תרגול - אלגוריתמים חמדניים

שאלה 1

יהונתן מוכר בחנות ירקות. המטבעות שיש ברשותו הן: שקל, 5 שקלים, 10 שקלים ו-25 שקלים (מטבע חדש שהוציאו). על יהונתן להחזיר עודף של C שקלים לקונה.

(a) מצאו אלגוריתם חמדן אשר יעזור ליהונתן להחליט אילו מטבעות להחזיר כך שמספר המטבעות שיוחזרו יהיה מינימאלי.

לדוגמא: עבור C=20 יהונתן יחזיר 2 מטבעות של 10 שקלים (ולא למשל מטבע של 10 שקלים ו-2 מטבעות של 5 שקלים).

הוכיחו את נכונות אופטימליות האלגוריתם ונתחו את סיבוכיות זמן הריצה של האלגוריתם.

(b) האם האלגוריתם יעבוד גם במקרה שערכי המטבעות הם: שקל, 7 שקלים, 10 שקלים, 25 שקלים. הוכיחו שכן או הראו דוגמא נגדית.

פתרון

A. האלגוריתם:

נסמן את אוסף סוגי המטבעות ב-$C=\{c_1, c_2, \ldots, c_k\}$ מקטן לגדול (כלומר, ספציפית במקרה שלנו: $c_1=1, c_2=5, c_3=10, c_4=25$). את הערך שצריך לצבור נסמן ב-S ואת וקטור הכמויות של המטבעות שעלינו למצוא נסמן ב-N.

$i \leftarrow k$

הגדר וקטור N בגודל k ואפס אותו

כל עוד $S > 0$ בצע

$N[i] \leftarrow \lfloor S/c_i \rfloor$

$S \leftarrow S - c_i \cdot N[i]$

$i \leftarrow i - 1$

סיבוכיות: הלולאה מבצעת עד k איטרציות ואיטרציה אחת מתבצעת ב-$O(1)$, לכן סה״כ $O(k)$.

נכונות: נשים לב, כי האלגוריתם הנ״ל נכון רק בתנאי שאוסף סוגי המטבעות מקיים: לכל $1 \le i < k$ מתקיים $c_i \le 2c_{i+1}$. כעת, נוכיח את נכונות האלגוריתם:

בהינתן סכום S שיש לפרוט למטבעות, נסמן ב-M_A את אוסף המטבעות (קבוצה עם חזרות) שהאלגוריתם יוצר. ויהי M^* אוסף המטבעות המינימלי האפשרי – פתרון אופטימלי. נניח בדרך השלילה ש-$|M_A| > |M^*|$ (כלומר, שהפתרון של האלגוריתם אינו אופטימלי).

נמיין את המטבעות בכל אחת מהקבוצות הנ״ל מגדול לקטן. יהיה i האינדקס הקטן ביותר של האיברים בשתי הסדרות אשר אינם שווים. נסמן אותם m_i ו-m_i^* בהתאמה. לא יתכן ש-$m_i<m_i^*$, כי אז האלג׳ היה בוחר את m_i^* במקום m_i, הרי בכל שלב האלג׳ בוחר במטבע מקסימלי האפשרי. לכן מתקיים $m_i>m_i^*$. לשם פשטות ובה״כ נניח כי $m_i=25$ (לכל מקרה אחר הוכחה אף יותר פשוטה).
נסמן ב-M^*_i את המשך הסדרה M^* החל מ-m_i^*. נשים לב, שהסכום של M^*_i הוא לפחות 25, אך אין בה מטבע של 25. כי אם היה בהמשך מטבע 25, שהוא הכי גדול בערכו, הוא היה צריך להיבחר עכשיו. אבל $m_i=25$ וגם $m_i>m_i^*$. לכן, ב- M^*_i אין בה מטבע של 25.
אם בתוך M^*_i קיימת קבוצת מטבעות שסכומה בדיוק 25, אז נחליף אותה במטבע אחד של 25 וכך נשפר את הפתרון האופטימלי – סתירה. אחרת, M^*_i חייבת להכיל לפחות 3 מטבעות של 10 אותם ניתן להחליף בשני מטבעות של 25 ו-5, וזה גם משפר את הפתרון האופטימלי – סתירה.

B. במקרה הנתון האלגוריתם לא מוצא את המינימום, כי עבור S=14, למשל, הוא יחזיר 5 מטבעות: 10, 1, 1, 1, 1. אולם, ניתן להרכיב את הסכום בפחות מטבעות: 14=7+7.

<u>שאלה 2</u>

בעיית התרמיל (Knapsack problem)

נתון תרמיל שלו קיבולת משקל W ונתונים n פריטים $a_1,a_2,a_3,\ldots,a_n$. לכל פריט a_i מיוחס משקל $w(a_i)$ ורווח $p(a_i)$. המטרה היא לבחור תת קבוצה של פריטים, כך שמשיגים רווח מקסימלי מבלי להפר את מגבלת המשקל (קיבולת התרמיל).

בעית התרמיל בשברים

תנאי נוסף (הקלה) – מותר לחלק (לשבור) את הפריטים. בחלוקת פריט ביחס מסוים הרווח מתחלק באותו יחס.
הציעו אלגוריתם חמדני לפתרון אופטימלי של הבעיה.

<u>פתרון</u>:

<u>רעיון</u>:

נגדיר רווח סגולי (רווח ליחידת משקל) של כל פריט כיחס בין רווח הפריט למשקלו, כלומר לכל a_i $u(a_i)=\dfrac{p(a_i)}{w(a_i)}$.

ברור שככל שהרווח הסגולי גבוה יותר, הפריט משתלם יותר.

<u>אלגוריתם</u>:

1. נמיין את כל הפריטים לפי הרווח הסגולי בסדר לא עולה (מגבוה לנמוך).
2. נמלא את התרמיל בפריטים לפי הסדר הזה כל עוד הדבר אינו גורם לחריגה מקיבולת התרמיל.
3. אם וכאשר הפריט האחרון a_t שמנסים להכניס לתרמיל לא נכנס משום שמשקלו $w(a_t)$ גדול מהמרווח הפנוי B שנשאר בתרמיל, אז נחתוך מהפריט חלק במשקל B ונכניס אותו לתרמיל. בזה נמלא את התרמיל עד "אפס מקום".

<u>נכונות</u>:

יהי S_A הפתרון (קבוצת הפריטים וחלקיהם) המיוצר ע"י האלגוריתם החמדני.
ונניח שהפריטים בקבוצה ממוינים בסדר לא עולה של הרווח הסגולי שלהם:
$S_A=\{x_1,x_2,x_3,\dots,x_k\}$. כמו כן, יהי $S^*=\{z_1,z_2,z_3,\dots,z_m\}$ פתרון אופטימלי כלשהו (וגם פה הפריטים ממוינים בסדר לא עולה של רווח סגולי).

נתבונן ב-i מינימלי עבורו $x_i \neq z_i$. לפי האלגוריתם x_i נבחר בזכות רווח סגולי מקסימלי, לכן מתקיים $\frac{p(x_i)}{w(x_i)} \geq \frac{p(z_i)}{w(z_i)}$.

<u>**אם**</u> <u>$w(x_i)=w(z_i)$</u>, אז ניתן להחליף את הפריט z_i בפתרון S^* בפריט x_i, ובכך לשפר (או לפחות לא להוריד) את הרווח הכולל.

<u>**אם**</u> <u>$w(x_i)<w(z_i)$</u>, אז נחתוך מהפריט z_i חלק במשקל $w(x_i)$ ונבצע החלפה עם החלק הזה.

ולהיפך, <u>**אם**</u> <u>$w(x_i)>w(z_i)$</u>, אז נחתוך חלק מ-x_i במשקל $w(z_i)$ ונבצע החלפה.

ע"י סדרת פעולות כנ"ל ניתן "להפוך" את S^* ל-S_A ללא הפסד ברווח הכולל.
לכן הפתרון של האלגוריתם הינו אופטימלי.

שאלה 3

בעיית כיסוי בצמתים על עצים

בהינתן גרף לא-מכוון $G(V,E)$ **כיסוי בצמתים** (Vertex Cover) של G הוא קבוצת צמתים $V'\subseteq V$ כך שלכל קשת $(u,v)\in E$ לפחות אחד מהצמתים u,v שייך ל-V'.

המטרה: למצוא כיסוי בצמתים מינימום.

גם בעיה זו "קשה" עבור גרפים כללים, אך ניתנת לפתרון אופטימלי ע"י אלגוריתם חמדני במקרה שהגרף הנתון הוא **עץ**.

פתרון:

רעיון:

נסתכל על קשת (u,v) כך ש-v עלה (צומת שדרגתו 1). כל כיסוי בצמתים חייב להכיל את u או v. עם זאת משתלם יותר לבחור את u לכיסוי משום ש-v מכסה רק את הקשת (u,v) ואילו u עשוי לכסות קשתות נוספות.

האלגוריתם:

1. $V'\leftarrow\varnothing$
2. כל עוד קיים עלה v בגרף:
3. הוסף ל-V' את u – השכן של v.
4. מחק מ-E את כל הקשתות הנוגעות ב-u (ועדכן את דרגות הצמתים המתאימים).

סיבוכיות: ניתן לשמור את הצמתים שדרגתם 1 בתור. בכל צעד נשלוף צומת מראש התור, אם דרגתו 1 (יתכן שתרד ל-0 במהלך שהותו בתור) נבצע את שלבים 3 ו-4 באלגוריתם, תוך הוספת צמתים שדרגתם ירדה ל-1 לתור. הסיבוכיות תהיה לכן $\mathrm{O}(|V|+|E|)=\mathrm{O}(|V|)$.

נכונות: יש להראות שקיבלנו כיסוי בצמתים ושזהו כיסוי מינימום.

בכל עץ ישנם לפחות שני עלים. לאחר מחיקת קשתות מקבלים יער (עץ אחד או יותר). לכן כל עוד יש קשתות בגרף ישנם צמתים מדרגה 1. לכן כאשר האלגוריתם עוצר כל הקשתות נמחקו. כל קשת שנמחקה מכוסה ולכן התקבל כיסוי בצמתים.

נסמן ב- v_i את הצומת ה- i שנוסף ל- V'.

טענה: קיים כיסוי אופטימלי שמכיל את הצמתים $v_1, v_2, \ldots, v_i$.

הוכחה: באינדוקציה על i. בסיס: הצומת הראשון שנבחר v_1 הוא שכן של עלה v. נסתכל על כיסוי אופטימלי שלא מכיל את v_1, ולכן מכיל בהכרח את v (אחרת הקשת ביניהם לא מכוסה). אם נחליף את v ב- v_1 נקבל כיסוי (v כיסה רק את הקשת (v, v_1) וכעת היא מכוסה ע"י v_1) שגודלו זהה לגודל הכיסוי המקורי ולכן גם הוא אופטימלי.

צעד: יהא v_i הצומת ה- i ($i > 1$) שנוסף ל- V' ויהא v העלה שגרם להוספת v_i. לפי הנחת האינדוקציה קיים כיסוי אופטימלי שמכיל את הצמתים $v_1, v_2, \ldots, v_{i-1}$. אם כיסוי זה מכיל גם את v_i – סיימנו. אחרת, הכיסוי מכיל בהכרח את v. כל הקשתות ש- v מכסה, למעט (v, v_i), מכוסות ע"י $v_1, v_2, \ldots, v_{i-1}$, כי אחרת v לא היה הופך לעלה. לכן, ניתן להחליף את v ב- v_i ולקבל כיסוי אופטימלי שמכיל את $v_1, v_2, \ldots, v_i$.

לכן הכיסוי המתקבל ע"י האלגוריתם החמדן הינו אופטימלי.

<u>שאלה 4</u>

בקיוסק של ראובן החליטו על מבצע. כל מי שקונה זוג מוצרים מקבל את הזול מבניהם חינם. לאחר שבחרנו $2n$ מוצרים $a_1, a_2, a_3, \ldots, a_{2n}$ (המחיר של כל אחד הוא $p(a_i)$), ברצוננו לסדר אותם בזוגות ע״מ לקבל הנחה מרבית. הציעו אלגוריתם המניב זוגות של מוצרים כך שהסכום הכולל שנאלץ לשלם על המוצרים יהיה מינימלי האפשרי.

פתרון, הוכיחו נכונות והראו סיבוכיות.

<u>פתרון:</u>

<u>אלגוריתם</u> :

1. נמיין את $2n$ המוצרים לפי המחירים.
2. נחלק את הסדרה המתקבלת לזוגות (ראשון-שני, שלישי-רביעי וכך הלאה).

<u>נכונות</u> :

לשם נוחות, נמספר את הפריטים בסדר עולה של מחירם. כלומר, a_1 יהיה הפריט הזול ביותר, a_2 – הפריט הזול הבא (הזול מכולם חוץ מ-a_1) וכך הלאה עד הפריט היקר ביותר a_{2n}.

<u>טענת עזר</u> : קיים פתרון אופטימלי בו a_1 ו-a_2 נמצאים באותו זוג.

<u>הוכחת טענת עזר</u> :

יהיה S^* פתרון אופטימלי. יהיה a_i הפריט שנמצא בזוג עם a_1 בפתרון S^*.

אם $i > 2$, אז נתבונן גם בזוג בו נמצא a_2. נסמן את הפריט השני בזוג הזה ב-a_j.

מאחר והפריטים a_1 ו-a_2 הינם הכי זולים בקבוצה, אז בני הזוג יקרים מהם, כלומר $p(a_i), p(a_j) \geq p(a_1), p(a_2)$. לכן התרומה של הזוגות האלה למחיר הכולל היא $p(a_i) + p(a_j)$.

אם נחליף בין a_2 ל-a_i, נקבל את הזוגות (a_1, a_2) ו-(a_i, a_j). המחיר של הזוג (a_1, a_2) הינו $p(a_2)$, ומחירו של הזוג (a_i, a_j) הוא $\max\{p(a_i), p(a_j)\}$. לכן מחירו של הפתרון החדש הינו

$$P(S^*) - \left(p(a_i) + p(a_j)\right) + p(a_2) + \max\{p(a_i), p(a_j)\} \leq P(S^*)$$

כלומר, ע״י ההחלפה לא העלינו את מחיר הפתרון, לכן קיבלנו פתרון אופטימלי חדש המקיים את הטענה.

טענה: האלגוריתם הנ"ל מייצר פתרון אופטימלי, כלומר מחירו הכולל הוא המינימלי האפשרי.

הוכחה: נוכיח את הטענה באינדוקציה על n.

בסיס: עבור n = 1 מדובר בשני פריטים בלבד. המחיר של הפתרון הוא המחיר של הפריט היקר מבין השניים, שזה אכן המינימום במקרה זה.

הנחה: עבור n < k הפתרון הינו אופטימלי.

צעד: נוכיח עבור n = k.

יהיה S_A הפתרון המתקבל מהאלגוריתם. נשווה את מחירו עם מחיר הפתרון האופטימלי S^* אשר מכיל את הזוג (a_1, a_2). מאחר וגם S_A מכיל את (a_1, a_2), אז ניתן להוריד את הזוג משני הפתרונות ולקבל פתרונות עבור הבעיה ללא הפריטים a_1 ו-a_2. אבל בבעיה המוקטנת n < k, לכן עפ"י הנחת האינדוקציה, האלגוריתם פותר אותה אופטימלית, כלומר, $P(S_A \backslash \{a_1, a_2\}) \le P(S^* \backslash \{a_1, a_2\})$. מכאן נקבל

$$P(S_A) = P(S_A \backslash \{a_1, a_2\}) + p(a_2) \le P(S^* \backslash \{a_1, a_2\}) + p(a_2) = P(S^*)$$

כלומר, הפתרון של האלגוריתם אינו יקר מהפתרון האופטימלי, לכן גם הוא אופטימלי.

שאלה 5

תזמון משימות

נתון סט T של n משימות עם זמני התחלה $s_1, s_2, s_3, \ldots, s_n$ וזמני סיום $f_1, f_2, f_3, \ldots, f_n$. רוצים לתזמן את כל המשימות תוך הקצאת מספר מינימלי של מכונות. (מבלי שיווצרו קונפליקטים).

פתרון:

אלגוריתם:

נמיין את המשימות לפי סדר עולה של זמני ההתחלה.

לכל משימה i אם יש מכונה פנויה, היא תוקצה.

אחרת – יש להקצות מכונה חדשה.

חזור על האלגוריתם עבור שאר המשימות.

<u>האלגוריתם באופן פורמאלי</u>:

```
TaskSchedule(T)
    ➔ Input: A set T of tasks with start time and end time
    ➔ Output: A non-conflicting schedule of the T tasks
    M ← 0
    While T ≠∅ do
        removeMin(T)
        if ∃ machine j with no conflicts
            then   schedule(i,j)
            else
                   m ← m +  1
                   schedule(i,m)
```

<u>משפט</u>: בהינתן בעיית תיזמון המשימות עם n משימות, האלגוריתם לעיל מספק תיזמון עם מספר מכונות מינימלי.

<u>הוכחה</u>:

נניח שהאלגוריתם לעיל, מייצר פתרון עם k מכונות. ונניח ש- k מייצג את המכונה האחרונה שהוקצתה וש- i מייצג את המשימה הראשונה שמבוצעת ע"י המכונה k. מהאופן שבו פועל האלגוריתם, הקצאתה של מכונה k בוצעה בעת תיזמון משימה i מפני שלכל שאר המכונות תוזמנו כבר משימות. מאחר והמשימות מסודרות לפי זמני ההתחלה שלהן, כל יתר המשימות המקבילות זמני ההתחלה שלהן קודמים ל- s_i. מאחר ונוצר קונפליקט – זמני הסיום שלהן גדולים מ- s_i. מכאן ש-$k-1$ המשימות הללו, לא רק בקונפליקט עם המשימה ה-i אלא גם בקונפליקט אחת עם השנייה. מכאן שישנה קבוצה כלשהי של k משימות בתוך T, שמתבצעות בו זמנית. לכן לא ניתן להקצות פחות מ-k מכונות.

<u>שאלה 6</u>

בתחילת השיעור למדנו את האלגוריתם לפתרון בעיית בחירת הפעילויות.

מר באג, המתכנת המפוזר טעה, וחיבר את האלגוריתם הבא. תאר מה מחשב האלגוריתם הנ"ל ומה הסיבוכיות שלו?

Greedy_Activitity (*A*)

Sort A by $f(a)$ in ascending order

$G \leftarrow \{a_1\};\ i \leftarrow 1;$

for $j \leftarrow 2$ to $|A|$ do:

if $f(a_i) \le s(a_j)$ then

$G \leftarrow G \cup \{a_j\};$

$i \leftarrow i+1$

endIf;

endFor;

return *G*.

<u>פתרון</u>:

יש טעות בעדכון j: במקום לבצע $i \leftarrow j$ האלגוריתם פשוט מקדם את i ב-1, לכן בשלב הבא הבדיקה $f(a_i) \le s(a_j)$ מבצעת השוואה עם זמן סיום לא נכון. כך הטעות הזו עלולה לגרום לחפיפות בתוך הקבוצה A.

הסיבוכיות של האלגוריתם הינה $O(n \log n)$.

<u>שאלה 7</u>

בחברת "המקודדד" התקבלה משימה לקודד באופן אופטימלי את 5 האותיות a, b, c, d, e, ששכיחות הופעתן מצויינת בטבלה למטה. אולם, קיימת מגבלה שאורך הקוד עבור אות בודדת לא יעלה על 3 ביטים.

הצע קוד מתאים.

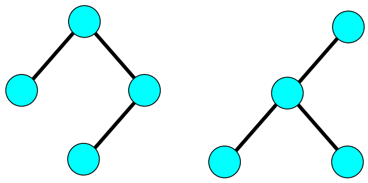
טבלת שכיחויות:

אות	שכיחות
a	10000
b	1000
c	100
d	10
e	1

<u>פתרון</u>:

מאחר ועץ Huffman רגיל הינו בעל גובה 4, יש לחפש עץ אחר. כלומר, יש לבנות עץ T <u>בינרי מלא בגובה 3</u>, עבורו הסכום $B(T)=\sum_{c\in\Sigma} f(c)\cdot d_T(c)$ יהיה מינימלי.

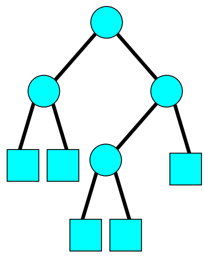
לעץ יהיו 5 עלים. אם נוריד את כל העלים, נישאר עם עץ בינרי בעל 4 צמתים וגובה 2. נבדוק את כל האפשרויות של עץ כזה:



a b

נחזיר את העלים לשני העצים, ונשווה ביניהם:

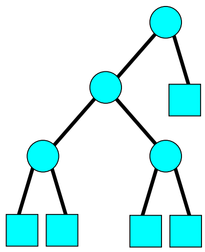
a) בעץ המתקבל



ישנם 3 צמתים בעומק 2 ו-2 צמתים בעומק 3, לכן

$B(T)=2\cdot(10000+1000+100)+3\cdot(10+1)=22233$.

b) בעץ השני



ישנם 4 צמתים בעומק 3 וצומת אחד בעומק 1, לכן

$B(T)=10000+3\cdot(1000+100+10+1)=13333$.

ברור שהעץ השני מניב קוד יותר חסכוני. לכן נשתמש בו. קוד אפשרי אחד:

אות	קוד
a	0
b	100
c	101
d	110
e	111