27.6.2010

מבחן מועד א' בקורס מבני נתונים, שנת הלימודים תש"ע

מרצה: פרופ' דורית אהרונוב
מספר קורס: 67109
משך הבחינה: שעתיים וחצי

הורואת כלליות:

1. למבחן שני חלקים. עליכם לפתור 4 שאלות מתוך 5 שאלות בחלק א' (כל שאלה שווה 10 נקודות), ו2 שאלות מתוך 3 בחלק ב' (כל שאלה שווה 30 נקודות).
2. אנא סמנו בבירור אילו שאלות ברצונכם להגיש. במידה ועניתם על יותר ממספר השאלות הדרוש, השאלות הראשונות יבדקו.
3. אנא נמקו כל צעד בהוכחות ובניתוחי הסיבוכיות, בבהירות רבה ככל האפשר, בכתב ברור ככל האפשר ובלי להאריך יתר על המידה! ינתן ניקוד שלילי על הסברים חסרים, אך גם על הערות לא רלוונטיות או שמראות על אי הבנה של השאלה.
4. כאשר אתם מתבקשים לכתוב פסיאודו קוד, כתבו בצורה שניתנת לקריאה שוטפת ו**לא** ב־$Java$!
5. כאשר אתם מתבקשים לנתח "סיבוכיות של אלגוריתם", אלא אם כן נכתב אחרת, הכוונה לחסם עליון אסימפטוטי ($Big-O$) על הסיבוכיות של המקרה הגרוע ביותר.
6. אין להשתמש בחומר נוסף מלבד כלי כתיבה.
7. ההוראות במבחן ניתנות בלשון זכר – אך מכוונות לנשים ולגברים כאחת!

בהצלחה!!!

דף נוסחאות:

1. משפט האב (Master Theorem):
   יהיו $a \geq 1,\ b > 1$ ותהי $T(n)$ פונקציה המוגדרת על הטבעיים ומקיימת

   $$T(n) = aT(\frac{n}{b}) + \theta(n^k)$$

   כאשר $\frac{n}{b}$ הוא $\lceil \frac{n}{b} \rceil$ או $\lfloor \frac{n}{b} \rfloor$:

   (א) אם $k < \log_b a$ אזי $T(n) = \theta(n^{\log_b a})$
   (ב) אם $k = \log_b a$ אזי $T(n) = \theta(n^k \log n)$
   (ג) אם $k > \log_b a$ אזי $T(n) = \theta(n^k)$

2. חוקי לוגריתמים:

   $$\log_b a = \frac{\log_2 a}{\log_2 b}$$

3. טורים גיאומטריים:

   $$\forall x \neq 1 : \quad \sum_{i=0}^{n} x^i = \frac{1 - x^{n+1}}{1 - x}$$

   $$\forall x < 1 : \quad \sum_{i=0}^{\infty} x^i = \frac{1}{1 - x} \quad \text{and} \quad \sum_{d=0}^{\infty} dx^d = \frac{x}{(1 - x)^2}$$

חלק א': ענה על 4 מתוך 5 השאלות (כל שאלה- 10 נק', סה"כ 40 נק')

1. נניח שמובטח שהשיגרה Partition מחלקת את הקלט לשני חלקים כך שהחלק הקטן יותר מהווה לפחות $3/10$ מהקלט. הוכח שתחת הנחה זו זמן הריצה של האלגוריתם Select הוא $O(n)$.

1. **פתרון:** כפי שנלמד בתרגולים האלגוריתם Select מוצא את האיבר ה- i הכי גדול במערך בגודל n. האלגוריתם עובד באופן הבא: מפעיל את Partition לחלוקת המערך סביב פיבוט כלשהו (איברים קטנים מהפיבוט משמאלו וגדולים מימינו). אם הפיבוט איננו האיבר שחיפשנו, אז האלגוריתם ממשיך באופן רקורסיבי על **אחד** מתתי המערכים (משמאל או מימין לפיבוט) עד אשר האיבר נמצא, או עד אשר מגיעים למערך בגודל 0.
במקרה הגרוע ביותר הקריאה הרקורסיבית תעשה תמיד על תת המערך הארוך יותר. מהנתון בשאלה מובטח שבכל פעם ש Partition נקרא הוא מחלק את המערך כך שתת המערך הקצר יותר מהווה לפחות $3/10$ מתת המערך עליו הוא נקרא, מכאן הצד הארוך יותר יהווה לכל היותר $7/10$ מתת המערך. כפי שראינו בכיתה ובתרגולים סיבוכיות המקרה הגרוע ביותר של Partition היא $\Theta(n)$ ומכאן הסיבוכיות מקרה הגרוע ביותר של Select תחת ההנחות שלנו תקיים את נוסחת הנסיגה הבאה:

$$T(n) \leq T\left(\frac{7 \cdot n}{10}\right) + \Theta(n)$$

מכיוון שאנו מחפשים חסם עליון על הסיבוכיות מקרה גרוע ביותר ניתן לחסום את $T(n)$ על ידי $T'(n)$ המקיימת את נוסחת הנסיגה:

$$T'(n) = T'\left(\frac{7 \cdot n}{10}\right) + \Theta(n)$$

מכיוון שנוסחת הנסיגה היא מהצורה:

$$T(n) = a \cdot T\left(\frac{n}{b}\right) + \Theta(n^k)$$

עבור $a = 1, b = \frac{10}{7}, k = 1$ ניתן להפעיל את משפט האב. $log_b a = log_{\frac{10}{7}} 1 = 0 < 1 = k$ ולכן אנחנו במקרה השלישי. כלומר סיבוכיות המקרה הגרוע ביותר של $T'(n)$ היא $\Theta(n)$ ומכאן סיבוכיות המקרה הגרוע ביותר של $T(n)$ היא $O(n)$.
הערה: מכיוון שסיבוכיות המקרה הגרוע שביותר של $Partition$ היא $\Theta(n)$ נובע כי סיבוכיות המקרה הגרוע של $T(n)$היא $\Theta(n)$ ולא רק $O(n)$ אך לא נידרשנו להראות זאת בשאלה.

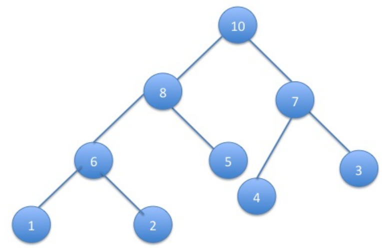
טעויות נפוצות: הטעות הנפוצה ביותר הייתה אי הבנה שהאלגוריתם פועל על צד אחד של המערך בלבד. לדוגמא רבים הגיעו לנוסחת נסיגה בסגנון:

$$T(n) = T\left(\frac{7 \cdot n}{10}\right) + T\left(\frac{3 \cdot n}{10}\right) + \Theta(n)$$

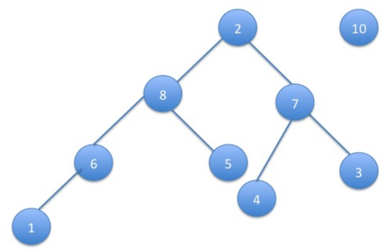
רובם המכריע של הסטודנטים שהגיעו לנוסחא זאת או דומה לה פתרו אותה בצורה לא נכונה והגיעו לתוצאה של $O(n)$ ולא לתוצאה של $\Theta(nlog(n))$ שהיא פתרונה הנכון של הנוסחא.

2. הרץ את extract-max על הערימה $[10, 8, 7, 6, 5, 4, 3, 1, 2]$. תאר בעזרת ציורים את מצב הערימה בשלבי הביניים של האלגוריתם.

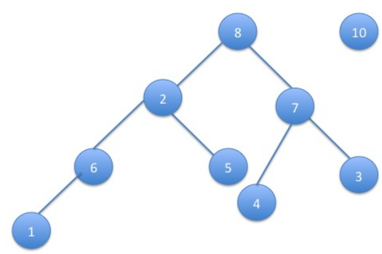
2. **פתרון:** לפני הוצאת המקסימום הערימה נראית כך:



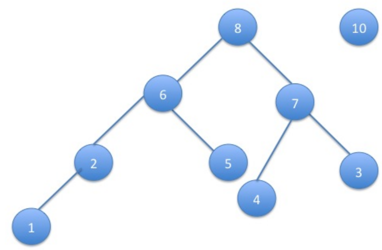
אחרי הוצאת השורש (10) שמים את האיבר האחרון בערימה (2) בתור השורש ואז הערימה נראית כך:



קוראים לפונקציה $max - heapify$ על השורש ובשלב הראשון הערימה נראית כך:



ולבסוף היא נראית כך:



טעויות נפוצות:

(א) אחרי הוצאת המקסימום לשים בתור השורש את האיבר הכי קטן (1) במקום האיבר האחרון (2).

(ב) בניית עץ חיפוש בינארי במקום ערימה.

(ג) לא ידעו לצייר נכון את הערימה שכתובה במערך.

(ד) בפונקציה $max - heapify$ להוריד את 2 כך שיהיה בסוף הבן של 1.

3. הגדר במדוייק $f(n) = o(g(n))$ ו־ $f(n) = \Omega(g(n))$ (אומגה גדול ו o־קטן). הוכח או הפרך: אם $f(n) = \theta(g(n))$ אזי $f(n) = (1 + o(1)) \cdot g(n)$.

3. **פתרון:** $\lim_{n\to\infty} \frac{f(n)}{g(n)} = 0 \iff f(n) = o(g(n))$ או באופן שקול, אם לכל $\epsilon > 0$ קיים $N \in \mathbb{N}$ כך שלכל $n > N$ יתקיים $\frac{f(n)}{g(n)} < \epsilon$.
$f(n) = \Omega(g(n))$ אם קיימים קבועים $c > 0$ ממשי חיובי ו־N טבעי, כך שלכל $n > N$ יתקיים $f(n) > c \cdot g(n)$.
הטענה שאם $f(n) = \theta(g(n))$ אז $f(n) = (1 + o(1)) \cdot g(n)$ שגויה. נראה דוגמה נגדית: הפונקציות

$$f(n) = 3n,\ \ g(n) = 2n$$

מקיימות את התנאי $f(n) = \theta(g(n))$ אולם

$$f(n) = (1 + \frac{1}{2}) \cdot g(n)$$

והפונקציה הקבועה $\frac{1}{2}$ איננה $o(1)$ (משום שהיא לא שואפת ל־0 כש־n שואף ל־∞.
טעויות נפוצות:

(א) השמטת האיפיון $c \in \mathbb{R}$, $c > 0$, או הכימות על n_0.

(ב) שגיאה מהותית בהגדרה, למשל בסדר הכמתים.

(ג) הפרכה לא מספיק מפורטת, למשל לא להסביר למה פונקציה קבועה איננה $o(1)$ או לא לתת דוגמה נגדית.

(ד) פורמליות חלקית, למשל לטעון ש־$2 > 1 + o(1)$ בלי לציין שזה נכון רק החל מ־n מסויים.

(ה) הפרכת הטענה עבור $h \in o(1)$ באמצעות דוגמה פרטית של h כזו, במקום הפרכת הטענה לכל h כזו.

4. תן דוגמה לגרף שהוא DAG, בעזרת ציור יחד עם ייצוג רשימת שכנויות, כך שהרצת DFS עליו (לפי הסדר ברשימת השכנויות), וסידור הקדקודים על פי סדר עולה של pre **אינו** נותן מיון טופולוגי. מהו המיון הטופולוגי שמתקבל משימוש באותה הרצה של DFS וסידור הקדקודים ע"פ סדר יורד של $post$? כתוב את הפרמטרים pre,post ליד כל קדקוד בגרף שציירת.

4. **פתרון:** עלינו למצוא גרף מכוון חסר מעגלים וייצוג רשימת שכנויות עליו כך שהרצת DFS עליו לפי סדר הקודקודים ברשימת השכנויות יתן תוצאות כדלהלן: סידור הקודקודים לפי סדר עולה של pre ייתן מיון טופולוגי, בעוד שסידור הקודקודים לפי $post$ יורד כן ייתן. הגרף הפשוט ביותר שניתן לתת הוא גרף בן שני קודקודים $x \longrightarrow y$, כאשר ייצוג רשימת השכנויות הוא

y	$NULL$
x	y

. לפי ייצוג זה, הרצת DFS תתחיל ב y ואז תעבור ל x, ולכן

$$pre(y) = 1, post(y) = 2, pre(x) = 3, post(x) = 4$$

הסידור לפי pre הוא קודם y ואז x, וזה לא מיון טופולוגי כי יש צלע מ x ל y ולכן y חייב להופיע אחרי x. אם מסדרים לפי $post$ יורד מקבלים קודם x ואז y וזה אכן מיון טופולוגי. פתרונות אחרים שבהם היו גרפים גדולים הרבה יותר כמובן גם אפשריים. כמעט כל גרף שמנסים עובד כדוגמא נגדית אם בוחרים אם סדר הקודקודים בייצוג רשימת השכנויות כמו שצריך.

טעויות נפוצות:

(א) דוגמא שהיא גרף לא מכוון - בגרף זה לא מוגדר כלל מיון טופולוגי, כי אין סדר.

(ב) הרבה תלמידים לא סימנו חצים על הצלעות למרות שמייצוג רשימת השכנויות היה ברור שהם התכוונו לגרף מכוון. על כך לא הורדו נקודות אבל שימו לב לכך בקורסים הבאים...

(ג) תלמידים רבים נתנו כדוגמא גרף עם מעגלים - זה הרי מראש לא הגיוני. הראינו בכיתה שלגרף כזה לא יתכן שיש מיון טופולוגי, וכמו כן נתבקשתם במפורש לתת דוגמא של גרף חסר מעגלים.

(ד) מספר תלמידים נתנו גרף עם מעגלים, ואז דיברו על המיון הטופולוגי של גרף רכיבי הקשירות החזקים שלו, על פי pre או $post$ של אחד הקודקודים בכל רכיב (מבלי להגדיר במדויק איך בוחרים את הקודקוד שלפיו קובעים..) זה לא מה שנשאלתם - בגרף חסר מעגלים כל רכיב קשירות חזק בהכרח מורכב מקודקוד בודד.

(ה) תלמידים רבים לא הבינו את ההגדרה של מיון טופולוגי, ונמקו בצורה מוטעית מדוע הסידור שקיבלו אינו מיון טופולוגי: הם חשבו למשל שמיון טופולוגי אמור בצורה מסוימת לתת את הדרך הקצרה ביותר בין קודקודים, או שבמיון טופולוגי, אם קודקוד אחד מופיע אחרי אחר, אז חייבת להיות צלע מהקודקוד הראשון לשני. לא נדרשתם לנמק מדוע המיון הטופולוגי נכשל בגרף, אבל מי שנימק בצורה שהראתה חוסר הבנה של המושג מיון טופולוגי, הורדו לו נקודות.

(ו) טעות קטנה אך מוזרה היתה שתלמידים רבים חשבו משום מה שהשעון מתעדכן בנפרד ל pre ול $post$.

(ז) תלמידים רבים לא ציינו את ייצוג רשימת השכנויות או השתמשו בייצוג מטריצת שכנויות, למרות שהתבקשתם במפורש להשתמש ברשימת שכנויות.

5. עבור מערך עם n מספרים שלמים בין 1 ל- k, כאשר $k = n^3$, באיזה מהמיונים עדיף להשתמש- Counting-Sort או Randomized Quick-Sort, ומדוע?

5. **פתרון:** ראיתם בקורס שזמן הריצה של $C.S$ הוא $\theta(n+k)$. במקרה זה, מכיוון ש $k = n^3$ ו $n = o(n^3)$, נקבל שזמן הריצה של $C.S$ הוא $\theta(n+n^3) = \theta(n^3)$. כמו כן, ראיתם שתוחלת זמן הריצה של $R.Q.S$ היא $\theta(n \log n)$, וכן שבמקרה הגרוע (אם במקרה כל פעם איבר הציר יוצא בקצה המערך) אזי זמן הריצה הוא $O(n^2)$, כך שבכל מקרה האלגוריתם רץ ב $O(n^2)$. מכיוון ש $n^2 = o(n^3)$, אזי עדיף להשתמש ב $R.Q.S$.

טעויות נפוצות:

(א) המון תלמידים כתבו ש $C.S$ רץ ב $O(n)$, ואח"כ ציינו שזה רק כאשר k קבוע או קטן מ n (למעשה זה נכון לכל $k = O(n)$). לא הורדו נקודות על כך, כי זו לא ממש טעות, אבל בכל זאת עדיף לכתוב ש $C.S$ רץ ב $O(n+k)$.

(ב) נימוקים למה $R.Q.S$ עדיף משיקולי זיכרון ($C.S$ מגדיר מערך עזר וכו'). מכיוון שהבעיה ה"אמיתית" כאן היא זמן הריצה, נימוקים כאלו הם לא תשובה מלאה. אפשר היה להוסיף אותם לנימוק על זמן הריצה, אך הם לא מספיקים.

(ג) התייחסות לכך שמערך העזר של $C.S$ יהיה ריק ברובו- זה לא מראה כלום, בכל מקרה נרוץ על כל המערך.

(ד) כתיבת זמן הריצה של $R.Q.S$ כ $O(k \log k)$.

(ה) נפנופי ידיים מוגזמים- חוסר התייחסות לזמני הריצה ("$C.S$ יהיה הרבה יותר איטי" וכו'), תיאור מילולי של האלגוריתמים במקום כתיבת זמן הריצה, ועוד כהנה וכהנה נימוקים לא משכנעים.

(ו) תשובה לא נכונה- $C.S$ יהיה יעיל יותר.

חלק ב': ענה על 2 מתוך 3 השאלות. (כל שאלה־ 30 נק', סה"כ 60 נק')

1. גיבוב:

(א) הגדר מהי משפחה אוניברסלית של פונקציות גיבוב, תן דוגמא, ללא הוכחה, למשפחה כזו (כולל התנאים על הפרמטרים), וכמו כן הגדר את פקטור העומס α בטבלת גיבוב.

(ב) הוכח כי זמן החיפוש של איבר Y שנמצא בטבלת גיבוב כאשר משתמשים בגיבוב אוניברסלי הוא $O(1+\alpha)$.

(ג) נתון מערך עם n איברים <u>שלמים חיוביים,</u> וכן שלם חיובי z. עליך להחליט אם קיימים שני איברים x, y במערך (יתכן ש $x=y$), כך ש $x+y=z$. (לדוגמה, במערך הבא $[12,14,23,24,2,19,27,41]$ עם $z=29$ התשובה היא כן, שכן ניתן לבחור $x=27, y=2$).
תאר אלגוריתם מבוסס גיבוב אוניברסלי, שמכריע בתוחלת זמן $O(n)$ אם אכן קיימים זוג איברים כאלו במערך נתון. הסבר מדוע הסיבוכיות היא אכן כנדרש.

1. **פתרון:**

(א) משפחה אוניברסלית של פונקציות גיבוב היא קבוצה של פונקציות

$$\mathcal{H}=\{h:U\rightarrow\{1,...,m\}\}$$

כאשר U הוא עולם המפתחות ו m הוא גודל הטבלה, כך שלכל זוג מפתחות $x,y\in U$ כך ש $x\neq y$ מתקיים ש

$$Pr_{h\in_R\mathcal{H}}(h(x)=h(y))\leq\frac{1}{m}$$

כלומר אם מגרילים h בהתפלגות אחידה, הסיכוי של x ו y להתמפות לאותו תא קטן יחסית. דוגמה למשפחה כזו:יהי p ראשוני גדול ($p>>|U|$), עבור a,b שלמים ומפתח k שלם, נגדיר $h_{a,b}(k)=((ak+b)mod\ p)mod\ m$.
משפחה אוניברסלית של פונקציות גיבוב היא

$$\mathcal{H}=\{h_{a,b}:a\in\{1,...,p-1\},b\in\{0,...,p-1\}\}$$

פקטור העומס בטבלת גיבוב הוא $\alpha=\frac{|N|}{m}$ כאשר $|N|$ הוא מספר המפתחות אותם אנחנו מגבבים בפועל.
טעויות נפוצות:

i. הטעות הכי "חמורה" היא לא להגדיר נכון על מה ההסתברות במשפחה אוניברסלית. שימו לב־ אם אתם מתחילים משפט ב "לכל $h\in H$ ולכל $x\neq y\in U$", אזי קיבעתם גם את הפונקציה וגם את המפתחות, ואין שום משמעות להסתברות: או ש $h(x)=h(y)$ או שלא. כמו כן, ההסתברות היא לא על המפתחות השונים, בהנתן פונקציה קבועה, אלא על הפונקציות, בהנתן מפתחות קבועים. וודאו שאתם מבינים את הנקודה הזו.

ii. המשפחה $h(k)=kmod\ p$ מורכבת מפונקציה אחת בלבד, ובוודאי שאיננה אוניברסלית.

iii. טעויות בהגדרת פקטור העומס־ $\alpha=\frac{|U|}{m}$.

iv. נפנופי ידיים בהגדרת פקטור העומס־ "מספר המפתחות בכל תא" (זו פשוט טעות), "מספר המפתחות המקסימלי בתא מסויים" (שוב, טעות).

(ב) נתון ש Y נמצא בטבלה, לכן כדי למצוא אותו נצטרך להפעיל את פונקצית הגיבוב, ובמקרה הגרוע לרוץ על כל הרשימה של האיברים שגובבו לאותו תא של Y. לכן, תוחלת זמן הריצה של החיפוש היא תוחלת אורך אותה רשימה. נחשב את אותה תוחלת. נגדיר את המשתנה המקרי X שמחשב את מספר האיברים שגובבו לאותו תא כמו Y. נגדיר את המשתנים המקריים הבאים: לכל $x,y\in U$

$$C_{x,y}=\begin{cases}1 & h(x)=h(y)\\0 & h(x)\neq h(y)\end{cases}$$

אזי אורך הרשימה שאליה Y גובב הוא בדיוק מספר האיברים x <u>מתוך הקבוצה שאותה גיבבנו בפועל,</u> שעבורם $C_{xY}=1$. נסמן את קבוצת המספרים שגיבבנו ב N, אזי

$$\mathrm{X}=\sum_{x\in N}C_{xY}$$

ולכן אנו מעוניינים בתוחלת הסכום הזה. מלינאריות התוחלת נקבל ש

$$E[X] = \mathbb{E}[\sum_{x\in N} C_{xY}] = \sum_{x\in N} \mathbb{E}[C_{xY}] = \sum_{x\in N} 1\cdot Pr(C_{xY}=1) =$$

$$= Pr(C_{YY}=1) + \sum_{x\in N\setminus\{Y\}} Pr(C_{xY}=1) \leq 1 + \sum_{x\in N\setminus\{Y\}} \frac{1}{m} = 1 + \frac{N-1}{m} < 1+\alpha$$

ולכן תוחלת אורך הרשימה היא $O(1+\alpha)$, כפי שרצינו.

טעויות נפוצות:

i. נפנופי ידיים־ בעיה חמורה בסעיף הזה, כל מני הסברים מילוליים למה הטענה נכונה. שימו לב־ יש הרבה טענות שהן במבט ראשון מאוד ברורות באופן אינטואיטיבי, חלקן באמת לא קשה להוכיח, אבל זה מה שאתם צריכים לדעת לעשות. הסיבה לכך היא שטענות אחרות הן ברורות אינטואיטיבית, אבל קשות מאוד להוכחה, ואת חלקן אפילו לא יודעים להוכיח עד היום (ע"ע $P\neq NP$).

ii. תלמידים רבים זכרו את ההוכחה בקווים כלליים, וזכרו לאן צריך להגיע בסוף, אבל לא את האמצע. יצאו כל מני חישובים תמוהים בסכימת המשתנים. הרבה יותר טוב להתקע באמצע חישוב ולהבין שיש טעות מאשר להכריח את החישוב לצאת כמו שאתם רוצים.

(ג) הרעיון של האלגוריתם: נגבב את המערך לטבלת גיבוב בגודל n באמצעות גיבוב אוניברסלי. כך נקבל $\alpha=1$, ולכן תוחלת זמן החיפוש של איבר היא $O(1+1)=O(1)$. כעת, עלינו לבדוק האם יש זוג איברים x,y שמקיימים $z=x+y$. די לעבור על כל האיברים במערך ועבור כל איבר x לבדוק אם יש איבר y כך ש $z=x+y$. איבר y שכזה מקיים $y=z-x$. לכן, מספיק לעבור על כל איברי המערך ולכל איבר x לבדוק בטבלת הגיבוב האם $z-x$ נמצא.

מדוע זמן הריצה הוא $O(n)$? בניית הטבלה לוקחת $O(n)$, שכן בחרנו לבנות טבלה בגודל $O(n)$ ולגבב n איברים. בזמן החיפוש אנחנו רצים על n איברים לכל היותר. מכיוון שחיפוש אחד עולה בתוחלת $O(1)$, נקבל בסופו של דבר תוחלת זמן ריצה של $O(n)$, כנדרש.

טעויות נפוצות:

i. הבעיה הכי נפוצה־ התעלמות מפקטור העומס (ומגיבוב אוניברסלי בכלל). שימו לב, חשוב שהטבלה תהיה מספיק גדולה כדי שפקטור העומס יהיה קבוע. אם הטבלה בגודל 10, למשל, אז פקטור העומס יהיה גדול ($O(n)$). מצד שני, אם הטבלה תהיה גדולה מדי, אז בנייתה תיקח יותר מדי זמן. לכן צריך בדיוק טבלה בגודל $O(n)$, כדי ששני התנאים יסופקו.

ii. עוד טעות נפוצה הייתה לא לכתוב שנגבב את האיברים לטבלת גיבוב ע"י פונקציית גיבוב אוניברסלי ואז להניח שהם כבר מגובבים.

iii. להשתמש בפונקציות שאינן פונקציות גיבוב אוניברסליות לדוגמא: $h(k)=kmod\ m$ $h(k)=z-k$.

iv. פתרונות מבוססי z־ בניית טבלה בגודל $O(z)$ היא לא טובה! אין שום סיבה ש z יהיה קשור בגודלו ל n, ובוודאי אין סיבה ליחס לינארי ביניהם. כל פתרון שכלל טבלאות בגודל z פספס את הנקודה.

v. אלגוריתמים משונים כיד הדמיון הטובה עליכם. ביקשנו אלגוריתם מבוסס גיבוב אוניברסלי, זה רמז די גדול. בכל זאת ראינו פתרונות מבוססי מיון (מהו החסם התחתון על מיון? איך זה מסתדר עם דרישות זמן הריצה?), פתרונות מבוססי מיזוג, ולעתים את הפתרון הנאיבי של לעבור על כל הזוגות ($O(n^2)$).

2. נאמר שעץ חיפוש בינארי הוא עץ $2AVL$ אם לכל קדקד ההפרש בין הגבהים של צאצאיו הוא לכל היותר 2. נסמן ב־n_k את מספר הקדקדים המינימאלי בעץ $2AVL$ בגובה k.

(א) הוכח ש־n_k מונוטוני עולה ב־ k. נמק במפורט כל טענה!

(ב) כתוב נוסחת רקורסיה ל־n_k כולל תנאי שפה מלאים. נמק במפורט מדוע הנוסחה וכל תנאי השפה נכונים!

(ג) הסק מהסעיפים הקודמים ש־$n_k \geqslant 2n_{k-3}$, והוכח חסם עליון $O(log(n))$ על גובה עץ $2AVL$ (לאו דוקא מינימלי) בעל n קדקודים.

2. **פתרון:**

(א) צ"ל ש $n_k < n_{k+1}$. **הוכחה 1:** נבחר עץ מינימאלי כלשהו בגובה$k+1$. העץ הנ"ל מורכב מתת עץ שמאלי, תת עץ ימני ושורש. כיוון שדרישת ה־ $AVL-2$ מוגדרת על כל הקודקודים בעץ, כל תת עץ יהיה $AVL-2$ גם כן, שהרי כל הקודקודים בו מקיימים את הדרישה. בכדי שהגובה של העץ יהיה $k+1$ אחד מתתי העצים חייב להיות בגובה k (אחרת לא יהיה מסלול באורך $k+1$ מהשורש לעלה מכיוון שאין מסלול באורך k מאף אחד מבניו של השורש לעלה).

טענה: תת העץ הנ"ל חייב להיות מינימאלי ביחס לגובהו, כלומר מספר הקודקודים בו חייב להיות n_k. הוכחה: נניח בשלילה שלא כלומר שיש בתת העץ יותר מ n_k קודקודים. מכאן, ניתן להחליפו בתת עץ אחר שהוא $AVL-2$ בגובה k ובעל מספר קודקודים קטן יותר. ההחלפה לא תפגע בתכונת ה־ $AVL-2$ של אף אחד מהקודקודים: השורש, תת העץ הנותר ותת העץ החדש, מכיוון שהגובה של בניהם לא השתנה. כלומר העץ שיתקבל הוא עץ $AVL-2$ חוקי בעל מספר קודקודים קטן יותר מהעץ המקורי בסתירה למינימאליות.

מכאן $n_{k+1} \geq n_k + 1 > n_k$ מכיוון שבעץ המקורי (בגובה $k+1$) יש לפחות מספר הקודקודים בתת העץ בגובה k ועוד השורש.

הוכחה 2: נניח שיש בידינו עץ $AVL-2$ עם מספר קודקודים מינימלי, בגובה $k+1$. נוריד ממנו את כל העלים בגובה $k+1$. <u>טענה:</u> קיבלנו עץ $AVL-2$ בגובה k. <u>הוכחה:</u> נניח שיש קודקוד מסוים בעץ המקורי, שלו שני תתי עצים, בגבהים h_1, h_2. אם באף אחד מתתי העצים אין עלה ברמה ה $k+1$ הגבהים לא השתנו אחרי הורדת רמה זו. אם בשני תתי העצים יש קודקוד כזה שני הגבהים ירדו באחד בדיוק אחרי הורדת הרמה $k+1$ ולכן ההפרש ביניהם לא השתנה, ואילו אם רק באחד מתתי העצים יש קודקוד ברמה התחתונה ואילו בשני לא, אזי העץ שבו יש כזה קודקוד הוא העץ הגבוה יותר, ולכן הורדת הקודקודים ברמה זו רק מקטינה את ההפרש בין גבהי שני תתי העצים.

התחלנו עם עץ בגובה $k+1$ ובו n_{k+1} קודקודים והורדנו לפחות קודקוד אחד ממנו (שכן חייב להיות קודקוד אחד לפחות ברמה התחתונה) ולפי הטענה קיבלנו עץ $AVL-2$ בגובה k. לפי הגדרה, חייבים להיות בעץ זה לפחות n_k קודקודים. ולכן $n_k + 1 \leq n_{k+1}$.

טעויות נפוצות:

i. תלמידים רבים טעו טעות שדיברנו עליה פעמים רבות בהקשר של אינדוקציה. הם התחילו מעץ $AVL-2$ בגובה k ואמרו שנוסיף לו כעת קודקודים בשביל לבנות עץ $AVL-2$ בגובה $k+1$.
חזרנו על הבעיה בטיעון כזה שוב ושוב: ראו תירגול 9 בהקשר של אינדוקציה. יש להוכיח שבאמת ניתן להגיע לכל עץ $AVL-2$ מינימלי בגובה $k+1$ על ידי הוספת קודקודים לעץ מינימלי בגובה k. זה דורש הוכחה, ולא עובד באופן כללי. כמובן שאם היה אפשר לטעון את זה בקלות ההוכחה וכל הסעיף היו הופכים לטריויאליים. מי שטעה טעות זו, כדאי מאוד שיחזור על התירגול שהסביר את הבעיה הזו!

ii. טענות שעץ $AVL-2$ הוא עץ כמעט שלם, טעות שמעידה על חוסר הבנה של עצי AVL.

iii. טעות נפוצה נוספת היתה להגיד "נוכיח באינדוקציה" מבלי לנסח במדויק את הטענה שאותה מוכיחים. הרי הטענה שנתבקשת להוכיח בשאלה אינה טענה שמתייחסת ל k מסוים אלא לכל ה k־ים. אי ניסוח מדויק בדרך כלל הוביל לבעיות בהמשך. באופן כללי זה דבר שיש להקפיד עליו מאוד ־ ניסוח הטענה במדויק הוא שלב הכרחי.

iv. טעות נפוצה נוספת היתה שימוש בנוסחת הנסיגה $n_k = n_{k-1} + n_{k-3} + 1$ מבלי להוכיח אותה. ההוכחה שלה (כמו שנראה בסעיף הבא וכמו שראינו בכיתה) כמובן מתבססת על המונוטוניות של n_k וזה בדיוק מה שאתם מנסים להוכיח! כלומר, להשתמש בנוסחת הנסיגה זה בעצם להשתמש במה שמנסים להוכיח בשביל להוכיח אותו... זה כמובן מעגלי ולא ניתן לעשות זאת.

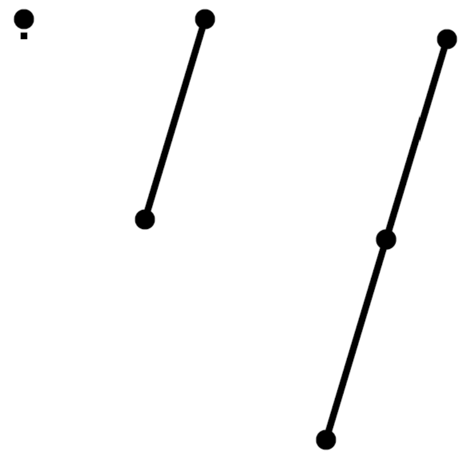
(ב) כפי שראינו בסעיף הקודם שני תתי העצים של העץ הם עצי $AVL-2$ חוקיים ומינימאליים ביחס לגובהם (הוכחת הטענה שהוכחנו בסעיף הקודם עובדת ללא שינוי גם עבור תת העץ השני, כמובן שיש להתחשב בגובה תת העץ).
כפי שראינו אחד מתתי העצים של עץ מינימאלי בגובה k חייב להיות בגובה $k-1$. בכדי שדרישת ה־ $AVL-2$ תתקיים בשורש והעץ הכולל יהיה בגובה $\geq k$, תת העץ השני יכול להיות בגובה $k-1, k-2, k-3$. אילו תת העץ היה בגובה גדול מ־ $k-1$ אז העץ כולו היה בגובה גדול מ־ k ואילו תת העץ היה בגובה קטן מ $k-3$ אז דרישת ה $AVL-2$ הייתה מופרת בשורש. ממונוטונית n_k נובע כי $n_{k-3} < n_{k-2} < n_{k-1}$. מכאן תת העץ השני חייב להיות עץ מינמאלי בגובה $k-3$ כדי שהעץ הכולל יהיה מינימאלי. שוב, אם נניח בשלילה שהוא לא כזה (מינימאלי בגובה $k-3$), נוכל להחליפו בכזה, לשמור על דרישת ה־ $AVL-2$ ולהקטין את מספר הקודקודים הכולל בעץ, בסתירה למינימאליות.
לאור מה שהוכחנו n_k מקיימת את נוסחת הנסיגה הבאה:

$$n_k = n_{k-1} + n_{k-3} + 1$$

מכיוון שבנוסחא מופיע $k-3$ נזדקק לשלושה תנאי ההתחלה הבאים:

$$n_0 = 1,\ n_1 = 2,\ n_2 = 3$$

המתקבלים מכך שהעצים הבינאריים המינימאליים בגובה $0,\ 1,\ 2$ הם עצי $AVL-2$ חוקיים:



טעויות נפוצות:

i. נוסחת נסיגה לא נכונה, אי שימוש במונוטוניות n_k כאשר טוענים שתת העץ השני הוא בגובה $k-3$, תנאי התחלה לא מספיקים או לא נכונים וכו...

ii. תלמידים רבים לא הזכירו את העובדה שהוכחת נוסחת הנסיגה מסתמכת על מונוטוניות של n_k על מנת להסביר מדוע מופיע n_{k-3} ולא למשל n_{k-1} או n_{k-2}.

iii. הרבה תלמידים שכחו את קודקוד השורש.

iv. תלמידים רבים לא הבינו שנוסחת נסיגה שיש בה שימוש ב n_{k-3} מחייבת שלושה תנאי בסיס ולא אחד או שניים.

(ג) נתון עץ $AVL-2$ בגובה k. מהגדרת n_k נובע כי $n_k \le n$. מהסעיף הקודם נובע כי $n_k = n_{k-1} + n_{k-3} + 1 \ge 2n_{k-3} + 1 \ge 2n_{k-3}$ כאשר האי שיוויון השמאלי נובע ממונוטוניות.

דרך א: על ידי שיטת האיטרציה נקבל $n_k \ge 2n_{k-3} \ge 4n_{k-6} \ge \ldots \ge 2^i n_{k-3i}$. עבור $i = \lfloor k/3 \rfloor$ נקבל $n_k \ge 2^{\lfloor k/3 \rfloor} n_{k\,(mod\,3)}$, מכך ש $k\,(mod\,3) \in \{0,1,2\}$. החלק השלם של $k/3$ גדול מ $k/3-1$ וכמו כן $n_0, n_1, n_2 \ge 1$. קיבלנו ש $n_k \ge 2^{\lfloor k/3 \rfloor} n_{k(mod3)} \ge 2^{k/3}/2$. נוציא log משני האגפים ונקבל ממונוטוניות פונקצית הלוגריתם כי $k/3 - 1 \le log(n_k)$. מכאן $k = O(log(n_k)) = O(log(n))$ כאשר השיוויון האחרון נובע מכך ש $n_k \le n$.

דרך ב: נוכיח באינדוקציה שלמה על k ש $k = O(log(n_k))$ כלומר שקיים קבוע $C > 0$ כך ש $k \le C \cdot log(n_k)$.
בסיס האינדוקציה: עבור $k = 0,\ 1,\ 2$ מספיק לבחור $C \ge 2$ בכדי שהטענה תתקיים.
שלב האינדוקציה: נניח עבור כל $l < k$ כי $l \le C \cdot log(n_l)$ ונוכיח עבור k:

$$log(n_k) \ge log(2n_{k-3}) \ge \frac{k-3}{C} + log(2) = \frac{k-3+C}{C}$$

כאשר האי שיוויון השמאלי נובע ממונוטוניות פונקצית הלוגריתם וממה שהראנו קודם, והאי שיוויון הימני מהנחת האינדוקציה. מכאן:

$$k \le C \cdot log(n_k) + 3 - C$$

עבור בחירה של $C \ge 3$ מתקיים $k \le C \cdot log(n_k) \le C \cdot log(n)$ כלומר $k = O(log(n))$.

טעויות נפוצות:

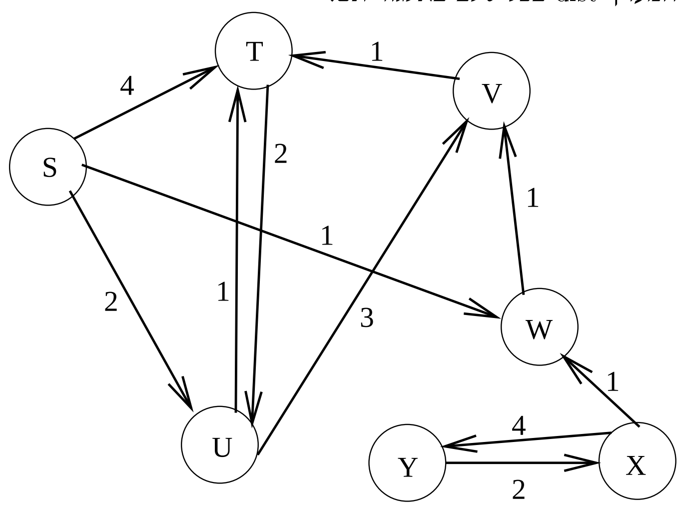
i. אי התחשבות בגורם $n_{k\,(mod\,3)}$, הוכחות חלקיות, הוכחות בכיוון ההפוך כלומר סטודנטים שההוכחה שלהם הראתה כי $log(n) = O(k)$, עובדה נכונה עבור כל עץ בינארי ולאוו דווקא עץ $AVL-2$, וכו...

ii. תלמידים רבים התבלבלו בכיוון החסם שנדרשתם להוכיח. הם בעצם התעלמו מכך שמדובר בעץ $AVL-2$ וניסו להוכיח את הטענה לעץ בינארי כללי. הם חשבו שמתוך הטענה כי לעץ בינארי בגובה k יש לכל היותר 2^k קודקודים כלומר $n \leq 2^k[*]$ תנבע הטענה. אבל כפי שניתן לראות אם לוקחים לוג בשני האגפים של$[*]$ מקבלים חסם תחתון ולא עליון על k. טעות זו היתה מתגלה בקלות על ידי רוב התלמידים ששגו בה, אם היו רושמים את הטענה וההוכחה באופן מדויק ולא בנפנוף ידיים.

iii. תלמידים רבים לא התחשבו בתנאי ההתחלה בפתרון הרקורסיה. למעשה, יש כאן שלושה מקרים, לפי שלושת המקרים של השארית של k בחלוקה בשלוש. כל שארית מובילה לתנאי התחלה אחר בבסיס הרקורסיה, וצריך לבדוק שהחסם שאתם כותבים לגבי n_k נכון לכל המקרים. הורדנו על כך מעט מאוד ניקוד, אבל שימו לב לכך בהמשך.

3. אלגוריתם דייקסטרה (Dijkstra):

(א) הרץ את האלגוריתם על הגרף הבא החל מצומת S, כלומר כתוב ייצוג רשימת שכנויות של הגרף, והרץ את האלגוריתם על הגרף לפי סדר הקדקודים והצלעות בייצוג שכתבת. על מנת להדגים את ההרצה, כתוב מה מצב המערך dist ב**כל** שלב באלגוריתם.



(ב) כתוב פסיאודו־קוד לאלגוריתם מבוסס דייקסטרה המקבל (G, w, s) (כאשר G הוא גרף, s קדקוד בגרף ו $w: E \to \mathbb{R}$ פונקצית משקל על הצלעות), כך ש $\forall e \in E[G]: w(e) > 0$, ומחזיר מערך NumberDist בגודל $V[G]$ כך שעבור כל קדקוד v מתקיים ש NumberDist[v] מכיל את **מספר** המסלולים הקצרים ביותר מהקדקוד s לקדקוד v. כלומר, האלגוריתם סופר לכל קדקוד v, כמה דרכים באורך מינימלי קיימות על מנת להגיע לקדקוד v מ s. **הסבר במלים כיצד פועל האלגוריתם שכתבת, בהתייחס לפסאודו־קוד שכתבת, והסבר מהו זמן הריצה** (ללא הוכחה פורמלית).

(ג) הראה שהאלגוריתם מסעיף א' נכשל אם מרשים לתת לחלק מהצלעות משקל 0. נמק!

3. **פתרון:**

(א) נעבוד עם רשימת שכנויות בסדר אלפאבתי עולה.

$iteration$ \ $vertex$	S	T	U	V	W	X	Y
0	0	∞	∞	∞	∞	∞	∞
1(S)	0	4	2	∞	1	∞	∞
2(W)	0	4	2	2	1	∞	∞
3(U)	0	3	2	2	1	∞	∞
4(V)	0	3	2	2	1	∞	∞
5(T)	0	3	2	2	1	∞	∞
6(X)	0	3	2	2	1	∞	∞
7(Y)	0	3	2	2	1	∞	∞

(ב) האלגוריתם הבא מקבל גרף, פונקצית משקל ומקור ומחזיר את מספר המסלולים הקצרים ביותר מהמקור לכל קדקד בגרף:

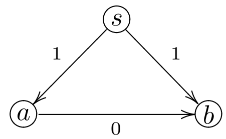
Algorithm 1 CountDistances(G,w,s)

for each $u \in V[G]$ **do**
  $\text{dist}[u] = \infty$
  $\text{countDist}[u] = 0$
end for
$\text{dist}[s] \leftarrow 0$
$\text{countDist}[s] \leftarrow 1$
$Q \leftarrow$ Build-Min-Heap[V sorted by dist]
while Q is not empty **do**
  $v \leftarrow$ Extract-Min$[Q]$
  for each w adjucent to v **do**
    if $\text{dist}[w] > \text{dist}[u] + w(v,w)$ **then**
      $\text{dist}[w] \leftarrow \text{dist}[u] + w(v,w)$
      Decrease-Key$[Q,w]$
      $\text{countDist}[w] \leftarrow \text{countDist}[u]$
    else if $\text{dist}[w]=\text{dist}[u] + w(v,w)$ **then**
      $\text{countDist}[w] \leftarrow \text{countDist}[w]+\text{countDist}[u]$
    end if
  end for
end while

זמן הריצה של האלגוריתם שווה לזמן הריצה של האלגוריתם של דייקסטרה, כלומר $O\big((|E|+|V|)\log(|V|)\big)$, היות וגם אלגוריתם זה רץ בלולאה על כל הצלעות, ובכל איטרציה מבצע שליפה מערימת מינימום שגודלה לכל היותר $|V|$.

האלגוריתם סופר את מספר המסלולים הקצרים ביותר לכל קדקד כך: ראשית, מספר המסלולים הקצרים ביותר אל המקור הוא 1. מכאן, האלגוריתם ממשיך על פי הכלל שמספר המסלולים הקצרים ביותר לקדקד הוא סכום המסלולים הקצרים ביותר להורה שלו בכל מסלול קצר ביותר אליו. לפיכך, אם מתגלה מסלול קצר ביותר ראשון, מספר המסלולים הקצרים ביותר מאותחל למספר המסלולים הקצרים ביותר אל ההורה. אם מתגלה מסלול קצר ביותר נוסף, מספר המסלולים הקצרים ביותר אל ההורה החדש שהתגלה מתוסף למספר המסלולים הקצרים ביותר הידועים.

(ג) נכונות האלגוריתם נשענת על טענה אינדוקטיבית שאומרת שכל המסלולים הקצרים ביותר במשקל **<u>קטן מ-</u>**l מתגלים לפני שמתגלה מסלול קצר ביותר באורך l. הטענה נכונה רק בגרפים בהם אין קשתות באורך 0. הנה דוגמה עליה האלגוריתם יכשל:



אם האלגוריתם ישלוף את b לפני שישלוף את a (מה שעשוי לקרות) אז הוא יספור רק אחד משני המסלולים הקצרים ביותר שיש מ־s ל־b.
בנוסף, בגרף עם מסלול מעגלי במשקל 0 עשוי להיות מספר אינסופי של מסלולים קצרים ביותר, ואילו האלגוריתם הנתון תמיד חוזר על מספר מסלולים קצרים ביותר סופי לכל קדקד.

שגיאות נפוצות:

i. איתחול ועידכון מספר המסלולים ב־1 במקום במספר המסלולים שמובילים להורה שלו.

ii. עידכון מספר המסלולים רק כשמתגלה מסלול קצר יותר, או רק כשמתגלה מסלול שווה אורך, או איחוד שני המקרים.

iii. דוגמה נגדית לא מספיק מפורטת. צריך לכתוב במפורש מה הפלט של האלגוריתם ואיך הוא שונה מהפלט הרצוי.

iv. תיאור שגוי של ריצת האלגוריתם (למשל טענה שגויה שהוא לא עוצר) .

v. רשימת שכנויות חסרה או שגויה.

vi. חישוב שגוי או חסר של הסיבוכיות. למשל, היעדר הסבר מדוע הסיבוכיות נשארת כשל דייקסטרה, הסמטת ה־$\log(|V|)$, טענה שההבדל בין זמני הריצה הוא $o(1)$ (כאשר למעשה ההבדל הוא ב־$o(1)$ פעולות שנוספו ללולאה, ולכן בפקטור קבוע).

vii. איתחול שגוי של $NumberDist[s]$ ל־0.

viii. היעדר פסיאודו קוד.

ix. שימוש בדייקסטרה וחישוב מספר המסלולים הקצרים ביותר באמצעותו (במקום בעזרת אלגוריתם שמבוסס עליו, כלמור שעובד כמוהו), ומעבר על הקדקדים לא על פי מרחקם מ־s (מה שמוביל לקוד שגויי).