שאלה 1 (15 נק')

הכפילו את שני הפולינומים הבאים:

$$p(x) = 7x^3 - x^2 + x - 10$$

$$q(x) = 8x^3 - 6x + 3$$

א. (3 נק') בשיטת ה-"כפל ארוך".

ב. (12 נק') תוך שימוש בהצגה בנקודות:

i. (1 נק') חשבו את שורשי היחידה מדרגה 8

ii. (3 נק') חשבו את ערכי p(x), q(x) והפולינום שמצאתם בסעיף א בכל אחד מהשורשים הנ"ל

iii. (2 נק') ודאו שהתוצאה בסעיף ii עיקבית. כלומר, שהערך שמתקבל מההצבה של כל אחד מהשורשים בפולינום שמצאתם בסעיף א אכן שווה למכפלה של הערכים המתקבלים עבור הצבה של אותו שורש ב-p(x) וב-q(x).

iv. (6 נק') חשבו, באמצעות DFT^{-1}, את מקדמי פולינום המכפלה.

א.

X^6	X^5	X^4	X^3	X^2	X^1	X^0	
			21	-3	3	-30	p * q[1]
		-42	6	-6	60		p * q[X]
	0	0	0	0			p * q[X^2]
56	-8	8	-80				p * q[X^3]
56	-8	34	-53	-9	63	-30	result

ב.

תת-סעיף 1: מכיוון שהחזקה הגבוהה ביותר היא 6, חזקה 2 שתשלים אותה היא 3:

$8 = 2^3 >= 6$

```
for(step = 0; step < 8; step++)
        double rad = 2*PI*step;  // the angle in radian
        result[step] = e^(rad*i/8) = cos(rad/8) + i*sin(rad/8);

result = {1, (1+i)/sqrt(2), i, (-1+i)/sqrt(2), -1, (-1-i)/sqrt(2), -i, (1-i)/sqrt(2)}
```

(המשך סעיף ב' בהמשך)

תת-סעיף 2: (חושב ע"י www.symbolab.com)

	X_0=1	X_1=(1+i)/sqrt(2)	X_2=i	X_3(-1+i)/sqrt(2)
p	-3	-(sqrt(2)*3 + 10) + i*(sqrt(2)*4 - 1)	-9 – 6i	(sqrt(2)*3 - 10) + i*(sqrt(2)*4 + 1)
q	5	(-sqrt(2)*4 + 3) + i*(sqrt(2)*4 - 6)	9 – 8i	(sqrt(2)*4 + 3) + i*(sqrt(2)*4 + 6)
pq	-15	(-44 + sqrt(2)*59) + i*(-sqrt(2)*6 + 1)	-129 + 18i	-(44 + sqrt(2)*59) - i*(sqrt(2)*6 + 1)

	X_4=-1	X_5=(-1-i)/sqrt(2)	X_6=-i	X_7=(1-i)/sqrt(2)
p	-19	(sqrt(2)*3 - 10) - i*(sqrt(2)*4 + 1)	-9 + 6i	-(sqrt(2)*3 + 10) + i*(-sqrt(2)*4 + 1)
q	-11	(sqrt(2)*4 + 3) - i*(sqrt(2)*4 + 6)	9 + 8i	(-sqrt(2)*4 + 3) + i*(-sqrt(2)*4 + 6)
pq	209	-(sqrt(2)*59 + 44) + i*(sqrt(2)*6 + 1)	-(129 + 18i)	(-44 + sqrt(2)*59) + i*(sqrt(2)*6 - 1)

תת-סעיף 3: הנוסחה הכללית היא:

$A_j = n^{-1} * \{sigma\_between(k=0 \text{ until } n-1): y_k*w^{-jk}\}$

הנוסחה במקרה שלנו:

$A_j = 8^{-1} * \{sigma\_between(k=0 \text{ until } 7): pq[X_k]*(X_k)^{-j}\}$

כלומר, עבור האיבר הראשון A_0 (by calc in www.symbolab.com):

$A_0 = 8^{-1} * \{sigma\_between(k=0 \text{ until } 7): pq[X_k]*(X_k)^{-0}\} = 8^{-1} * \{sigma\_between(k=0 \text{ until } 7): pq[X_k]*1\} = -30$

האיברים הבאים:

$A_1 = 8^{-1} * \{sigma\_between(k=0 \text{ until } 7): pq[X_k]*(X_k)^{-1}\} = 63$
$A_2 = 8^{-1} * \{sigma\_between(k=0 \text{ until } 7): pq[X_k]*(X_k)^{-2}\} = 63$
$A_3 = 8^{-1} * \{sigma\_between(k=0 \text{ until } 7): pq[X_k]*(X_k)^{-3}\} = -9$
$A_4 = 8^{-1} * \{sigma\_between(k=0 \text{ until } 7): pq[X_k]*(X_k)^{-4}\} = -53$
$A_5 = 8^{-1} * \{sigma\_between(k=0 \text{ until } 7): pq[X_k]*(X_k)^{-5}\} = -34$
$A_6 = 8^{-1} * \{sigma\_between(k=0 \text{ until } 7): pq[X_k]*(X_k)^{-6}\} = -8$
$A_7 = 8^{-1} * \{sigma\_between(k=0 \text{ until } 7): pq[X_k]*(X_k)^{-7}\} = 56$

<u>סוף שאלה 1!</u>

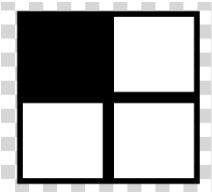
שאלה 2 (10 נק')

נתון ריבוע בדף משובץ עם צלע הבנויה מ- 2^n משבצות. בריבוע חסרה משבצת אחת באחת הפינות. בנוסף נתונים אריחים הבנויים מ-3 משבצות בצורה הבאה (ניתן לסובב ולהפוך את האריחים):

א. (2 נק') הוכיחו שניתן לכסות (לרצף) את הריבוע המקורי באריחים שכאלו לכל n טבעי.

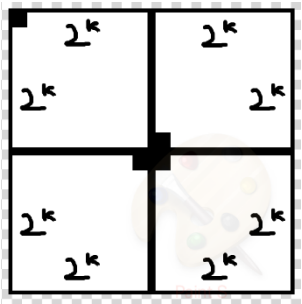
ב. (8 נק') הגדירו אלגוריתם שמוצא כיסוי שכזה. מהי סיבוכיות האלגוריתם?

א. נוכיח ע"פ אינדוקציה:



מקרה בסיס: עבור n = 1 נקבל ריבוע 2*2 כאשר אחת המשבצות חסרה. דבר הנותן את צורת האריח עצמה.

הנחת האינדוקציה: נניח כי עבור כול n המקיים: n <= k, כאשר n, k מספרים טבעיים (גדולים מ-0), הריבוע שצלעו 2^n בר-חיסוי ע"י מספר שלם של אריחים.



נסתכל על המקרה n = k+1 שלכאורה לא כלול בהנחה, ונוכיח שניתן לחסות אותו. כלומר נקבל ריבוע שצלעו היא 2^{k+1} משבצות. נבין שכול צלע היא 2^k*2 ע"פ חוקי חזקות.
כלומר הריבוע החדש מורכב מ-4 ריבועים שצלע כול אחת מהן שווה ל-2^k, כפי שמצויר משמאל.
נעביר צורה אחת שתשלים ריבוע 2*2 במרכז הריבוע ביחס למשבצת החסרה בריבוע הגדול.
כרגע קיבלנו 4 ריבועים בעלי צלעות של 2^k עם משבצת חסרה וע"פ הנחת האינדוקציה ניתן לחסות כול אחד מהם => ניתן לחסות את הריבוע 2^{k+1}.

מ.ש.ל

(סעיף ב' בהמשך)

ב.

```
cover(Square s, Point p) {

        if(s.dim == 2*2)
                coverBasic(s);
        else {
                // getRelativeCenter returns Point array [size = 3]
                // which complete center

                Point complementCenters[3];
                complementCenters = getRelativeCenter(s, p);
                Square s1 = getQuarterSquareBasedOnPoint(s, p);
                Square complementSquares[3];

                for(i = 0 to 3)
                        complementSquares[i] =
getQuarterSquareBasedOnPoint(s, complementCenters[i]);

                cover(s1, p);
                for(i = 0 to 3)
                        cover(complementSquares[i],
complementCenters[i])

        }

}
```

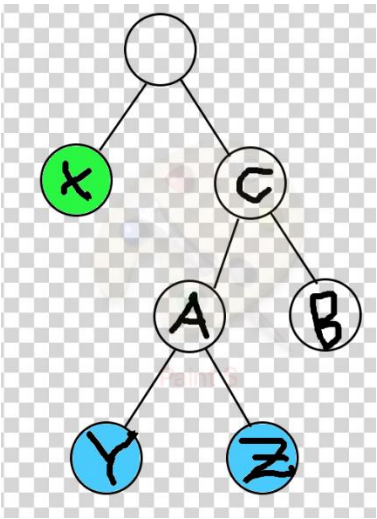
<u>סוף שאלה 2!</u>

שאלה 3 (15 נק')

יהי C א"ב בגודל k. לכל $1 \le i \le k$ נסמן ב-f_i את השכיחות של התו c_i בטקסט מסויים. נתון כי השכיחויות מקיימות $f_1 > f_2 > f_3 > \cdots > f_k > \frac{f_1}{2}$.

הוכיחו שבעץ הופמן המתאים אין שני עלים הנמצאים בעומקים בהפרש של 2 או יותר. במלים אחרות, בעץ הופמן T לכל $1 \le i, j \le k$ מתקיים $-1 \le d_T(c_i) - d_T(c_j) \le 1$.

נוכיח את הטענה ע"י הנחת השלילה.
נתון* (אשר נתון בשאלה): $f_1 > f_2 > f_3 > \ldots > f_k > f_1/2$
נניח בשלילה כי קיים עץ הופמן המתאים כך שקיימים בו שני עלים הנמצאים בעומקים בהפרש של 2 או יותר.
נשרטט אותו:



נגדיר את הטענות הבאות ע"פ סדר מחשבתי:

1) מהנתון בשרטוט: X, Y, Z הינם צמתים השייכים לעץ הופמן שבנינו, כאשר **X בהכרח עלה** (Y ו-Z לא חייבים להיות עלים כחלק מסיפוק התנאי: "שני עלים הנמצאים בעומקים בהפרש של 2 **<u>או יותר</u>**.")
2) לא ניתן לדעת דבר בעניין היחסים שבין השכיחויות f_x, f_c
3) ע"פ אלגוריתם לבניית עץ הופמן: $f_A, f_B <= f_X$
4) ע"פ אלגוריתם לבניית עץ הופמן: $f_Y + f_Z = f_A$
5) על סמך טענות 2 ו-3: $f_Y + f_Z <= f_X$
6) מנתון* ניתן להסיק: כול עלה i בעץ הופמן מוכרח לקיים: $f_1/2 < f_i$
7) ע"פ אלגוריתם הופמן, ידוע שכול צומת בעץ שהוא אינו עלה הוא סכום של שכיחויות של מספר (הגדול מ-1) עלים. בנוסף על הסתמכותנו על טענה 6 ניתן לומר כי כול צומת j בעץ הופמן אשר בנינו מקיים: $f_1/2 < f_j$
8) מטענה 7 נובע: כול שני צמתים i, j בעץ הופמן מקיימים: $f_i + f_j > f_1/2 + f_1/2 = f_1$ --> $f_i + f_j > f_1$
9) המקרה הפרטי של טענה 8: $f_Y + f_Z > f_1$
10) מטענות 5 ו-9: $f_X >= f_Y + f_Z > f_1$ --> $\mathbf{f_X > f_1}$

נוצרה סתירה לוגית בין טענה 10 לנתון* כי אין אף עלה בעל שכיחות גבוהה או שווה ל-f_1, משמע אין לעלה X שום שכיחות שתתאים לו. לכן הוא לא יכול להתקיים בעץ שלנו. ועל כן עץ הופמן אשר ציירנו גם כן לא קיים, בסתירה להנחת השלילה כי קיים עץ כזה.
לכן הנחת השלילה קרסה.
מ.ש.ל

שאלה 4 (20 נק')

נתון מערך של N מספרים ממשיים $[a_1, a_2, a_3, \ldots, a_N]$ (ייתכנו גם שליליים). רוצים לחלק את המערך ע"י K מחיצות ל- $K+1$ קטעים רציפים זרים:

$$[a_1, a_2, \ldots, a_{i_1}], [a_{i_1+1}, a_{i_1+2}, \ldots, a_{i_2}], [a_{i_2+1}, a_{i_2+2}, \ldots, a_{i_3}], \ldots, [a_{i_K+1}, a_{i_K+2}, \ldots, a_N]$$

נמספר את הקטעים בסדר הנ"ל. נגדיר משקל של קטע j -י להיות סכום הערכים בקטע, כלומר $w_j = a_{i_{j-1}+1} + a_{i_{j-1}+2} + \ldots + a_{i_j}$.

כובד החלוקה מוגדר כמקסימום של משקלות הקטעים בחלוקה. אנו מעוניינים בחלוקה בעלת כובד מינימלי.

לדוגמא: עבור המערך $[10,\ 2,\ 1,\ 13,\ 1]$ יתכנו כמה חלוקות אפשריות ל-3 קטעים (כלומר, $K=2$):

1) $[10],[2],[1,\ 13,\ 1] \Rightarrow w_1 = 10, w_2 = 2, \underline{w_3 = 15}$
2) $[10,2],[1,\ 13],[1] \Rightarrow w_1 = 12, \underline{w_2 = 14}, w_3 = 1$
3) $[10,2,1],[13],[1] \Rightarrow \underline{w_1 = 13}, \underline{w_2 = 13}, w_3 = 1$

⋮

כאן, כובד החלוקה (1) הוא 15, לחלוקה (2) כובד שווה ל-14, וכובד החלוקה השלישית – 13 – הוא המינימום.

הציעו אלגוריתם יעיל בתכנות דינמי למציאת חלוקה עם כובד מינימלי (בין כל האפשרויות).
חשבו את סיבוכיות האלגוריתם המוצע.

```
// W is the array of numbers
// g is num of groups the user wish to divide W
Q4(int[] W, int g)
{
        int Sums[W.length][W.length];
        int T[g][W.length];

        for m=0 to W.length-1 do:
                Sums [m][m] = W[m];

        for i=0 to W.length-1 do:
                for j=i+1 to W.length-1 do:
                        Sums [i][j] = Sums [i][j-1] + W[j]

        for j=0 to W.length-1 do:
                T[0][j] = Sums [0][j];

        int temp[];
        for r=1 to g-1, do:
                for c=r to W.length-1, do:
                        // num of optimal options for divide r time W':
                        for k=r-1 to c-1, do:
                                min_sol = T[r-1][k];
                                sum = Sums[k+1][c];
                                temp[k-(r-1)] = max(min_sol, sum);
                        T[r][c] = min(temp);

}
```

סיבוכיות האלגוריתם:
ע״י מתמטיקה הגענו לסיבוכיות הבאה:

ע״י אתר symbolab ואתר brilliant (כול אחד בבדיקה נפרדת), הגענו למסקנה כי הסיבוכיות היא: $O(n^3)$ במצב הגרוע ביותר

$$\sum_{i=2}^{n-1} ((n+3-i)\cdot(n-i))\cdot 0.5$$

אולם, כשניסינו להריץ את האלגוריתם במחשב (בשפת גאווה) על קלטים n_1=10,000 n_2 = 20,000 עבור מספר קבוצות החלוקה קטן (5), הופתענו לראות כי מספר האיטרציות בכול אחד מהקלטים בהתאמה היה: $8*10^8$ | $2*10^8$, שדומה דווקא לאלגוריתם בעל סיבוכיות $2n^2$, דהיינו מהסוג $O(n^2)$. בנוסף לכך, ניתן לראות כי הקפיצה במספר האיטרציות בין n_1 ל-n_2 הייתה צריכה לכאורה לגדול פי 8, אבל בפועל הייתה קפיצה של פי 4 בלבד. תואם לאלגוריתם מסוג $O(n^2)$.
הסקנו מכך כי בעצם <u>הסיבוכיות $O(n^3)$ אינה מתארת כמו שצריך את סיבוכיות האלגוריתם</u>.
לא

הגדרנו k כמספר קבוצות החלוקה שדרשנו, משמע הוא מעיין קבוע מקומי (אל אף שהוא משתנה במימד האוניברסלי). במקרה שבדקנו בניסוי k = 5. נדרוש נוסחה חדשה התואמת לשינויים הקיימים:

$$\sum_{i=2}^{k+2} ((n+3-i)\cdot(n-i))\cdot 0.5$$

ועל כן קיבלנו כי סיבוכיות האלגוריתם היא: **$O(k*n^2)$**
נוסחה זו מתיישבת עם התוצאות עבור כול שינוי שהו (בפרט השינוי בניסוי שלנו) ואף במצב קיצון המתואר בנוסחה הראשונה הנ״ל שפיתחנו.

שאלה 5 (10 נק')

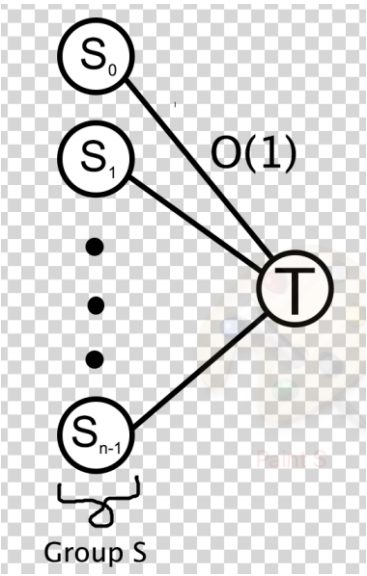
תנו דוגמה למשפחות של רשתות זרימה עם n קדקודים כך ש:

א. לרשתות הזרימה במשפחה יש חתך מינימלי יחיד
ב. לרשתות הזרימה במשפחה יש O(n) חתכים מינימליים שונים
ג. לרשתות הזרימה במשפחה יש $O(n^2)$ חתכים מינימליים
ד. לרשתות הזרימה במשפחה יש O(n x log n) חתכים מינימליים
ה. לרשתות הזרימה במשפחה יש $O(2^n)$ חתכים מינימליים

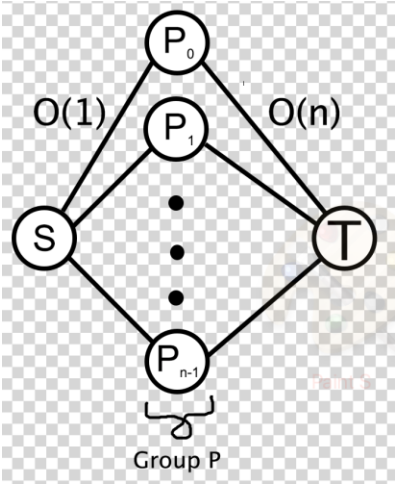
לכל משפחה יש לתאר מהי רשת הזרימה המתאימה ל-n ולהסביר מדוע לרשת המתקבלת יש את הכמות הנדרשת של החתכים המינימליים.

הגדרה לחתך מינימום:
חתך מינימום הוא חתך (משמע מחלק את קבוצת הקדקודים הכוללת ברשת זרימה לשתי קבוצות: האחת שכוללת את המקור, והשנייה את הבור) כך ששתי הקבוצות מחוברות ע״י לפחות קשת אחת, כך שסכום קיבולי קשת/ות אלו יהיה מינימלי.

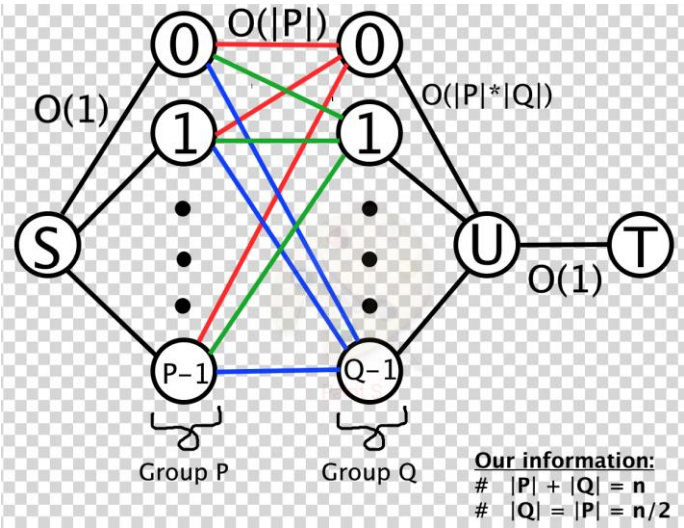
סעיף א:
רשתות זרימה 0/1, כמו שניתן לראות באיור מצד שמאל, ישנם n מקורות, ובור אחד. ישנו רק חתך מינימלי אחד, והוא: (Group S, T)



סעיף ב.
ניקח את רשת הזרימה מסעיף א׳ ונחבר לכול מקור באמצעות קשת התואמת רשתות זרימה 0/1 לקודקוד S, ובכך הגדלנו את מספר החתכים המינמליים ל-(2*n+1) כאשר הינו קבוע.
לכן מספר החתכים המינימליים חסום ע״י O(n).



ג.



במשפחת רשתות הזרימה המתוארת להעיל, כך שמדובר ברשת זרימה (0,1), ישנם n קודקודים שמשתנים (שלושת הקודקודים S, U, T אינם משתנים, מלבד מספר הקשתות הנכנסות ויוצאות מהן). אנו מניחים כי n הינו מספר זוגי לצורך הפשטת ההוכחה.
אזי: ישנם O(n + 3) = O(n) קודקודים. נוכיח כי מספר החתכים המינימליים שווה ל-$O(n^2)$.
חצינו את קבוצה n לשתי קבוצות שוות בגודלן (בהנחה כי n זוגי). כמו שניתן לראות בציור, הקשתות שבין קודקוד P לקודקודי קבוצה Q מעניקה לנו O(|P|*|Q|) חתכים מינימליים שונים. ניתן להוכיח זאת ע״י שימוש בעקרונות הסידור באלכסון קנטור:
(S, p from P, q from Q)

(המשך בדף הבא)

(S,0,[0]), (S,0,[0,1]), (S,0,[0,1,2]) … (S,0,[0,…,q-1])

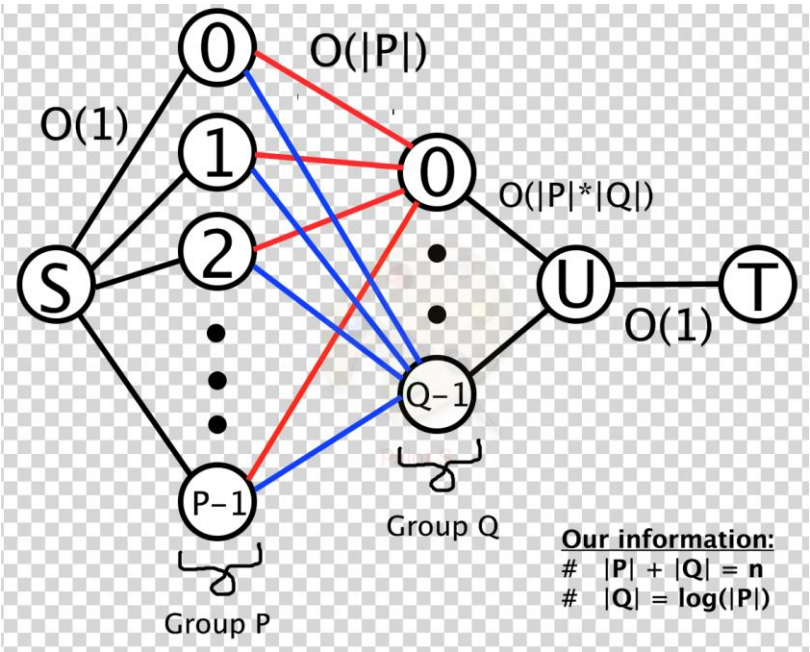
(S,[0,1],[0]), (S,[0,1],[0,1]), … (S,[0,1],[0,…q-1])

……

(S,[0…p-1],[0]), (S,[0…p-1],[0,1]), … (S,[0…p-1],[0,…q-1])

מכיוון שהחתך אינו בהכרח חייב להתחיל מהאיבר ה-0, אזי יש לנו מספר אפשרויות כפול ממה שתיארנו. ועדיין, אנו תחומים תחת O(|P|*|Q|) חתכים מינימליים. משמע:
min_cuts = O(1) + O(|P|) + O(|P|*|Q|) = O(1 + n/2 + n^2/4) = O(n^2)
בהסרת ההנחה כי n זוגי, מדובר בפלוס/מינוס של O(n/2) חתכים מינימליים שיכולנו להפיק או לפסיד. ועדיין: O(n^2) – O(n/2) = O(n^2)

ד.



כנ״ל כמו בסעיף ג׳, גם כאן ישנם S, U, T קודקודים קבועים, ו-n קודקודים שמשתנים. כמו כן, בדומה לסעיף קודם, אנו מחלקים את n הקודקודים לשתי קבוצות: P, Q. רק שבניגוד לסעיף הקודם, השתנו מאזני הגדלים שבין הקבוצות P, Q. כמו הוכחה בסעיף קודם, ששם הוכחנו כי מספר החתכים המינימליים חסום ע״י O(|P|*|Q|), ההוכחה תקפה לשם הוכחת

הטענה במקרה זה:
מספר החתכים המינימליים חסום ע״י O(n*log(n)).

(ההוכחה בדף הבא)

הוכחה:
מהמידע שבצד ימין למטה שבתמונה, ניתן להסיק: $|P| + \log(|P|) = n$. נפעיל את פעולת ה-O() על שני האגפים, ונקבל:

$O(n) = O(|P| + \log(|P|)) \rightarrow$ **$O(n) = O(|P|)$**

מהנתון השני שבשרטוט בנוסף לטענה: **$O(n) = O(|P|)$** ניתן להסיק:

$O(|Q|)$ $= O(\log(|P|))$ = **$O(\log(n))$**

טענו (והוכחנו) כי מספר החתכים המינימליים במשפחה רשתות-הזרימה הנ"ל חסום ע"י $O(|P|*|Q|)$. לכן:

$O(|P|*|Q|) = O(n*\log(n))$

ה.
לא קיימת רשת-זרימה כזו (ובטח שלא משפחה שלמה), מכיוון שצריכה להיות קשת מפרידה לכל חתך וישנן $O(n^2)$ קשתות, על כן כמות החתכים המינימליים תחומה $O(n^2)$.

<u>סוף שאלה 5!</u>

שאלה 6 (20 נק')

א.

The ***0-1 knapsack problem*** is the following. A thief robbing a store finds n items. The ith item is worth v_i dollars and weighs w_i pounds, where v_i and w_i are integers. The thief wants to take as valuable a load as possible, but he can carry at most W pounds in his knapsack, for some integer W. Which items should he take? (We call this the 0-1 knapsack problem because for each item, the thief must either take it or leave it behind; he cannot take a fractional amount of an item or take an item more than once.)

Give a dynamic-programming solution to the 0-1 knapsack problem that runs in $O(n\,W)$ time, where n is the number of items and W is the maximum weight of items that the thief can put in his knapsack.

ב. ליוסי n פריטים שונים. משקלו של כל פריט הינו w_i ק"ג. יוסי צריך לארוז את הפריטים בארגזים, כאשר מותר לארוז בארגז משקל מקסימלי של עד W ק"ג. תארו אלגוריתם המקבל כקלט את משקלי הפריטים והמשקל המקסימלי שמותר לארוז בארגז ומחזיר את מספר הארגזים המינימלי שעל יוסי להזמין.

סעיף א:

הנחות:

-לכול אביזרים מובנה משקל וערך משלו בטרם עת.

-הגבלת המשקל חייבת להיות >=0

-כול אביזר בעל משקל חיובי (מבחינה הגיונית), על אף שהאלגוריתם עובד גם עבור אביזרים בעלי משקל 0. משקלים שליליים אסורים.

- מערך האביזרים אינו חייב להיות מסודר ע"פ משקל ו/או ערך.

(האלגוריתם בנוי בדף הבא מכיוון שהוא די ארוך, וכדי לשמור על רצף קריאות, החלטנו לשלב אותו בדף נפרד)

```
knapsack_algorithm(item[] items, int weightLimit)
{
	// items: each item contains weight and value
	// weightLimit: the total limited weight

	int T[items.length+1][ weightLimit+1]  // The matrix which will contains
the solution

	// when the number of the items you can take is 0:
	for i=0 to items.length do:
		T[i][0] = 0;

	// when the limited weight is 0:
	for j=0 to weightLimit do:
		T[0][j] = 0;

	for i=1 to T.length do:
		for j=1 to T[0].length do:
			item = items[i-1];
			if(j < item.weight)
				T[i][j] = T[i-1][j]
			else
				T[i][j] = max(item.value + T[i-1][j-item.weight), T[i-1][j])

	// getting the indexes of the items in items[] which the algorithm decide
to take
	ArrayList<Int> solutionIndexes;
	i=T.length, j=T[0].length;
	while i>0 and j>0 do:
		while T[i][j] == [i-1][j] do:
			i--;
		solutionIndexes.add(i-1);
		j -= items[i-1].weight;
		i--;

	// “getItemsByIndexes” returns an array of all the items which their
indexes are in.
	return getItemsByIndexes(items, solutionIndexes);
}
```

(סעיף ב בהמשך)

סעיף ב:

הנחות:
כול משקל של פריט חייב להיות =< W (כלומר, חיוב קיום פתרון אפשרי (לא בהכרח מינימלי) שחסום מלמעלה, והוא שמספר הסלים המושמים שווה למספר המשקלים שנשלחו (= מספר הפריטים))

הבעיה של יוסי ניתנת לפתרון בעזרת הפיתרון המוצא בסעיף א׳ לבעיית ה-knapsack.
קיבלנו מיוסי מערך של משקלים של פריטים. נגדיר מערך של פריטים items בגודל השווה למספר המשקלים, כך שלכול פריט בו, ערכו יהיה שווה בגודלו למשקלו. דהיינו: item.value = item.weight.
נבנה לולאה שתוגדר כך:
כול עוד רשימת הפריטים items לא ריקה, נקיים את האיטרציה הבאה:
ניקח סל שניתן לשים בו משקל של עד W (בדיעבד: +1 למונה-סופר שיספור את מספר הסלים שהשתמשנו בהם)
נקרה לפונ׳ מסעיף א:

```
Item[] itemsInSack = knapsack_algorithm(items, W);
```

שתחזיר את רשימת הפריטים שנכנסו לסל. כעט נוציא את אותם פריטים שהוכנסו לסל מרשימת הפריטים הכוללת:

```
removeFrom(items, itemsInSack);
```

סוף האיטרציה. נחזור חלילה על תנאי הלולאה.
לבסוף נחזיר את תוצאתו של המונה-סופר.

<u>סוף שאלה 6!</u>

<u>רפלקציה:</u>
שאלות 2,3 היו יחסית פשוטות (אבל עדיין טריקיות) ביחס לאחרות. עיניין של הוכחות זו המומחיות שלנו.
שאלות 4 ו-6 תוכנתו על ידינו, ואנחנו יודעים שהצלחנו בהן בפועל. התכנות עזר מאוד להבנת האלגוריתם. כתיבת התוכנות לקחה הרבה זמן, אבל זה היה מאתגר ומעניין מאוד עבורנו.
שאלה 1 ארוכה מאוד (ארוכה מדיי), ומדובר בהמון עבודה שחורה וסתמית. היה עדיף להביא שני פולינומים כך שמכפלתם תביא פולינום ממעלה 3, ובכך לקצר מצורה דרסטית את העבודה השחורה, ולהעמיק יותר באלגוריתם DFT.
שאלה 5 מתחכמת, אבל בסוף הצלחנו ע״י חזרה אין סופית על ההגדרה מה הוא חתך מינימלי.