2 – אלגוריתמים חמדניים

אלגוריתם חמדן פותר בעיית אופטימיזציה. בכל שלב לוקחים את הפתרון ה"מקומי" הכי טוב. בכל שלב הבעיה נהיית קטנה יותר. לפעמים נוכיח את נכונות האלגוריתם ע"י אינדוקציה. כדי להוכיח נכונות של אלגוריתם חמדן, צריך להוכיח:

1) **תכונת הבחירה החמדנית:** הבחירה הטובה המקומית היא גם הבחירה הטובה הגלובלית. אפשר להוכיח את זה ע"י כך שנראה שהבחירה הראשונה בפתרון אופטימלי כלשהו, היא אותה בחירה באלגוריתם החמדן.
2) **תכונת תת-מבנה אופטימלי:** הפתרון האופטימלי עבור כל הבעיה הוא גם פתרון אופטימלי לכל תת-בעיה. לדוגמה אם יש בעיה שדורשת סידור כלשהו של מערך: ניקח את הסדר שבו נסדר בשביל הפתרון האופטימלי הכללי. אם ניקח את המערך בלי האיבר האחרון (או כל תת קבוצה של המערך הזה) ונשתמש באותו סדר, זה יהיה פתרון אופטימלי.

שינוי בלולאה – loop invariant:
עוד דרך להוכיח נכונות היא ע"י אינווריאנטה של הלולאה: נניח שיש תכונה P כלשהי שמוכיחה או מגדירה את נכונות הפתרון. נראה שאם P מתקיימת לפני איטרציה של הלולאה, היא תתקיים גם אחרי אותה איטרציה. וכך באינדוקציה מוכיחים נכונות.

אלגוריתם חמדן לא תמיד יעבוד – לדוגמה בעיית העודף, בעיית תרמיל הגב בשלמים.

<u>בעיית תרמיל הגב בשברים:</u> בהינתן n פריטים, לכל אחת משקל וערך כולל. צריך לבחור שבר (בין 0 ל1) של הפריט שאותו ניקח בתרמיל שיש בו הגבלת משקל: $\sum_{i=1}^{n} f_i v_i,\ \text{such that: } 0 \le f_i \le 1,\ \sum_{i=1}^{n} w_i f_i \le C$
השיטה: נמיין את הפריטים לפי v_i/w_i , בסדר יורד. תמיד ניקח כמה שיותר מהפריט הבא, עד שהתיק מתמלא:

```
While(C>0)
    if(Wi < C) take Gi, C ← C – Wi   (take all of Gi)
    else C ← 0   (take C weight of Gi)
```

<u>בחירת פעילויות:</u> בהינתן n פעילויות, כל אחד עם זמן התחלה וסיום. נרצה לבחור כמה שיותר פעילויות כך שאין 2 פעילויות באותו זמן. נמיין את הפעילויות לפי זמן סיום. בכל שלב ניקח את הפעילות עם הזמן סיום הכי מוקדם שאין לה זמן התחלה חופף לפעילויות שבחרנו. הוכחת נכונות – לפי אינווריאנטה של הלולאה:
נגדיר תכונות: A- יש לנו מספר מקסימלי של פעילויות. B- יש לנו זמן סיום מינימלי. באינדוקציה על i:
בסיס: לפעילות הראשונה שנבחר יש זמן סיום מינימלי. וזה מספר הפעילויות כי גדול שאפשר אחרי בחירה אחת.
צעד: נניח שA,B מתקיימים לפני האיטרציה ה-i. אחרי האיטרציה: נב"ש שA לא מתקיים. כלומר יש לנו פחות אירועים ממה שיכול להיות לנו בשלב הזה. אבל לפי האלגוריתם, היינו לוקחים את המאורע הבא אם הזמנים לא חופפים. כלומר, הסיבה שכרגע אין לנו מספר מקסימלי של אירועים זה כי יכולנו לסיים מוקדם יותר ולקחת את המאורע הבא. אבל זה סתירה ל B.
ובגלל שבכל שלב לוקחים את הפעילות עם זמן סיום הכי מוקדם, B מתקיים אחרי כל שלב. מש"ל.

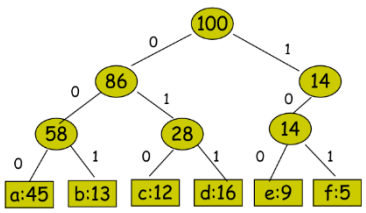
<u>בעיית תחנת הדלק:</u> בהינתן n רכבים בתור לתחנת דלק, לכל רכב יש זמן t_i – הזמן שדרוש לו כדי לתדלק. נרצה לסדר את הרכבים כך שסך הזמן שאנשים מחכים יהיה מינימלי. נסדר לפי t_i, בסדר עולה. הוכחת נכונות:

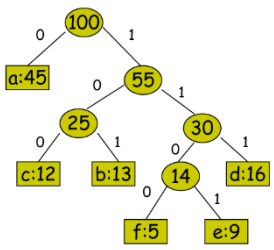
1) תכונת הבחירה החמדנית: נב"ש שיש פתרון אופטימלי אחר שבו הזמן המינימלי לא הראשון. אז נוכל להחליף בינו לבן מה שראשון, ולקבל זמן קצר יותר. סתירה.
2) תכונת תת-מבנה אופטימלי: יהי פתרון אופטימלי עבור $\{1 \cdots n\}$, נב"ש שהוא לא אופטימלי עבור $\{1 \cdots (n-1)\}$. אז ניקח את הפתרון שכן אופטימלי, נוסיף את n, ונקבל פתרון יותר טוב. סתירה.

<u>בעיית קוד בינארי:</u> רוצים לקודד מחרוזת של תווים לקוד בינארי, כך שלכל תו יש קוד בינארי ייחודי. הדרך הנאיבית היא לתת לכל תו קוד מאותו אורך. אבל אפשר לייעל את זה, אם ניתן לתווים היותר תדירים קוד קצר יותר, זה ייקצר את האורך של המחרוזת הבינארית. הבעיה היא שלא נדע איפה תו אחד מסתיים והשני מתחיל. צריך קוד **חסר-רישות**. אין קוד שהוא רישא של קוד אחר. אפשר לעשות את זה עם עץ בינארי: כל עלה הוא תו, פנייה שמאלה זה 0 ופנייה ימינה זה 1. המסלול לעלה זה הקוד של התו. איך נבנה עץ אופטימלי? (קוד שנותן את המחרוזת הבינארית הכי קצרה):

	a	b	c	d	e	f
Frequency	45	13	12	16	9	5
Fixed-length codeword	000	001	010	011	100	101

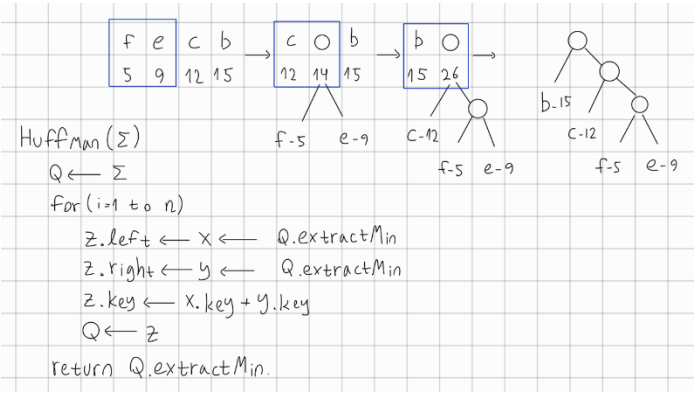
	a	b	c	d	e	f
Frequency	45	13	12	16	9	5
Variable-length codeword	0	101	100	111	1101	1100





יש את אלגוריתם **שאנון-פאנו**, ממיינים את התווים לפי תדירות ובכל שלב מחלקים באמצע. הוא נותן קוד יעיל אבל לא אופטימלי.

אלגוריתם **הופמן** נותן קוד אופטימלי: כל תו מתחיל בתור קודקוד, הערך שלו זה התדירות. (נשמור אותם בתור עדיפויות). בכל פעם ניקח את שני הקודקודים עם התדירות הכי נמוכה, נמזג אותם לשורש אחד והוא מקבל ערך שהוא סכום הערכים.



הוכחת נכונות: צ"ל: עבור משקלים $w_1 \cdots w_n$, הופמן נותן אורכים $l_1 \cdots l_n$ כך שהסכום $\sum_{i=1}^{n} w_i l_i$ יהיה מינימלי.

נצטרך 3 טענות עזר:

1) קוד אופטימלי יהיה עץ בינארי מלא – לכל קודקוד יש 0 או 2 ילדים.
הוכחה: אם יש קודקוד עם ילד אחד, הקודקוד הזה מיותר.
2) בקוד אופטימלי, 2 המשקלים הכי קטנים (w_{n-1}, w_n) נמצאים ברמה הכי נמוכה (נקרא לה L).
הוכחה: מכיוון שהעץ מלא, יש ברמה הכי נמוכה לפחות שני קודקודים. אז אם הם לא ברמה הכי נמוכה, נוכל להחליף אותם עם הקודקודים שכן, ולקבל קוד קצר יותר.
3) בעץ אופטימלי, (w_{n-1}, w_n) הם אחים.
הוכחה: לכל הקודקודים באותה רמה יש קוד באותו אורך. מכיוון ששניהם בL, נוכל פשוט להחליף מקומות.

הוכחת המשפט: באינדוקציה על n: בסיס n=1: האלגוריתם נותן קוד 0 לתו. זה אופטימלי.

נניח שהאלגוריתם אופטימלי עבור n-1. נתבונן בעץ שהופמן מייצר עבור n. בשלב הראשון, c_{n-1}, c_n מתמזגים לקודקוד אחד z, ועכשיו יש עץ עם n-1 קודקודים. נגדיר את T' העץ שהופמן מייצר מהקודקודים האלה. ע"פ הנ"א, הוא אופטימלי. עכשיו, נפצל את z חזרה ל c_{n-1}, c_n, נקרא לעץ הזה T. נגדיר $L(T)$ האורך של המחרוזת הבינארי לפי T. נשים לב שמתקיים: $L(T) = L(T') + w_{n-1} + w_n$. (כי כשמיזגנו אותם, הורדנו ביט אחד מכל מופע שלהם). כלומר $L(T') = L(T) - w_{n-1} - w_n$

נב"ש שקיים T^* כך ש: $L(T^*) < L(T)$. לפי טענות העזר, נוכל להניח שב T^* , c_{n-1}, c_n הם אחים ברמה הכי נמוכה. נשים במקומם את z ונקבל $L(T^*) - w_{n-1} - w_n < L(T')$, שזה פתרון טוב יותר עבור n-1. סתירה להנ"א.

<u>בעיית מקדמי הפולינום, בעיית המבצע:</u> שני מקרים שהאלגוריתם החמדן עובד, מסדרים את האיברים לפי סדר יורד. הוכחת הנכונות היא באינדוקציה, ומגיעה מכך שמניחים בשלילה שיש פתרון אופטימלי שבו האיבר הגדול לא ראשון, מחליפים אותו עם האיבר שבמקום הראשון ומקבלים פתרון יותר טוב.

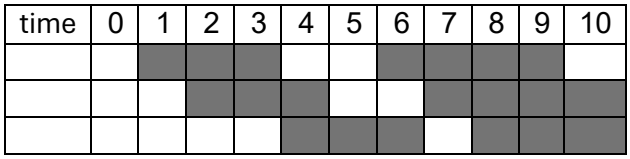
<u>בעיית הרציפים:</u> נתונים שני מערכים A- כניסות, D- יציאות של רכבות מתחנה. רוצים לדעת מה מספר הרציפים המינימלי הדרוש כדי שלא יהיו שתיים באותו רציף באותו זמן. נמיין את המערכים בסדר עולה. count שומר את מספר הרכבות בתחנה ברגע נתון, max שומר את הcount הכי גדול שראינו, המצביעים i, j רצים על המערכים. בכל שלב:

- אם $A[i] < D[j]$, i++, count++.
- אם $A[i] > D[j]$, j++, count--.
- אם $A[i] = D[j]$, i++, j++.

Index	0	1	2	3	4	5
A	1	2	4	6	7	8
D	3	4	6	9	10	10

דוגמה:

i	0	1	2	2	3	4	5	5	5
j	0	0	0	1	2	3	3	4	5
count	1	2	3	2	2	2	3	2	1
max	1	2	3	3	3	3	3	3	3



הוכחת נכונות באינדוקציה: בסיס עבור n=1, טריוויאלי.

עבור n=2 – אם זמן הכניסה השני קטן מהראשון, צריך 2 רציפים.

נניח שקיים פתרון אופטימלי G_n עבור n. נב"ש שהוא לא אופטימלי עבור n-1, כלומר קיים פתרון H_{n-1} אחר כך ש $\max(H_{n-1}) < \max(G_{n-1})$. ניקח את H ונוסיף את הרכבת ה-n. אם $A[n] \le D[n-1]$, נצטרך להוסיף 1 לשני הפתרונות. אם $A[n] > D[n-1]$, בשניהם לא נוסיף. בסה"כ קיבלנו ש $\max(H_n) < \max(G_n)$. סתירה להנחה.

<u>שוטרים וגנבים:</u> נתון מערך באורך n, בכל תא יש P- שוטר או T- גנב. כל שוטר יכול לתפוס עד גנב אחד, בטווח נתון k כלשהו. מה המספר המקסימלי של גנבים שאפשר לתפוס?

נעבור על המערך עם שני מצביעים – p,t. אם $|p - t| < k$, השוטר תופס את הגנב. ++count ונקדם את שני המצביעים ב1. אחרת, נקדם את הקטן מביניהם.

```
P_and_T(A, k){
	p ← first index of P in A
	t ← first index of T in A
	c ← 0
	while(p<=n && t<=n){
		if(|p-t|<k){
			c++
			p←P.next
			t ← T.next
		}
		else {
			if(p<t) p←P.next
			else t←T.next
		}
	}
	return c
}
```