פתרון עבודה 3 – אלגוריתמים 172

תרגיל 1

סעיף א'
עבור סדרת העלויות $\{b_1, \dots, b_n\}$ נגדיר את $OPT(i)$ עבור $1 \le i \le n$, להיות הערך המינימלי של סכום הבזבוזים עבור תת-הבעיה של סדרת העלויות $\{b_1, \dots, b_i\}$.
מיקום הפתרון הינו $OPT(n)$.

סעיף ב'

$$OPT(i) = \begin{cases} b_i + min_{1\le k\le 6}\{OPT(i-k)\}, & i > 0 \\ 0, & i = 0 \\ \infty, & i < 0 \end{cases}$$

סעיף ג'
ניתוח מקרי הבסיס:
עבור $i = 0$, נקבל שאכן $OPT(i) = 0$ מהגדרת הבעיה. באופן דומה, נקבל כי אכן $OPT(i) = \infty$ עבור $i < 0$.

הוכחת יתר נוסחת המבנה:
נחלק את הפתרונות האפשריים לקבוצות: נגדיר 6 תתי קבוצות של פתרונות: $S_1, \dots, S_6$ כך ש-S_k מכילה את כל הפתרונות לבעיה בהן ההטלה הראשונה בקובייה הניבה את התוצאה k.

תת- הקבוצות הנ"ל מכסות את קבוצת כל הפתרונות האפשריים (כיסוי):
יהי פתרון u לתת הבעיה $\{b_1, \dots, b_i\}$. לפי ההגדרה, פתרון חוקי הינו סדרת הטלת קוביה. נתבונן באיבר הראשון בפתרון, $1 \le k \le 6$. כלומר הטלת הקובייה הראשונה בסדרה הניבה את התוצאה k. לכן לפי הגדרת הקבוצה S_k מתקיים: $u \epsilon S_k$.

מסקנה לצורת נוסחת המבנה:
נגדיר את $O^*(S_k)$ להיות עלות הפתרון המינימלי בS_k. כלומר:

$$O^*(S_k) = min_{u\epsilon S_k}\{price(u)\}$$

כאשר $price$ מחושב בהתאם להגדרת הבעיה: $price(u) = \sum_{i\epsilon u} b(i)$.
אזי מניתוח המקרים והכיסוי נוכל להסיק שצורת נוסחת המבנה הינה: $OPT(i) = min_{1\le k\le 6}\{O^*(S_k)\}$.

ניתוח האופטימום של כל קבוצה:
נוכיח כי לכל $1 \le k \le 6$ מתקיים: $O^*(S_k) = b_i + OPT(i-k)$.

כיוון 1: $\boldsymbol{O^*(S_k) \le b_i + OPT(i-k)}$.
יהי u_1 פתרון חוקי לתת הבעיה $\{b_1, \dots, b_{i-k}\}$ כך שמתקיים $OPT(i-k) = price(u_1)$. נגדיר פתרון חדש u בו נקודת ההתחלה הינה החנות ה-i כך שההטלה הראשונה היא k ושאר סדרת ההטלות מורכבת מסדרת ההטלות בu_1. לפי ההגדרת מחיר הפתרון מתקיים: $price(u) = b_i + price(u_1) = b_i + \sum_{j\epsilon u_1} b(j)$
כלומר: $price(u) = b_i + OPT(i-k)$.
k הינו ערך בין 1 ל-6, ו-u_1 הינו פתרון חוקי עבור תת הבעיה $\{b_1, \dots, b_{i-k}\}$. ולכן u הינו פתרון חוקי לתת הבעיה $\{b_1, \dots, b_i\}$ אשר ההטלה הראשונה בו הינה k. מהגדרת S_k נסיק כי $u \in S_k$ ולכן מהגדרת $O^*(S_k)$ נסכם ש-$O^*(S_k) \le price(u) = b_i + OPT(i-k)$.

כיוון 2: $\boldsymbol{O^*(S_k) \ge b_i + OPT(i-k)}$.
יהי u פתרון ב-S_k, כך שמתקיים $O^*(S_k) = price(u)$. לפי הגדרת S_k, ההטלה הראשונה ב-u הינה k ומסע הקניות מתחיל בחנות ה-i. נגדיר פתרון u_1 המורכב מכל ההטלות ב-u מלבד ההטלה הראשונה. הפתרון u_1 הינו פתרון חוקי עבור תת הבעיה $\{b_1, \dots, b_{i-k}\}$ (עם החנות ה-$(i-k)$ כנקודת התחלה) ולכן:
$price(u_1) \ge OPT(i-k)$.
נשים לב כי מחיר הפתרון u זהה למחיר הפתרון u_1 בתוספת מחיר החנות ה-i. כלומר:

$O^*(S_k) = price(u) = b_i + price \geq\ b_i + OPT(i-k)$ ולכן $O^*(S_k) = b_i + OPT[i-k]$.

משני הכיוונים נקבל ש-$\boldsymbol{O^*(S_k) = b_i + OPT(i-k)}$, ובשילוב עם המסקנה לצורתה הכללית של נוסחת המבנה - מתקבלת נכונותה.

סעיף ד'

נעשה שימוש במערך n-מימדי A.

1. $A[0]\ =\ 0$
2. $for\ i = 1\ to\ n$:
   3.1. $A[i]\ =\ \infty$
   3.2. $for\ k = max(0, i-6)\ to\ (i-1)$:
      3.2.1. $if\ A[i]\ <\ \min\{A[i], b_i\ +\ A[i-k]\}$:
         3.2.1.1. $A[i]\ =\ \min\{A[i], b_i\ +\ A[i-k]\}$
3. $return\ A[n]$

זמן ריצת האלגוריתם הינו $O(n)$ שכן מבוצע מעבר על כל איברי המערך (לצורך חישוב ערכן של תתי-הבעיות), כך שחישובו של כל תא אורך מספר קבוע של צעדי חישוב.

<u>הוכחת נכונות:</u>

<u>טענה 1 (סדר חישוב הבעיות):</u> בכל שלב של האלגוריתם, הערכים הדרושים לצורך חישוב $A[i]$ חושבו בעבר.

<u>הוכחה:</u> באינדוקציה שלמה על מספר השלב.

בסיס: בשלב האתחול מבוצע חישוב "ישיר", ללא צורך בערכים קודמים ולכן הטענה נכונה באופן ריק.

הנחה: נניח כי עבור כל שלב $j < i$, בעת השלב ה-j חושבו כבר כל הערכים הרלוונטים לצורך חישוב התא $A[j]$ (כלומר $A[j-1], A[j-2], \ldots, A[j-\max\{0, j-6\}]$).

נוכיח כי בשלב ה-$\boldsymbol{i}$ חושבו בעבר כל הערכים הרלוונטים לצורך חישוב $\boldsymbol{A[i]}$ עבור $\boldsymbol{1 \leq i \leq n}$:

מאופן פעולת האלגוריתם, ישנם לכל היותר 6 תאים הדרושים לצורך חישוב $A[i]$ והם $A[i-1], \ldots, A[i-\max\{0, i-6\}]$. נבחין כי כל אחד מהם הינו תא בעל אינדקס הקטן מ-i ולכן מהנחת האינדוקציה נקבל כי הוא חושב בעבר ומכאן נובעת נכונות צעד האינדוקציה.

<u>טענה 2 (האלגוריתם הינו מימוש של נוסחת המבנה):</u> לכל $0 \leq i \leq n$, $M[i] = OPT(i)$.

<u>הוכחה:</u> באינדוקציה על צעדי האלגוריתם.

<u>אבחנה:</u> לאחר הצעד ה-i של האלגוריתם, הערך $A[i]$ לא משתנה עד סיום הריצה.

בסיס: ההשמה הראשונה באלגוריתם הינה $A[0] = 0$, כמתבקש מהגדרת נוסחת המבנה.

הנחה: נניח כי לכל $0 \leq\ j < i \leq n$, $M[j] = OPT[j]$.

נוכיח כי $\boldsymbol{A[i] = OPT(i)}$, לכל $\boldsymbol{1 \leq i \leq n}$: מהנחת האינדוקציה, לכל $k \in \{i-1, i-2, \ldots, \max\{0, i-6\}\}$, מתקיים ש-$A[k] = OPT(k)$. לפי שורות 5, 6 ו-7 באלגוריתם אנו מעדכנים את $A[i]$ רק כאשר $A[i]\ <\ min(A[i], b_i\ +\ A[i-k]$.

כמו-כן, אנו בודקים רק את $\max\{0, i-6\} \leq k \leq 6$.

לכן: $A[i]\ =\ min_{\max\{0,i-6\} \leq k \leq 6}\{b_i + A[i-k]\}$. נשים לב כי כאשר $i-k < 0$, לפי הגדרה $OPT(i-k) = \infty$ ולכן ברור כי אין טעם בבחינת מקרים בהם מגיעים לאינדקס שלילי.

בנוסף לכך, מנהת האינדוקציה נקבל כי $A[j] = OPT(j)$ לכל $j < i$, ולכן בשילוב עם נוסחת המבנה נקבל כי $A[i] = OPT(i)$.

טענות 1 ו-2 יחדיו מוכיחות שבתום ריצת האלגוריתם, בתא $A[n]$ ימצא האופטימום עבור הקלט.

תרגיל 2

סעיף א'

נסמן את אלף-בית הקלט $\Sigma = \{\sigma_1, \sigma_2, \dots, \sigma_n\}$ ואת רשימת התדירויות $F = \{f_1, \dots, f_n\}$, כך ש- f_i ($1 \le i \le n$) הינו מספר המופעים עבור התו σ_i. נגדיר את $OPT(i,j)$ כמספר ההקשות עבור סידור אופטימלי של התוים $\{\sigma_1, \dots, \sigma_i\}$ (עם תדירויות $\{f_1, \dots, f_i\}$) על-פני j מקשים (כאשר $1 \le j \le 12$).

סעיף ב'

$$OPT(i,j) = \begin{cases} 0 & ,i = 0 \\ \Sigma_{1 \le k \le i} f_k * k & ,j = 1 \\ \min\limits_{0 \le k \le i} \{OPT(i-k, j-1) + \Sigma_{i-k+1 \le w \le i} f_w * (w - i + k) & ,otherwise \end{cases}$$

את הפתרון נקבל ע"י הפעלת $OPT(n, 12)$.

סעיף ג'

<u>הערה:</u> בהינתן פתרון חוקי O, נסמן ב-$cost(O)$ את ערכו (=מספר ההקשות שיבוצעו) של O.

<u>הגדרה:</u> בהינתן $\Sigma = \{\sigma_1, \sigma_2, \dots, \sigma_i, \dots \sigma_j, \dots, \sigma_n\}$ נגדיר את הפתרון O' כ***הרחבה*** של O אם:
פתרון O' המשבץ באופן חוקי i תווים ל-k מקשים, ושיבוץ של $\{\sigma_{i+1}, \dots, \sigma_j\}$ למקש בודד, נגדיר את הפתרון O כפתרון המשבץ את $\{\sigma_1, \dots, \sigma_i\}$ ל-k מקשים באופן זהה ל-O', ואת התווים $\{\sigma_{i+1}, \dots, \sigma_j\}$ למקש ה-$k + 1$.

<u>נוכיח את מקרי הבסיס:</u>

$OPT(0, j) = 0$:

מכיוון שהאלף-בית הנידון הינו בגודל 0, לא ניתן ליצור אף מחרוזת ולכן מספר ההקשות הצפוי הינו 0.

$OPT(i, 1) = \Sigma_{1 \le k \le j} f_k * k$:

תהא Sol קבוצת כל הפתרונות החוקיים לשיבוץ התווים $\{\sigma_1, \dots, \sigma_i\}$ במקש בודד (שכן $j = 1$), ו-$O \in Sol$.
נבחין כי $|Sol| = 1$, כלומר קיים פתרון יחיד, ולכן $cost(O) = OPT(i.1)$. כמו-כן, קל לראות כי $cost(O) = \Sigma_{1 \le k \le j} f_k * k$, שכל התו σ_i יהיה התו ה-i במקש היחיד, ולכן מספר הלחיצות הנדרשות עבורו הינו k. בסה"כ $OPT(i, 1) = cost(O) = \Sigma_{1 \le k \le j} f_k * k$.

<u>חלוקה למקרים:</u> נגדיר את Sol_k (עבור $0 \le k \le i$) כקבוצת כל הפתרונות החוקיים עבור בעיית שיבוץ התווים $\{\sigma_1, \dots, \sigma_i\}$ ל-j מקשים, כך שהמקש ה-j מכיל את התווים $\{\sigma_{i-k+1}, \dots, \sigma_i\}$ (כלומר, המקש ה-j מכיל את k התווים האחרונים מתוך $\{\sigma_1, \dots, \sigma_i\}$).

<u>המקרים הנ"ל מכסים את כל הפתרונות החוקיים:</u> יהא O פתרון חוקי כלשהו. תהא O'' קבוצת התווים המשובצים למקש ה-j במסגרת הפתרון O. מאחר ו-O חוקי, נסיק כי $0 \le |O''| \le i$. נסמן ב-m את התו בעל האינדקס המינימלי בנמצא ב-O'', וב-M את התו המקביל בעל האינדקס המקסימלי. ברור כי מחוקיות הפתרון ניתן להציג את O'' כקבוצת כל התוים אשר נמצאים בין m ל-M:

$$O'' = \{\sigma \in \Sigma |\ m \le \sigma \le M\}$$

נסמן $|O''| = k$, אזי $O \in Sol_k$ (מכיוון שהינו פתרון בו התווים $\{\sigma_{i-k+1}, \dots, \sigma_i\}$ משובצים במקש ה-j).

<u>מסקנה למבנה הנוסחה:</u> נסמן $O^*(Sol_k) = \min_{O \in Sol_k} \{cost(O)\}$. מן האמור לעיל נסיק כי $OPT(i,j) = \min_{0 \le k \le i} \{O^*(Sol_k)\}$.

<u>ניתוח האופטימום של כל קבוצה:</u> נוכיח $O^*(Sol_k) = OPT(i-k, j-1) + \Sigma_{i-k+1 \le w \le i} f_w * (w - i + k)$.
<u>כיוון $\le$:</u> יהא $O \in Sol_k$ פתרון המקיים $cost(O) = O^*(Sol_k)$. מהגדרת Sol_k נסיק כי התווים $\{\sigma_{i-k+1}, \dots, \sigma_i\}$ משובצים למקש בודד והוא המקש ה-j. נסמן ב-O'' את מיפוי התווים בעוד יתר המקשים (כלומר $\{\sigma_1, \dots, \sigma_{i-k}\}$) ממופים ל-$j - 1$ המקשים הנותרים. נסמן את מיפוי יתר התווים המתקבל מ-O ע"י O'. ברור כי $cost(O) = cost(O') + cost(O'')$.
מהגדרת O' ומהגדרת OPT, נסיק כי $cost(O') \le OPT(i-k, j-1)$, בעוד $cost(O'') = \Sigma_{i-k+1 \le w \le i} f_w * (w - i + k)$ (וזאת מהוכחת מקרי הבסיס), ולכן בסה"כ:
$O^*(Sol_k) = cost(O) \le OPT(i-k, j-1) + \Sigma_{i-k+1 \le w \le i} f_w * (w - i + k)$.

כיוון $\geq$: יהא O' פתרון המשבץ את $i-k$ התווים הראשונים של Σ ל-$j-1$ מקשים המקיים $cost(O') = OPT(i-k, j-1)$. נסמן ב-O את הפתרון המרחיב את O' ע"י שיבוץ התווים $\{\sigma_{i_{i-k+1}}, \ldots, \sigma_i\}$ למקש אחד, הוא המקש ה-j (עלות ההרחבה נותחה באחד ממקרי הבסיס). אזי,

$$cost(O) = cost(O') + \Sigma_{i-k+1 \leq w \leq i} f_w * (w-i+k)$$
$$= OPT(i-k, j-1) + \Sigma_{i-k+1 \leq w \leq i} f_w * (w-i+k)$$

נבחין כי הפתרון O משבץ את i התווים של Σ ל-j מקשים שונים באופן חוקי (כלומר חלוקת כל תווי Σ ל-j מקטעים השומרים על יחס הסדר אשר מוגדר ע"י Σ), ולכן $O \in Sol_k$. מהגדרת $O^*(\ldots)$ נקבל ש-$cost(O) = OPT(i-k, j-1) + \Sigma_{i-k+1 \leq w \leq i} f_w * (w-i+k) \leq O^*(Sol_k)$, כנדרש.
משילוב שני הכיוונים, נקבל שיוויון בין שני האגפים, ולכן בסה"כ הוכחנו את נכונות נוסחת המבנה.

סעיף ד'

$\underline{Iterative - Pinokia - Algorithm\ (\Sigma, F):}$

1. $n = |\Sigma|$
2. $M: nX12\ matrix$
3. $for\ 1 \leq i \leq n$:
    3.1. $for\ 2 \leq j \leq 12$:
        3.1.1. $M[i,j] = +\infty$
4. $for\ 1 \leq j \leq 12$:
    4.1. $M[0,j] = 0$
5. $for\ 1 \leq i \leq n$:
    5.1. $M[i,1] = M[i-1,1] + f_i * i$
6. $for\ 1 \leq i \leq n$:
    6.1. $for\ 2 \leq j \leq 12$:
        6.2. $M[i,j] = \min_{1 \leq k \leq i}\{M(i-k, j-1) + \Sigma_{i-k+1 \leq w \leq i} f_w * (w - i + k)\}$
7. $return\ M[n,12]$

<u>ניתוח זמן ריצה:</u> שלב האיתחול (שורות 1 – 4), יארך כ-$O(12n) = O(n)$ צעדי חישוב. שורה 5 תארך אף היא $O(n)$ במימוש ה"עצל", ו-$O(n^2)$ במימוש נאיבי של החלק המתאים עבורה בנוסחת המבנה (ראו הערה א'). החלר המשמעותי ביותר הינו שורה 6 (ותתי השורות הנגזרות ממנה), שכן ישנן n איטרציות, כך שבכל איטרציה מבוצע חישוב מינימום על $O(n)$ איברים, כך שעלות החישוב של כל אחד מן האיברים הללו הינה $O(n)$ במקרה הגרוע, שכן אנו סוכמים את עלות שיבוצם של $i-k$ התווים האחרונים במקש האחרון. בסה"כ קיבלנו שעלות השלב ה-6 של האלגוריתם הינה $O(n^3)$.

<u>הערות:</u>
<u>א.</u> הדרך הנאיבית למימוש שורה 5.1 הינה שימוש ישיר בנוסחת המבנה. אלא שזמן עלות החישוב הישיר הינו $O(n^2)$, שכן הלולאה בסעיף 4 מבצעת n איטרציות, אשר כל אחת מהן מבצעת i פעולות חיבור, כאשר i הינו מספר האיטרציה. לכן סיבוכיות הפעולות בלולאה אשר בסעיף 4 הינו $\sum_{i=1}^{n} O(i) = O(n^2)$. החישוב ה"עצל" המופיע בסעיף 4.1 מפחית את זמן ריצת הלולאה ל-$O(n)$.
<u>ב.</u> ניתן לשפר את זמן ריצת האלגוריתם מ-$O(n^3)$ ל-$O(n^2)$ ע"י הפחתת חישוב העלות של שיבוץ $i-k$ התווים האחרונים למקש האחרון, וזאת ע"י חישוב "עצל" בו האיבר הבא מחושב ב-$O(1)$ בעזרת החישוב אשר בוצע עבור האיבר שלפניו. חישוב זה יבוצע באופן הבא:

6. $for\ 1 \le i \le n$:

6.1. $for\ 2 \le j \le 12$:

6.1.1. $sum = 0$

6.1.2. $cost = 0$

6.1.3. $for\ w = 1\ to\ i$:

6.1.3.1. $sum = sum + f_w$

6.1.4.1. $cost = cost + w * f_w$

6.1.4. $for\ w = 1\ to\ i$:

6.1.4.1. $M[i,j] = \min\{M[i,j], M(w-1, j-1) + cost\}$

6.1.4.2. $cost = cost - sum$

6.1.4.3. $sum = sum - f_{w-1}$

שימו לב שאכן זמן הריצה של סעיף 6 הינו $O(12n^2) = O(n^2)$. כמו-כן, עלות שיבוצם של $w-1$ התווים האחרונים באלף-בית במקש האחרון מחושב ב-$O(1)$ באמצעות החישוב המקביל עבור w התווים האחרונים.

תרגיל 3

סעיף א'

יהא $O = G_1, \ldots, G_k$ פתרון אופטימלי אשר בו אוסף העוגיות G_i ממוקם בנקודה p_i עבור $1 \le i \le k$. נניח בשלילה כי קיים p_i כך שמתקיים: $p_i \neq \min\{x(c) : c \in G_i\}$.

נחלק למקרים:

1. $p_i > \min\{x(c) : c \in G_i\}$, כלומר קיימת עוגיה ב$G_i$ בעלת מיקום קטן יותר מ-p_i. לכן, בכדי להעביר אותה ל-p_i היה עלינו להזיזה "קדימה" במעלה שורת העוגיות, וזו בסתירה להגדרת השאלה.
2. $p_i < \min\{x(c) : c \in G_i\}$. נגדיר ערימה חדשה G'_i במקום $\min\{x(c) : c \in G_i\}$. כלומר: $p'_i = \min\{x(c) : c \in G_i\}$. נבחין כי הערימה $G_i{}'$ מכילה את אותן עוגיות המכילה ערימה G_i, אלא שמיקומה שונה. מכיוון ש-$p_i < p_i{}'$:

$cost(G_i) = \sum_{c \in G_i} \mathrm{w(c)} * (\mathrm{x(c)} - p_i) > \sum_{c \in G'_i} \mathrm{w(c)} * (\mathrm{x(c)} - p'_i) = cost(G'_i)$.

לכן, הפתרון $O' = O \setminus \{G_i\} \cup \{G'_i\}$ הינו חלוקה של כל n העוגיות ל-k ערימות ולכן הינו פתרון חוקי המקיים $cost(O) > cost(O')$ וזו בסתירה לאופטימליות של O.

סעיף ב'

מכיוון ש-$c_{m_i}, c_{M_i} \in G_i$, עלינו להוכיח שלכל $m_i < j < M_i$, העוגיה c_j נמצאת אף היא ב-G_i (כלומר ניתן לוותר על השיוויון בטענה). נניח בשלילה כי קיים פתרון אופטימלי O ובו ערימה G_i ו- $m_i \le j \le M_i$ ומתקיים כי c_j לא שייך לקבוצה G_i. מאחר ו-O פתרון חוקי, קיימת קבוצה $G_k \epsilon O$ כך ש$c_j \in G_k$. נחלק למקרים:

$\boldsymbol{p_k < p_i}$: נבנה 2 ערימות חדשות: $G'_k = G_k / \{c_j\}$ ו- $G'_i = G_i \cup \{c_j\}$. עוד נבחין כי:

$cost(G'_k) = cost(G_k) - (x(j) - p_k)$ וגם $cost(G'_i) = cost(G_i) + (x(j) - p_i)$

מאחר ו- $p_k < p_i$, $cost(G'_i) + cost(G'_k) < cost(G_i) + cost(G_k)$. ולכן הפתרון $O' = O / \{G_i, G_k\} \cup \{G'_i, G'_k\}$ הינו פתרון חוקי ובעל עלות נמוכה יותר, וזו בסתירה לאופטימליות של O.

$\boldsymbol{p_k > p_i}$: נשים לב כי מאחר ומתקיים $m_i \le j \le M_i$ ולפי סעיף א' ברור ש- $c_j \ge p_k$ אזי $p_k \le M_i$. נבנה 2 ערימות חדשות $G'_i = G_i \setminus \{c_{M_i}\}$ ו- $G'_k = G_k \cup \{c_{M_i}\}$. משקלי הקבוצות החדשות הן: $cost(G'_k) = cost(G_k) + (x(c_{M_i}) - p_k)$וגם

$cost(G'_i) = cost(G_i) - (x(c_{M_i}) - p_i)$.

מאחר ו- $p_k > p_i$,

$cost(G'_i) + cost(G'_k) < cost(G_i) + cost(G_k)$

ולכן הפתרון $O' = O \setminus \{G_i, G_k\} \cup \{G'_i, G'_k\}$ הינו פתרון חוקי ובעל עלות נמוכה יותר, בסתירה לאופטימליות O.

סעיף ג'

עבור סדרת העוגיות $\{c_1, \dots, c_n\}$ ו-k ערימות, נגדיר את $OPT(i,j)$ (עבור $0 \le j \le k$ ו-$0 \le i \le n$) להיות העלות המינימלית של סידור i העוגיות הראשונות ב-j ערימות. הפתרון עבור הבעיה ימצא ב-$OPT(n,k)$.

סעיף ד'

i – מספר העוגיות בשביל,
j – מספר הערימות בהן עלינו לסדר את i העוגיות.

$$OPT(i,j) = \begin{cases} 0, & i = 0 \\ 0, & j = 0 \\ min_{1 \le t < i}\left(OPT(t-1, j-1) + \sum_{t<k\le i} w(k) * \big(x(k) - x(t)\big)\right), & i > 0, j > 1 \\ \sum_{1<k\le i} w(k) * (x(k) - x(1)), & i > 0, j = 1 \end{cases}$$

$$= \begin{cases} 0, & i = 0 \\ 0, & j = 0 \\ min_{1 \le t < i}\left(OPT(t-1, j-1) + \sum_{t<k\le i} w(k) * (k - t)\right), & i > 0, j > 1 \\ \sum_{1<k\le i} w(k) * (k-1), & i > 0, j = 1 \end{cases}$$

<u>הסבר לנכונות הנוסחה:</u>
מסעיף ב' נסיק כי הפתרון האופטימלי הינו חלוקה של n העוגיות ל-k מקטעים. מסעיף א' נסיק כי כל מקטע (כלומר רצף עוגיות) ימופה לערימה הממוקמת בדיוק בנקודה בה נמצאת העוגיה בעלת ערך ה-x הנמוך ביותר בתוך כל מקטע. לכן, נקבל שהפתרון האופטימלי עבור סידור i עוגיות ב-j ערימות, הינו סידורן של "רישא" של $t-1$ עוגיות ב-$j-1$ ערימות, כאשר העוגיות $c_t, \dots, c_i$ נערמות בנקודה $x(c_t) = t$. מאחר ואיננו יודעים איזו סיפא של עוגיות תקובץ לערימה ה-j, אנו בוחרים במינימום שבין כל האפשרויות להבחירת סיפא.
כמו-כן, אם יש נותרנו עם ערימה אחת, עלינו לשבץ את כל העוגיות אשר נותרו לערימה זו, שכמובן תמוקם בנקודה $x = 1$ (וזאת מסעיף א').

סעיף ה'

<u>$Iterative - CookieMonster - Algorithm\ (\mathrm{n}, k)$:</u>

1. $k = \min\{k, n\}, M = n\ X\ k\ matrix, \pi = n\ X\ k\ matrix$
2. $for\ 1 \le j \le k$:
    2.1. $M[0, j] = 0$
3. $for\ 1 \le i \le n$:
    3.1. $M[i, 1] = M[i-1, 1] + w_i * (x(i) - x(1))$
4. $for\ 1 \le i \le n$:
    4.1. $for\ 2 \le j \le k$:
        4.1.1. $M[i, j] = +\infty$
        4.1.2. $sum = 0$
        4.1.3. $cost = 0$
        4.1.4. $for\ f = 2\ to\ i$:

4.1.4.1. $sum = sum + w_f$
4.1.4.2. $cost = cost + w_f * (x(f) - 1)$
4.1.5. $for\ f = 1\ to\ i$:
4.1.5.1. $if\ M[i,j] > M[f-1, j-1] + cost$:
4.1.5.1.1. $M[i,j] = M[f-1, j-1] + cost$
4.1.5.1.2. $\pi[i,j] = f$
4.1.5.2. $cost = cost - sum$
4.1.5.3. $sum = sum - w_{f+1}$
5. $return\ M[n,k]$

<u>הסבר:</u>
נכונות האלגוריתם נובעת מנכונות נוסחת המבנה, מאחר ובכל שלב אנו למעשה מבצעים:

$$M[i,j] = min_{1 \le w < i}\left(M[w-1, j-1] + \sum_{w<k\le i} w(k) * \big(x(k) - x(w)\big)\right)$$

בנוסף לכך, חישוב מחיר הערימה מתבצע ב-$O(1)$ פעולות בעזרת מחיר הערימה הקודמת.

<u>זמן ריצה:</u>
ראשית, ישנו אתחול המתבצע ב-$O(k+n)$. לאחר מכן ישנה לולאה מקוננת, ובה $k * n$ איטרציות.
בכל אחד מ-$k * n$ השלבים אנו מבצעים לולאה נוספת המבצעת $O(n)$ צעדים לכל היותר. נשים לב כי בכל צעד של לולאה זו, אנו מבצעים $O(1)$ פעולות ובכך מחשבים עלויות באופן "עצל". בסה"כ נקבל שהאלגוריתם רץ זמן $O(n^2 k)$. מאחר ואין טעם להשתמש בערימות ריקות, מצב בו $k > n$ יניב תוצאות זהות לשימוש ב-n ערימות בלבד (שכן ערימה ריקה אינה משפרת את עלויות הפתרון). לכן, בשורה הראשונה אנו מקטינים את k ל-n (במידת הצורך). בסה"כ נקבל שלאחר השורה הראשונה, $k \le n$ ולכן $O(n^2 k) = O(n^3)$ כמתבקש.
ראו ההערות אשר בסעיף ד' של שאלה 2 בנוגע לחישוב "עצל".

<u>סעיף ו'</u>

בהינתן המטריצה π, נבצע את הצעדים הבאים לצורך שחזורו של פתרון אופטימלי:

<u>$Reconstruction(\pi, n, k)$:</u>
1. $S = \emptyset, i = n, j = k.$
2. $while\ i > 0$:
2.1. $S = S \cup \pi[i,j]$
2.2. $i = \pi[i,j]$
2.3. $j = j - 1$
3. $return\ S$

הקבוצה S מכילה את מיקומי הערימות, ומהן ניתן (בהתבסס על סעיפים א' ו-ב') להסיק באופן מידי מיהן העוגיות המשויכות לכל ערימה.