תרגיל 6- אלגוריתמים 67504:

שמות: דן קורנפלד, מתן קמינסקי

שאלה 1:

סעיף 1:

1. יהא $n \in \mathbb{N}$ המהווה את מספר הימים של משך החופשה, נראה עבור פתרון אופטימלי s^* ועבור הפתרון הנתון s מתקיים: $\frac{s(n)}{s^*(n)} \leqslant 2$.
2. לפי אופן הגדרת האלגוריתם $s(n) = \begin{cases} n & n \leqslant 20 \\ 40 & n > 20 \end{cases}$ וזאת מאחר ואנו לא יודעים מראש את כמות הימים שבהם נשאר.
3. על פי הפתרון האופטימלי נשלם לפי מספר הימים שנשאר לו ידענו מראש. לכן, $s^*(n) = \begin{cases} n & n \leqslant 20 \\ 20 & n > 20 \end{cases}$.
4. עבור $n \leqslant 20$ נראה כי $2 \geqslant 1 = \frac{n}{n} = \frac{s(n)}{s^*(n)}$, עבור היום ה-21: $2 \geqslant 2 = \frac{\overbrace{20}^{\text{ימי גלישה 20}} + \overbrace{20}^{\text{קניית מגלשיים}}}{\underbrace{20}_{\text{מחיר פתרון אופטימלי}}} = \frac{s(n)}{s^*(n)}$. לאחר מכן, עבור כל יום אחרי היום ה-21 נשלם אותו יחס של המחיר כמו ביום ה-21, לכן הקירוב יהיה "2-קירוב"

סעיף 2:

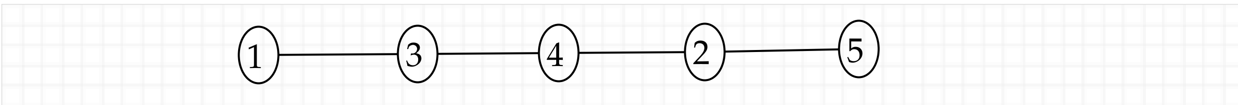
1. נגדיר את הפתרון שהחבר הציע כפונקציה $f(n) = \begin{cases} n & n \leqslant 10 \\ 30 & n > 10 \end{cases}$ נראה שהיחס בין: $c = 3 \geqslant \frac{f(n)}{s^*(n)}$, כלומר נראה שמצב זה הוא "3- קירוב" עבור $n \leqslant 10$: $3 \geqslant 1 = \frac{n}{n} = \frac{f(n)}{s^*(n)}$, עבור $10 < n \leqslant 20$ נקבל:
$$3 = \frac{30}{10} \geqslant \frac{\overbrace{10}^{\text{ימי סקי ראשוניים}} + \overbrace{20}^{\text{קניית מגלשיים}}}{n} = \frac{f(n)}{s^*(n)}$$ עבור הימים $n > 20$
$$3 \geqslant \frac{30}{20} = \frac{\overbrace{10}^{\text{ימי סקי ראשוניים}} + \overbrace{20}^{\text{קניית מגלשיים}}}{20} = \frac{f(n)}{s^*(n)}$$ ולכן אנו רואים שקיים c כך שהיחס בין הפתרון של החבר לפתרון האופטימלי הוא "3-קירוב".

סעיף 3:

1. לפי האלגוריתם המוצעים, ביום ה-50, הפתרון השני (התשלום של החבר) זול יותר מהתשלום של החבר מכיוון ש: אנו נשלם $s(50) = 40$, והחבר ישלם $f(50) = 30$, אולם נראה שהקירוב לא מעיד על טיב התשלום עבור כל מספר ימים של חופשה לדוגמה: עבור היום ה-11, הפתרון הראשון ישלם 11 שקלים, אולם הפתרון של החבר ישלם 30 שקלים.

שאלה 2:

סעיף 1:



1. לפי הפתרון האופטימלי החתך המקסימלי יהיה בגודל 4, אולם בהרצת האלגוריתם החתך המקסימלי יהיה בגודל 3.

סעיף 2:

1. נרצה להציע אלגוריתם לבעיה כך שהאלגוריתם יהיה " $\left(1+\dfrac{1}{k-1}\right)$ - קירוב" לפתרון אופטימלי.
2. נשאב השראה מהאלגוריתם שלמדנו בתרגול.
3. נגיד קבוצה $A_1 = V, A_2 = \ldots = A_k = \varnothing$
4. בכל איטרציה, כשנסתכל על קודקוד v, נרצה למצוא את הקבוצה A_i האידיאלית כך שהעברת הקודקוד v ל-A_i יגדיל הכי הרבה את החתך, אם לא קיים, v ישאר בקבוצה הנוכחית שלו.
5. נחזור על (4) כל עוד האלגוריתם ביצע שינוי בחתך.

הוכחת נכונות: חוקיות, אופטימיזציה, זמן ריצה.

1. חוקיות: מאופן הגדרת האלגוריתם, נחזיר k קבוצות, כך שכל קודקוד נמצא בקבוצה אחת בלבד, ולא נמצא באחרות, ומתקיים: $\cup A_i = V$
2. למת עזר עבור אופטימיזציה (קירוב):

יהי $v \in V$, מספר השכנים של v בקבוצה בה הוא נמצא קטן שווה מ-$\dfrac{d(v)}{k}$

הוכחה: נניח בשלילה שכמות השכנים בקבוצה בה נמצא v גדול מ $\dfrac{d(v)}{k}$, אזי מעקרון שובך היונים אזי קיימת קבוצה בה כמות השכנים קטנה מ $\dfrac{d(v)}{k}$. על פי האלגוריתם היינו מעבירים את v לקבוצה הנ"ל ובכך מגדילים את החתך בסתירה לאופן פעולת האלגוריתם.

הערה: אם בקבוצה של v יש לכל היותר $\dfrac{d(v)}{k}$, אזי שאר הקבוצות יכילו $d(v)\left(1-\dfrac{1}{k}\right)$ לכל הפחות,

כלומר: $|E_{v,C}| \geqslant d(v)\left(1-\dfrac{1}{k}\right)$

3. אופטימיזציה (קירוב): נראה כי האלגוריתם מקיים:" $\left(1+\dfrac{1}{k-1}\right)$ - קירוב" לפתרון אופטימלי.

נגדיר: E_C- כקבוצת הצלעות החוצות את החתך

נגדיר: $E_{v,C}$- קבוצת הצלעות החוצות את החתך היוצאות מקודקוד v.

כמו כן נגדיר: $d(v)$ הדרגה של הקודקוד v.

$$|E_C| = \frac{1}{2}\sum_{v\in V}|E_{v,C}| = \frac{1}{2}\sum_{v\in V}\left(d(v)\left(1-\frac{1}{k}\right)\right) \overset{*}{\geqslant} \frac{1}{2}\left(1-\frac{1}{k}\right)\sum_{v\in V}d(v) = \frac{1}{2}\left(1-\frac{1}{k}\right)\cdot 2\cdot |E| =$$
$$\frac{k-1}{k}\cdot|E| \geqslant \frac{k-1}{k}\cdot OPT \blacksquare$$

4. זמן ריצה:

אתחול האלגוריתם $O(|V|)$- מספור הקודקודים מ-1 עד n, ויצירת הקבוצות $A_1,..,A_k$.
האלגוריתם מבצע $|E|+1$ איטרציות, בכל איטרציה האלגוריתם מוסיף לכל הפחות צלע אחת, לכן ייתכן לכל היותר שהאלגוריתם יוסיף לחתך את כל הצלעות, והאלגוריתם יבצע איטרציה נוספת אשר לא מתבצע שם מעבר של צלע בחתך.
עבור כל איטרציה: נעבור על כל הקבוצות ונבדוק עבור כל קודקוד, אם מספר השכנים שלו בקבוצה הנוכחית גדול ממש ממספר השכנים שלו בקבוצה השניה, נעביר אותו לקבוצה השניה (כך נבצע עבור כל הקבוצות).
לכן: כל איטרציה $|E|$ מעבר על כל הקבוצות k ביצוע הבדיקה עבור קבוצה נתונה וקבוצה פוטנציאלית וקודקוד הינו $|V|+|E|$ לכן: $O(\ |E|\cdot k\cdot(|V|+|E|)\ )$

שאלה 3:

1. נציע אלגוריתם עבור מציאת הקבוצה הב"ת אשר מקיימת "$d+1$-קירוב" לקבוצה המקסימלית.
2. נגדיר $U=\varnothing$ אשר תכיל את הקודקודים בקבוצה הב"ת.
   נגדיר $X=V$ אשר מכיל את קבוצת כל הקודקודים.
3. אלגוריתם: נרוץ בלולאה כל עוד $X\neq\varnothing$ ונבדוק:
   (a) נבחר קודקוד אקראי $v\in X$ נוריד אותו מ X, נוסיף אותו ל-U ונוריד מ-X את כל הקודקודים שהם שכנים ל-v.
   (b) לבסוף נחזיר את U.
4. נכונות:
   (a) חוקיות: עבור כל קודקוד שאנו מוסיפים, אנו מסירים את כל שכניו, ולכן לא יתכן שיהיו בקבוצה U 2 קודקודים בעלי צלע משותפת.
   (b) קירוב: נוכיח שהאלגוריתם הוא"$d+1$-קירוב" לקבוצה המקסימלית:
   נגדיר עבור n המהווה את גודל הקבוצה המקסימלי הב"ת.
   לפי האלגוריתם נוסיף ל-U בדיוק קודקוד אחד בכל איטרציה, כלומר, ככל שכמות האיטרציות קטנה יותר כך גם גודל הקבוצה. תנאי עצירת הלולאה הינו כאשר $X=\varnothing$, לכן ככל שנקטין את X מהר יותר נקבל את הנדרש. נתון כי לכל קודקוד $d(v)\leqslant d$, כלומר אם בכל איטרציה היינו מורידים d שכנים (+ הקודקוד שהאלגוריתם מוסיף) $\Leftarrow$ $d+1$ נקבל את הנדרש. אזי, סה"כ האיטרציות המינימלי שנבצע הינו $\dfrac{n}{d+1}$, וזה יהיה גם גודל הקבוצה, כלומר קיבלנו שגודל הקבוצה המקסימלית באלגוריתם במקרה הגרועה ביותר הינו $\dfrac{n}{d+1}$ מגודל הקבוצה המקסימלית בפועל, לכן הוכחנו את המבוקש.
5. זמן ריצה: יש לכל היותר $|V|$ איטרציות, כך שלכל אחד יש לכל היותר דרגה d ($d+1$ קודקודים שהאלגוריתם נדרש להסיר) ולכן זמן הריצה יהיה $O(d|V|)$.

שאלה 4:

1. נרצה להציע פתרון לבעיה כך שהאלגוריתם שנציע הוא "2 קירוב" לפתרון האופטימלי.
2. נעזר באלגוריתם החמדן שלמדנו בכיתה
3. ראשית, נמיין את כלל האיברים לפי $\frac{v_i}{w_i}$ מהגודל ביותר לקטן ביותר כך ש-v_i הינו הערך של הפריט, ו w_i משקל הפריט.
4. נכניס את כל האיברים לאחר המיון מהאיבר ה-1 עד ה$k-1$ כך שנגדיר את k להיות האינדקס המינימלי ש$W < \sum_{i=1}^{k} w_i$, כלומר כל האיברים מ-1 עד $k-1$ נכנסו לתרמיל באופן מלא.
5. עבור האיבר ה-k יש 2 אופציות:
   (a) $\sum_{i=1}^{k-1} v_i \geqslant v_k$ - במצב זה, נשאיר את ה$k-1$ האיברים בתרמיל ולא נוסיף את האיבר ה-k,
   ונגדיר את וקטור הפתרון x להיות: $x_1 = (\overbrace{1,..,1}^{k-1}, \overbrace{0}^{k},...,0)$
   (b) $\sum_{i=1}^{k-1} v_i < v_k$ - במצב זה, הערך של האיבר ה-k גדול יותר מכל האיברים שכבר בתרמיל, לכן נסיר את כולם ונכניס את האיבר ה-k. נגדיר את וקטור הפתרון x להיות:
   $x_2 = (\overbrace{0,..,0}^{k-1}, \overbrace{1}^{k},...,0)$
   (c) ★ - נגדיר את וקטור הפתרון הכללי להיות x המקיים: $f(x) = max\left\{f(x_1), f(x_2)\right\}$
6. חוקיות: במצב זה אנו מחזירים וקטור בגודל n, כמספר האיברים בבעיה, כך שלכל אחד מהאיברים הערך יהיה $\{0,1\}$, 1 עבור נכנס לתרמיל, 0 אחרת. מאופן פעולת האלגוריתם, האלגוריתם לא יוסיף איבר לתרמיל אם סה"כ המשקל עולה על W. כמו כן, ידוע שעבור כל איבר i מתקיים $w_i \leqslant W$ לכן הפתרון יהיה חוקי.
7. אופטימליות:
   נגדיר $f(x)$- הפתרון שהאלגוריתם שהצענו מחזיר.
   נגדיר $f\left(x^*\right)$- הפתרון שהאלגוריתם האופטימלי מחזיר
   נגדיר $f\left(z^*\right)$- הפתרון שהאלגוריתם האופטימלי-השברי מחזיר, נשים לב שמתקיים $f\left(x^*\right) \leqslant f\left(z^*\right)$ עבור פתרון אופטימלי של הבעיה השברית, נזכיר כי קיים $k \in [n]$ כך שלכל האיברים מ-1 עד $k-1$ נכניס את ה$x_i = 1$, עבור האיבר ה-k, נחזיר את המשקל המתאים שנשאר ביחס למשקלו הנתון, נגדיר ערך זה להיות α, ומתקיים $\alpha \in [0,1]$
   לכן:

$$f\left(x^*\right) \leqslant f\left(z^*\right) = \sum_{i=1}^{k-1} v_i + \alpha v_k \overset{\star}{=} f(x_1) + \alpha f(x_2) \overset{☆}{\leqslant} f(x_1) + f(x_2) \leqslant 2 \cdot max\left\{f(x_1), f(x_2)\right\}$$
$$= 2f(x) \blacksquare$$

8. ✶- לפי הגדרת הוקטורים בסעיף 5

☆ - $a \leqslant 1$

9. זמן ריצה: היות ואנו ממיינים תחילה את כלל האיברים בעלות של $O(nlogn)$, כמו כן האלגוריתם מבצע לכל היותר מעבר 1 נוסף עבור כל איבר, כך שכל מעבר אורך $O(1)$, מעבר על כלל האיברים יהיה $O(n)$ לכן זמן הריצה הכללי של האלגוריתם יהיה $O(nlogn)$.

שאלה 5:

1. אלגוריתם:

נמיין את הפריטים מהקטן לגדול ונגדיר את b כסדרה a לאחר המיון, לכן $b_1 \leqslant .. \leqslant b_n$.
ניקח את הפריט הגדול ביותר שעדיין לא הוכנס, נכניס אותו (באיטרציה הראשונה הפריט יהיה בעל המשקל המקסימלי).
לאחר מכן, נמשיך לעבור על כל הפריטים מהפריט הבא עד הקטן ביותר ונבדוק: אם התוספת של המשקל של הפריט הנוכחי עדיין קטן מ-1 ק"ג, נוסיף אותו (ונסיר את הפריט מאופציות ההכנסה).
נמשיך לעבור על הפריטים עד ש: משקל הקופסא יהיה שווה ל-1, או הגענו לפריט הקטן ביותר (או שנגמרו הפריטים). בכל אחד מהמצבים שבהם הפסקנו, נפתח קופסא חדשה ונתחיל לעבור שוב על האיברים מהאיבר הגדול ביותר שנותר, עד האיבר הקטן ביותר שנותר עד אשר יסתיימו כל הפריטים.

2. חוקיות:

עבור כל קופסא, האלגוריתם לא מכניס יותר מ-1 ק"ג. כמו כן, האלגוריתם ימשיך כל עוד יש פריטים שלא הוכנסו לקופסאות.

3. אופטימליות:

עבור פתרון אופטימלי, יש לכל הפחות $\sum_{i=1}^{n} x_i$ קופסאות, לכן אם נראה שעבור כל קופסא אנו ממלאים לפחות $\frac{1}{2}$ ק"ג, נראה שהאלגוריתם הוא "2 קירוב" לפתרון האופטימלי.
עבור כל פריט:

(a) אם משקל הפריט גדול מ $\frac{1}{2}$ והקופסא ריקה, לאחר ההכנסה הקופסא תהיה לפחות במשקל $\frac{1}{2}$ ק"ג, בפועל ייתכן שיכנסו בה איברים נוספים.

(b) אם משקל הפריט גדול מ $\frac{1}{2}$ והקופסא אינה ריקה, כלומר יש איבר גדול ממנו לפניו, לכן נפתח קופסא חדשה ונחזור ל(a). (ב-2 הקופסאות אנו מנצלים לפחות $\frac{1}{2}$ מהקיבולת של האריזה).

(c) אם משקל הפריט קטן או שווה ל- $\frac{1}{2}$ והקופסא ריקה, נכניס את הפריט, ואף יהיה מספיק מקום גם לאיבר הבא הקטן (היות והם ממויינים בסדר יורד), לכן הקופסא תהיה גדולה מ $\frac{1}{2}$ ק"ג.

(d) אם משקל הפריט קטן או שווה ל- $\frac{1}{2}$ והקופסא אינה ריקה, אם יש יכולת להכניס את הפריט לקופסא, נכניס וכך נמצא לפחות $\frac{1}{2}$ ק"ג של הקופסא, אחרת נפתח קופסא חדשה.

(e) ייתכן שיהיה קופסא אחרונה יחידה, כך שיכולת הקיבול לא נוצל $\frac{1}{2}$ ק"ג, במצב זה: לפי האלגוריתם אם קיימת קופסא שניתן להכניס את הפריטים לקופסאות קודמות ניתן לחסוך את הקופסא האחרונה, אך אם מצב זה לא אפשרי, מראש הסיבה שפתחנו את הקופסא הזו הייתה כי לא יכולנו להכניס את האיברים לקופסאות אחרות, כלומר סכום כל קופסא אחרת והקופסא

הנוכחית הוא לפחות 1, לכן בהכרח נצטרך 2 קופסאות ולכן הפתרון הוא 2 קירוב לפתרון האופטימלי.