך כלשהו $O(n)$
דו כיוונית - כמו קודם רק הוספה בסוף ב$O(1)$ ומעבר קדימה ואחורה ברשימה ב$O(1)$

48. לשים לב בתכנון דינמי בשאלות כמו תרמיל הגב בשלמים לבדוק תנאים כמו $b < w_j$ שכן ייתכן שככלל לא נוכל תמיד לקחת את האיבר.

49. תכנון דינמי - לשים לב במחרוזות שיש בעיות שהן מצא סדרה הכי גדולה, ויש בעיות שהן מצא תת סדרה מינימלית שמוכלת בשני תתי סדרות קודמות. במקרה כזה נוסחת הנסיגה תהיה כמו שמתואר מטה, נראה כי תנאי הבסיס מעניינים. אם $i = 0$ וכן $j > 0$ אזי בהכרח אורך תת הסדרה הקצר ביותר שמכיל את שתיהן הוא j כי מכיל סדרה ריקה באופן ריק

$$f(i,j) = \left\{ \begin{array}{cc} 0 & i = 0 \wedge j = 0 \\ 1 + f(i-1, j-1) & T[j] = S[i] \\ min\{f(i, j-1) + 1, f(i-1, j) + 1\} & o.w \\ j & i = 0 \wedge j > 0 \\ i & j = 0 \wedge i > 0 \end{array} \right\}$$

50. אתה רואה בעיות מתמטיות או בעיות חלוקה? לרוב השלב לפתרון יהיה קודם להגיע

לתובנה מתמטית!!

51. הם מתים על בעיות של הפרד ומשול בסגנון: לך על אמצע, חשב מס' אפשרויות מימין, משמאל, וכאלו שמתחילים משמאל ונגמרים בימין וצרף. כמובן החלק האחרון עולה $O(n)$. הנה פסודו שמדגים למשל מס' היפוכים במערך. חשוב להדגיש בשאלות כאלו ממש דומות למרג' סורט וכאן בדוגמה מטה שינינו דבר אחד בדיוק במרג' סורט - זה הבסיס לשאלות כאלו וזה חשוב.

```
function countInversions(arr, left, right):
  if left >= right:
    return 0
mid = (left + right) / 2
  leftInv = countInversions(arr, left, mid)
  rightInv = countInversions(arr, mid+1, right)
  crossInv = mergeAndCount(arr, left, mid, right)
  return leftInv + rightInv + crossInv

function mergeAndCount(arr, left, mid, right):
  leftArr = arr[left...mid]
  rightArr = arr[mid+1...right]
  invCount = 0
  i = 0, j = 0, k = left
  while i < leftArr.length AND j < rightArr.length:
    if leftArr[i] <= rightArr[j]:
      arr[k] = leftArr[i]
      i++
    else:
      arr[k] = rightArr[j]
      invCount += (leftArr.length - i)
      j++
    k++

  // העתק שאר איברים...

  return invCount
```

52. בהמשך לסעיף הקודם = בעיית תת מערך הכבד ביותר היא חשובה ומופיעה בנוסחים שונים. יש את האפשרות הנאיבית לכאורה ב$O(nlogn)$ כאשר בודקים תת מערך מימין שמאל ואמצעי. ויש אפשרות ב$O(n)$ בלבד אם נפעל בהפרד ומשול ונחזיר ברקורסיה בכל שלב את הדברים הבאים: רישא, סיפא וסך כל האיברים ואז נוכל לחשב סיפא ורישא ב$O(1)$ ונוסחת הנסיגה תהיה $T(n) = 2T(\frac{n}{2}) + O(1)$.

53. כדאי לזכור החלפת משתנים לפתרון נוסחת נסיגה כמו זו - $T(n) = 4T(\sqrt{n}) + log^2 n$. נבצע החלפת משתנים. נסמן $m = log_2 n \iff 2^m = n$. ולכן שקול ל $T(2^m) = 4T(2^{\frac{m}{2}}) + m^2$. כעת נציב $s(m) = T(2^m)$ ולכן $s(m) = 4s(\frac{m}{2}) + m^2$ נוסחה זו אנחנו כבר יודעים לפתור בקלות עם מאסטר.

54. עוד תכנון דינמי- אם נרצה לעבור על מחרוזת למשל כדאי לנקוט בגישה הבאה: אם משהו טוב לך צמצם את טווח המחרוזת בשני הצדדים ותוסיף אחד לקלט, אחרת תעבוד על תתי הבעיות שבפנים כלומר על k שבטווח ותעשה אותם נקודת חלוקה של המקטע. סיבוכיות נוסחה כמו כאן בדינמי תצא $O(n^3)$:

$$f(i,j) := \left\{ \begin{array}{cc} 0 & i \geq j \\ 1 + f(i+1, j-1) & S[i], S[j] \in AU, UA, CG, GC\}\} \\ max_{1 \leq k \leq j-1}\{f(i,k) + f(k+1,j)\} & o.w \end{array} \right\}$$

55. זכור ! במערך ממויין תוכל לחפש בקלות ב$O(logn)$.

56. מיזוג שתי ערימות מינימום בגדלים m,n המיוצגות בעצים: נוסיף ערך חדש שיהיה אנסוף,

הבן השמאלי של קודקוד זה יהיה תת ערימה אחת בגודל m והבן הימני בגודל n. נחזיק מצביע לאיבר אנסוף. כעת נבצע פעולת $heapify$ רגילה ואז נמחק ערך זה. לאחר פעולת $heapify$ האיבר המינימלי מבין המינימליים בערימות יעלה אל השורש, בנו השמאלי של השורש יהיה אחד מבניו הקודמים וכן גם מימין יבחר המינימלי מבין השורשרים לעלות לשורש החדש, לכן בנו הימני גם יהיה גדול ממנו. סה"כ פעולת $heapify$ תעלה $max\{logm, logn\} = O(log(m+n))$.

57. יש שאלות תכנון דינמי שהן "בחירת אפשרויות". בשאלות כאלו אנחנו לא נבצע מקסום ואופטימיזציה אלא ממש סכימת אפשרויות קיימות. למשל - מס' אפשרויות לסדרה של n הטלות מטבע כאשר אסור פעמיים ברצף עץ. נוכל להגדיר $f(i,j)$ כמטבעות ה$1,....i$ כאשר j הוא **הבחירה הנוכחית שלי.** כלומר אם נסמן 1 פלי ו0 עץ אזי $j \in \{0,1\}$. כעת קל לחשב נוסחת נסיגה שתהיה

$$f(i,j) := \left\{ \begin{array}{cc} 1 & i=1 \\ f(i-1,1) & i>1 \wedge j=0 \\ f(i-1,1)+f(i-1,0) & i>1 \wedge j=1 \end{array} \right\}$$

כלומר בכל רגע נתון יש שני מסלולים, אם יש לי עץ כרגע בבחירה אזי אני רוצה רק את מס' האפשרויות לקודם לכן בהם בחרתי פלי, אחרת נחבר את האפשרויות כאשר מקודם לקחתי פלי או עץ. סה"כ נקבל פתרון ב$O(n)$ שכן טבלאות כאן אינן באמת דו מימדיות שעולות $O(n^2)$ אלא $n \times 2 = O(2n) = O(n)$. הפתרון כמובן יהיה סכום האפשרויות לבוא לסוף עם 0 או עם 1 כלומר $f(n,0)+f(n,1)$.

מה שחשוב בשאלה הזו הוא להבין שניתן לבחור מס' אפשרויות להגדרת פונקציה

א. מסתכלים על $f(i,...j)$ כתת סדרה והפתרון $f(1,n)$.

ב. מסתכלים על $f(i)$ כאשר i הוא איבר אחרון שבחרתי ולבסוף עושים $max_{1\leq i\leq n}\{f(i)\}$

ג. מגדירים $f(i,j)$ כאשר j הייתה בחירה אחרונה שביצעתי בסבב.

ד. מגדירים $f(i,j)$ ועוברים בנוסחה הרקורסיבית על $max_{i\leq k\leq j}\{f(k)\}$ שעושים משהו.....

ה. תורת המשחקים - מגדירים $f(i,j)$ והפתרון הוא הרווח פחות מינימזציה של סכום החבר השני.

ו. הרבה בעיות ניתן להמיר לסדרת פיבונאצ'י. הבעיה שכאן למשל יכולה להיות בדיוק לפי נוסחת פיבונאצ'י. אם סדרה מסתיימת בפלי היא יכולה לבוא מכל סדרה תקינה באורך $n-1$ ואם סדרה מסתיימת בעץ היא יכולה לבוא מכל סדרה תקינה באורך $n-2$.

ז. כדאי להשתמש בהגדרה של $state$ כלשהו למשל אם אני רוצה למצוא סדרת זיג־זג מקסימלית, כדאי להגדיר משתנה $state$ שיהיה $0,1$ ויציין בהתאמה עליה או ירידה, והוא יעזור לי לדעת מה המצב שכעת אני צריך להיות בו. אם זה תת סדרה זיג־זג רציפה פשוט נעבור על כל האפשרויות $_{i\leq k\leq j}\{f(k,state)\}$ כאשר נוסיף תנאים כמובן לבדיקה של ה$state$ שנשמר.

58. מציאת משהו בקטע $[a,b]$ בעץ AVL. בשיעורי הבית הייתה שאלה לבנות מבנה נתונים שבהינתן שני מפתחות מחזיר ממוצע מקומי של הערכים שבטווח בין שני המפתחות. השאלה הזו חוזרת בנוסחים שונים במבחנים, למשל עם מציאת המקסימום המקומי (הערך, לא המפתח) בתחום. הנה אלגוריתם שיפתור זאת ע"י הוספת שדות מינימום ומקסימום של תת העץ (מה שקל לתחזוק ב$O(1)$): יהי r שורש העץ.

אם $a > key(r)$ נלך ימינה כי גם בהכרח b שם. אם $b < key(r)$ נלך שמאלה כי גם בהכרח a קטן משורש העץ. אחרת, כלומר $a \leq key(r) \leq b$ כלומר המפתחות שלנו נמצאים בין קצוות העץ.

אם נקבל שתת עץ שלם נמצא בתוך הטווח שלנו, נחזיר את המקסימום שלו - הערך ששימרנו. הוא המקסימום הלוקלי. אחרת, אם לא כל תת העץ בטווח, נרד לחפש רקורסיבית בתתי העצים.

א. תת עץ שמאלי -

1. אם מקסימום תת העץ השמאלי קטן מa דלג על תת העץ השמאלי לגמרי.

2. אם כל תת העץ השמאלי ב$[a,b]$ קח את המקסימום שלו.
3. תמשיך רקורסיבית על תת העץ השמאלי
ב. תת עץ ימני
1. אם המינימום של תת העץ הימני גדול מb, דלג עליו לגמרי.
2. אם כל תת העץ הימני ב$[a,b]$ קח את המקסימום שלו
3. אחרת תמשיך רקורסיבית על תת העץ הימני.
התוצאה תהיה המקסימום המקומי שהוחזר מהרקורסיה. סה"כ רקורסיה $O(logn)$ על גובה העץ

59. **אלגוריתמים חמדניים:**
אני בספק גדול שיהיה = יש יותר מדי חומר. אבל במידה ואכן יהיה, זה לא כזה מסובך:
א. לרוב בוחרים למיין למשל לפי זמני סיום, ולוקחים כל פעם את זה שיהיה מינימלי.
ב. בבעיית תרמיל הגב בשלמים מחשבים את הערך שלו v_i ביחס למשקל שלו w_i כלומר $\frac{v_i}{w_i}$, וממיינים ואז יודעים את מי לקחת בכל שלב. זה נכון להרבה בעיות אחרות - אני ממיין את האיברים לפי כמה הם יתרמו לי ביחס למשקלם. (או לזמן שלהם)
ג. ההוכחת נכונות נראית אותו דבר. לזכור את הדוגמה עם המקטעים בכביש (תרגול ביוטיוב של צביקה) וזה אחד לאחד.

60. כדאי להכיר את הבעיה הבאה - בהינתן מערך עם מספרים ממשיים, ומס' k האם קיים תת מערך רציף שסכומו k. ניתן לפתור בתכנון דינמי $O(nk)$. ניתן גם ב$O(n)$ כך:
נבנה מערך סכומים חלקיים $S[i]=\sum_{k=1}^{i}A[k]$ ב$O(n)$ בצורה רקורסיבית עם התא הקודם. כעת, עבור כל ערך i נסתכל על $A[i]$. נראה כי אם קיים ערך k שכזה, הוא באינדקסים $i,..j$. וסכומו יהיה $S[i]-S[j-1]$. כעת עבור כל i, נרצה לבדוק אם קיים j עבורו $S[i]-S[j-1]=k$. הבעיה הומרה ל$2SUM$ קלאסית שכמובן נפתרת ב$O(n)$, $O(1)$ לכל איבר עם חיפוש בהאש.

תת מערך רציף כבד ביותר: נבנה מערך סכומים חלקיים $S[i]=\sum_{k=1}^{i}A[k]$ ב$O(n)$ בצורה רקורסיבית עם התא הקודם. תת מערך מקסימלי הוא באינדקסים $i,..j$. וסכומו יהיה $S[j]-S[i-1]$. נרצה לחפש שני אינדקסים עבורם $S[j]$ יהיה מקסימלי, וכן $S[i-1]$ יהיה מינימלי. נסרוק ב$O(n)$ ולאחר שמצאנו אותם קיבלנו את האינדקסים שממקסמים את הסכום.

61. כדאי להכיר כי חוזר על עצמו - תת סדרה משותפת ארוכה ביותר

$$F(i,j)=\left\{\begin{array}{cc} 0 & i=0\vee j=0 \\ F(i-1,j-1)+1 & x_i=y_j \\ max\{F(i-1,j),F(i,j-1)\} & otherwise \end{array}\right\}$$

62. בהינתן שתי קבוצות A,B. נרצה לבדוק האם $A\cap B=\emptyset$. נוכל להמיר את השאלה גם - האם בהינתן שני עצי AVL יש ערך שמשותף לשני העצים?
נתחזק 3 עצי AVL: אחד לאיברי A, אחד לאיברי B ואחד לכפולים. בעת כניסת איבר לקבוצה A, נבדוק אם הוא קיים בקבוצה B. אם כן? נכניס אותו אל עץ הכפולים. באופן דומה בעת כניסה לאיבר B. כאשר נמחק איבר, נבדוק האם הוא קיים גם בקבוצה השניה, אם כן נמחק אותו מהקבוצה שלו וכן מעץ הכפולים - כי אין כפילות עוד. כאשר נרצה לבדוק האם קיים איבר בשני העצים - נבצע חיפוש על עץ הכפולים ב$O(logn)$

63. מציאת חציון במערכים ממויינים בגודל זהה: נזכר בנוסחת הנסיגה $T(n)=T(\frac{n}{2})+O(1)=O(logn)$ כאשר n הוא מס' האיברים המשותף במערך המאוחד הכולל. נראה כי נרצה להפטר מחצי מהאיברים בכל שלב. נסתכל על איברי החציון של A וB (ממויינים ולכן אנחנו יודעים מיהם) כעת נסמנם בהתאמה $A[\frac{n}{4}]$ ו$B[\frac{n}{4}]$.

אם $A[\frac{n}{4}] > B[\frac{n}{4}]$ זה אומר שהחציון של A גדול מהחציון של B, ולכן האיברים בהם נלך לחפש כאן יהיו החצי האחרונים של B עם החצי הראשונים של A שהרי כעת שם יש $\frac{n}{2}$ איברים ובהם פוטנציאל להיות החציון
אם $A[\frac{n}{4}] < B[\frac{n}{4}]$ כעת החציון של B גדול מהחציון של A ולכן כעת נלך לחפש בחצי הראשונים של B עם חצי האחרונים של A.

64. מציאת האיבר הk בגודלו במערכים ממויינים בגודל זהה (הכללה של 63): האיבר הk בגודלו חייב להמצא בk האיברים הראשונים של A או B. לכן נסתכל עליהם בלבד. גודל הקלט ההתחלתי שלנו יהיה $2k$ ונרצה לעבוד בדומה לסעיף א' לנסות לצמצם את גודל הקלט. אם $k = \frac{n}{2}$ זה בדיוק הסעיף הראשון. כעת נסתכל בכל שלב על החציונים בk האיברים הראשונים בכל מערך. לפני כן נחלק למקרים.
נסתכל על האיבר המקסימלי במערך A והמינימלי במערך B. אם המקסימלי קטן מהמינימלי אזי שכל איברי מערך אחד קטנים מהשני. כעת, החיפוש יתבצע במערך הראשון בלבד. באופן סימטרי על המערך השני.
אחרת אנחנו לא במקרה קצה שכזה, לכן נסתכל על החציונים. אם סכום האיברים עם החציונים משמאל בשני המערכים קטן שווה לk נלך רקורסיבית שמאלה. אחרת, הוא גדול מk לכן נרצה ללכת ימינה. סה"כ נקבל

$$T(2k) = T(k) + O(1) = O(logk)$$

65. איחוד שני $BTREE$ באותו הגובה, כאשר כל הערכים בעץ אחד קטנים מבעץ השני: ניצור צומת חדשה עם מפתח מפריד שיקיים $min(T_2) > k > max(T_1)$. ניצור צומת פנימי חדש R ונשים בו מפתח k. כעת בן שמאלי שלו יהיה T_1 וימני T_2, אחרי זה נוציא את k ב$O(logn)$ וזה גם סיבוכיות הפתרון.
איחוד שניים כלליים: פעמיים $inorder$, למיין ב$O(n)$ מחדש ולבנות רקורסיבית מהמערך הממויין. העץ יבנה מלמטה למעלה.
גובה שונה אך כל הערכים בעץ אחד קטנים מבעץ השני: $h_1 < h_2$. המפתחות בT_1 קטנים יותר. הסיבוכיות תהיה ב$O(m * (h_2 - h_1))$ כאשר m דרגת העץ. נרד בעץ הגבוה ונלך שמאלה בכל פעם עד שנגיע לצומת בגובה h_1, זה יעלה $h_2 - h_1$ שלבים. כעת מכניסים את ערך k (בוחרים כמו שחברנו במקרה מלמעלה) ואת T_1 בתור בן שלו. אם יש מקום הכנסת הילד והמפתח תעלה $O(m)$, אחרת נעשה $split$ וזה יעלה עד $h_2 - h_1$ כגובה העץ. סה"כ נגיע לסיבוכיות הנדרשת. אם גדלי העצים $n < m$ ואנחנו בעץ $2-3$ למשל נקבל סיבוכיות של $O(3(logm-logn)) = O(3log(\frac{m}{n})) = O(logm)$.

66. עצ'י פיבונאצי: ישנם טענות שיכולות לעזור מאוד בזמן אמת אם זוכרים כי הוכחותן קלות מאוד. למשל:
מס' הצמתים בעץ פיבונאצ'י מסדר n יהיה $F_{n+2} - 1$
מס' העלים בעץ פיבונאצי מסדר n יהיה F_{n+1}
לכל $n \geq 2$ גובה עץ פיבונאצ'י מסדר n הינו $n - 1$
גובה עץ פיבונאצ'י עם n קודקודים הוא $O(logn)$
(F אלו מס' פיבונאצי)

67. בהינתן שני מערכים ממויינים בגודל n מצא בהינתן איבר $A[i]$ את מיקומו היחסי בהינתן $2n$ איברים.
פתרון: עבור האיבר שקיבלנו נדע מה האינדקס שלו, ואם לא אז נחפש בינארית במערך הזה. במערך השני נחפש בינארית עד שנמצא איבר שגדול ממנו. מיקום זה יתן לנו כמה קטנים ממנו

במערך השני +אנחנו יודעים כמה קטנים ממנו במערך הנוכחי = מצאנו מיקום יחסי שלו במערך כולל. סה"כ $O(logn)$

68. סלקט:
תאר אלגוריתם בסיבוכיות לינארית שבודק אם יש איבר שמופיע יותר מ$\frac{n}{3}$ פעמים - סלקט על $\frac{n}{3}, \frac{2n}{3}$ (פוטנציאלים) . דגש על המילה יותר!
תאר שמופיע לפחות $\frac{n}{3}$- כעת הפוטנציאלים הם $\frac{n}{4}, \frac{2n}{4}, \frac{3n}{4}$

69. עצי B: בביצוע פעולת הכנסה לעץ עם מקדם m של n איברים יתבצעו $\frac{n}{m}$ פיצולים. אם נקח עץ $2-3$ למשל לשיעורין מס' הפיצולים עבור פעולת הכנסה יהיה $\frac{\frac{n}{3}}{n} = \frac{1}{3} = O(1)$, לכן לשיעורין הכנסה לעץ B תהיה ב$O(1)$.

70. $3SUM$ משודרג: בהינתן מערך ומס' k מצא את שלושת המספרים שסכומם יהיה הכי קרוב לk.
נמיין את המערך, יעלה לי $O(nlogn)$. כעת, לכל i במערך נסתכל על האיבר $A[i]$ ונשווה אותו לשני מצביעים. בתחילה הם יהיו על $A[i+1], A[n-1]$., אם סכום שקיבלנו יהיה גדול מטרגט, אנחנו נזיז את המצביע הימני שמאלה, אם קטן מטרגט נזיז את המצביע השמאלי ימינה. בכל שלב נחשב את המרחק של האיברים מ$target$ בתוך משתנה שיזכור גם מי היו האיברים שהביאנו עד לכאן, ואם קיבלנו מרחק קטן יותר הוא יהיה החדש. סה"כ זה מעבר לינארי n איברים, כפול n פעמים = $O(n^2)$.

שאלה מעולה שחשבתי שכדאי להכניס לכאן:

עבור מערך $A[1,...n]$ המכיל מס' ממשיים נסמן $m_A = min_{1\le i\le n}A[i], M_A = max_{1\le i\le n}A[i]$.
א. הוכיחו כי קיימים $1 \le i \ne j \le n$ כך ש$0 \le A[i] - A[j] \le \frac{M_A - m_A}{n-1}$
רמז: הוכיחו כי במערך B אשר מתקבל ע"י מיון מערך A קיים $1 \le i \le n-1$ עבורו $0 \le B[i+1] - B[i] \le \frac{M_A - m_A}{n-1}$

פתרון:
נעזר ברמז ונמיין את מערך A. בהכרח נקבל $B[n] = M_A$ וכן $B[1] = m_A$. כלומר, צ"ל כי קיימים $0 \le B[i+1] - B[i] \le \frac{B[n]-B[1]}{n-1}$
נשים לב שאם הוכחנו את הטענה כמובן שהוכחנו אותה עבור A כי הם אכן קיימים בA.
נב"ש שלא קיים אינדקס כזה i. כלומר, לכל i במערך B מתקיים $B[i+1] - B[i] < 0$ ואז נקבל $B[i] > B[i+1]$ בסתירה.
או ש$B[i+1] - B[i] > \frac{B[n]-B[1]}{n-1}$. בפרט זה יתקיים לכל i ולכן

$$\sum_{i=1}^{n-1} B[i+1] - B[i] > (n-1)\frac{B[n]-B[1]}{n-1} = B[n] - B[1]$$

נראה כי הסכום המדובר משמאל הינו סכום טלסקופי.

$$B[2] - B[1] + B[3] - B[2] + ... + B[n] - B[n-1] = B[n] - B[1]$$

כלומר קיבלנו סה"כ

$$B[n]-B[1]>B[n]-B[1]$$

והרי שזו סתירה כיוון שזה ממש שווה.

ב. כתוב אלגוריתם שמקבל כקלט מערך $A[1,...n]$ ומוצא זוג אינדקסים $1\leq i\neq j\leq n$ שמקיימים $0\leq A[i]-A[j]\leq\frac{M_A-m_A}{n-1}$, זמן ריצת האלגוריתם $O(n)$

פתרון:

נסמן את הערך $t=\frac{M_A-m_A}{n-1}$, שיהיה יותר קל לעבוד, כעת צריך למצוא $0\leq A[i]-A[j]\leq t$ בזמן $O(n)$ והרי שקיימים כאלו מסעיף א'. את הערך t נוכל לחשב בקלות ע"י הפעלת סלקט (או פשוט סריקה של מקסימום ומינימום במערך, יעלה $O(n)$)

נשתמש במושג שנקרא בקט - "תא " אליו נשייך איברים. אנחנו יודעים את טווח המספרים של כל אחד מהאיברים $m_A\leq A[i]\leq M_A$. נפרוס את הקטע ל$n-1$ קטעים, כל אחד שווה ברוחבו, למשל אם יש לנו טווח מספרים $[1,99]$ ויש 4 מספרים, נחלק את המקטע ל3 בקטים, $[1,33],[34,66],[67,99]$ שווים באורכם. כעת לכל איבר נשלח אותו לבקט מסויים לפי הנוסחה $\lfloor\frac{x-m_A}{t}\rfloor$ כאשר t הוא הערך בו דנו קודם וm_A הוא המינימום של המערך. כך למשל האיבר 45 בתוך המערך $[1,45,58,99]$ בו $t=\frac{99-1}{4-1}=32.66$ יכנס לבקט $1=\lfloor\frac{45-1}{32.66}\rfloor=\lfloor\frac{x-m_A}{t}\rfloor$. כעת לאחר שביצענו פעולות אריתמטיות שעלו לכל היותר $O(n)$, יש לנו $n-1$ מקטעים, לפי סעיף א' ושובך היונים קיים איבר שנמצא בשניהם, ולכן נסרוק את הבקטים פעם אחרונה למצוא את הבקט עם שני האיברים (יתכנו יותר, נעצור כאשר נמצא את האחד הראשון).