שאלה 1:

בבעיית *הגיבוי* נתון גרף דו-צדדי לא מכוון $G = (C, S, E)$, כאשר C היא קבוצת לקוחות ו- S היא קבוצת שרתים. לכל לקוח $c \in C$ יש קובץ שהוא רוצה לגבותו באחד משכניו השרתים (כל הקבצים באותו גודל), וכל שרת s יכול לאחסן עד $\ell(s)$ קבצים.

פתרון לבעיה הוא פונקציה $b: C \to S$ שמתאימה שרת לכל לקוח, כך שמתקיים:

- לקוח מקבל שירות רק משרת שכן, כלומר $(c, b(c)) \in E$, לכל לקוח c.
- העומס על כל שרת חסום ע"י מגבלת העומס שלו, כלומר $|\{c : b(c) = s\}| \leq \ell(s)$, לכל שרת s.

המטרה בבעיית הגיבוי היא למצוא פתרון שמביא למקסימום את מספר הלקוחות שקיבלו שירות.

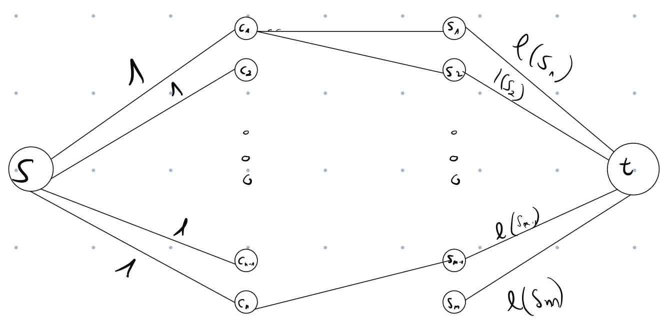
א. הראו שבעיית שידוך המקסימום בגרף דו-צדדי היא מקרה פרטי של בעיית הגיבוי.

ב. תארו אלגוריתם יעיל שפותר את בעיית הגיבוי.
נתחו סיבוכיות והוכיחו נכונות.

א) אפשר לראות שבעיה זו ניתנת שבעיית השרתים ע"י שידוך בתנאים רק שלא אבל בשמאל (לקוחות) יש התאמה

אחת בדיוק לצד ימין (שרתים). וצד ימין יכול לקבל כמות של עד [קיבולת של שרת הינה $\ell(s) \geq 0$]

נבנה את הרשת:



ב) מכיוון שהמרחק הינו שווה בכל מסלול (מרחק 3), אפשר להשתמש ב ford-fulkerson, שידוע שזמן הריצה שלו $O(E \cdot maxf)$.

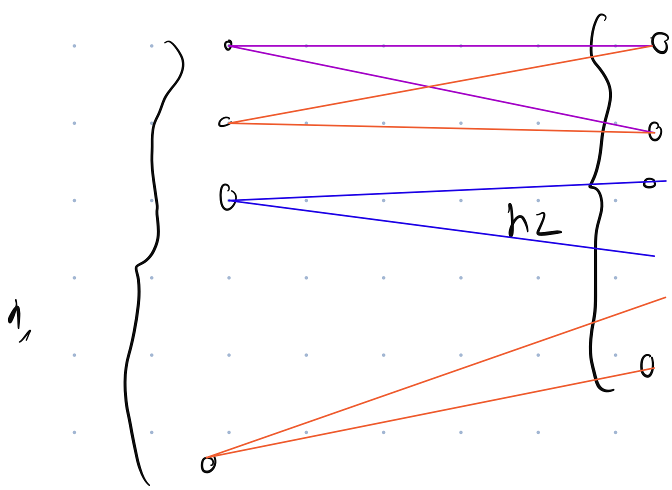
את הזרם המקסימלי יוצרים לפי החתך המינימלי, שהוא מ $s \to c$ שהוא $|c|$ שהוא n.

שאלה 2:

יהי $G=(L,R,E)$ גרף דו צדדי שבו דרגת כל צמת היא בדיוק d, עבור $d>0$.
שידוך מושלם הוא שידוך שבו כל צמת נוגע בקשת שידוך.

א. הוכיחו שמתקיים $|L|=|R|$.

ב. הוכיחו שקיים ב- G שידוך מושלם.
<u>הדרכה:</u> בנו רשת זרימה מתאימה, והראו קיום של זרימה שערכה $|L|$.



יהי G גרף שידוך דו צדדי כך ש

קיים $n_1=|L|$ ו $n_2=|R|$ אשר בעלות דרגה d

כך ש $d>0$.

נניח ש $n_1 \neq n_2$.

אזי לכל $l \in L$ מחובר ל $\{r_1,\dots,r_d\} \in R$

שאלה 2

נגדיר את הקבוצה

$$O_r = \{\forall r \in R : f(r) \to \{e \in E \wedge \ell \in L : \exists e \in (r,\ell)\}\}$$

לכל איבר r נגדיר קבוצה של d השכנים שלו, בעצם יש לנו קבוצה של קבוצות.

$$O_r = |R|$$

מכיוון שבגלל של כל איבר r.

נגדיר באותו אופן את $O_\ell = |L|$.

אם נפתח את הקבוצה לפי קבוצה אחת נגדיר $T_r = \{\dots O_r\}$

נקבל גודל קבוצה $|T_r| = d\cdot|O_r|$ מכיוון של r ב R אנחנו מייצרים בדיוק d קבוצות.

נראה $\forall \ell \in L$ מתקיים ℓ_i מופיע d פעמים בדיוק T_r.

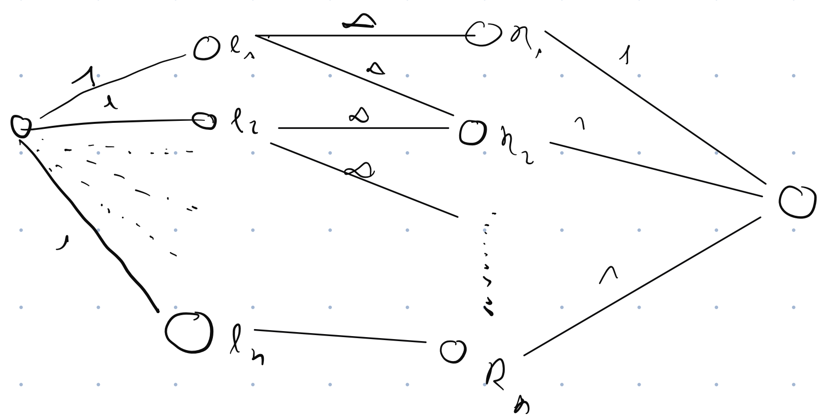
נניח בשלילה שקיים ℓ_i כך שהוא מופיע ב T_R כמות אשר שונה מ d, נגדיר כמות זו כ p.

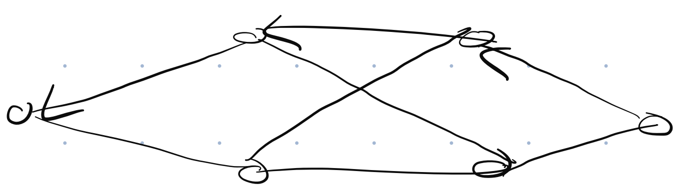
אם $p \neq d$ אזי הדרגה של G כבר היא לא d, בסתירה לנתון.

לכן אפשר לומר ש $|T_R| = d\cdot|L|$ שבעצם נקבל

$$|T_R| = d\cdot|O_R| = d\cdot|R| = d\cdot|L|$$

שאלה 2 סעיף ב.





נבנה רשת שיורית לאחר הורדה k קשתות ממקור
שיצאו

שאלה 3 ב

נשתמש במשפט החתונה על מנת להוכיח טענה, משפט החתונה אומר ש לגרף $G=(S,T,E)$, הוא גרף דו צדדי, קיים בו שידוך את S אם לכל תת קבוצה $W\in S$ מתקיים ש $|W|\le|N(W)|$, אזי קיים התאמה מושלמת

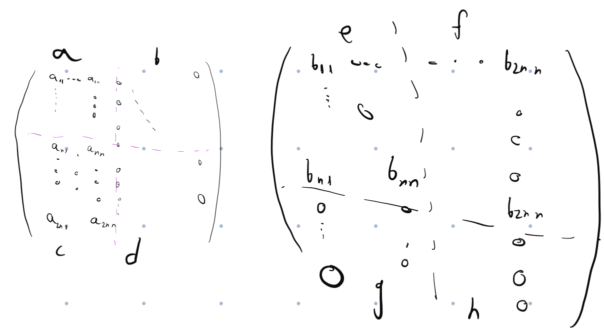
נתון ש $d\ge1$ ולכן $|W|\le d\cdot|W|\le|W|$ ולכן יש התאמה.

שאלה 3 סעיף א

נסביר כיצד ניתן לחשב את התוצאה של $C^{2n\times 2n} = A^{2n\times n}\cdot B^{n\times 2n}$ על ידי אלגוריתם Strassen, כאשר נצטרך להשלים את שני המטריצות כך שנוכל להשתמש באלגוריתם Strassen. נבצע זאת ע"י הוספת אפסים וליצור שני מטריצות ריבועיים בגודל $2n\times 2n$.

לאחר מכן נחשב את $P_1 \dots P_7$, ונתחיל את החישובים בצורה רקורסיבית.

$$P_1=(a+d)(e+h)=a\cdot e$$
$$P_2=d(g-e)=0$$
$$P_3=(a+b)h=0$$
$$P_4=(b-d)(g+h)=0$$
$$P_5=a(f-h)=a\cdot f$$
$$P_6=(c+d)\cdot e=c\cdot e$$
$$P_7=(a-c)(e+f)$$



$$\{b,d,g,h\}=0$$

שאלה 3 סעיף ב

$$A^{4\times 2} \times B^{2\times 4} = C^{4\times 4}$$

מה שקרה פה זה שלב הראשון של הריקורסיה נשתמש
במימדי המטריצה הידועים על מנת לחסוך בחישובים.
זה נותן לנו בצורה את הצורך לחשב רק $\{P_1, P_5, P_7\}$.
נעבור על כל אחד בצורה נפרדת על מנת לחשב זו.

$P_1 = (a+d)(e+h) = a\cdot e$

$P_2 = d(g-e) = 0$

$P_3 = (a+b)h = 0$

$P_4 = (b-d)(g+h) = 0$

$P_5 = a(f-h) = a\cdot f$

$P_6 = (c+d)\cdot e = c\cdot e$

$P_7 = (a-c)(e+f)$

$P_1 = a^{2\times 2} \times e^{2\times 2} = 8$ פעולות חישוב

$P_5 = a^{2\times 2} \times f^{2\times 2} = 8$ פעולות חישוב

$P_7 = a^{2\times 2}e^{2\times 2} + a^{2\times 2}f^{2\times 2} - c^{2\times 2}e^{2\times 2} - c^{2\times 2}f^{2\times 2} = 48$ פעולות חישוב

לאחר חישוב $P_1 \dots P_7$ כמות הפעולות הנוספות לחישוב המטריצה.

$$\begin{pmatrix} a & b \\ c & d \end{pmatrix}\begin{pmatrix} e & f \\ g & h \end{pmatrix} = \begin{pmatrix} c_{11} & c_{12} \\ c_{21} & c_{22} \end{pmatrix}$$

$c_{11} = P_1 + P_2 - P_3 + P_4$

$c_{12} = P_5 + P_3$

$$\left.\begin{aligned} c_{21} &= P_6 + P_2 \\ c_{22} &= P_5 + P_1 - P_6 - P_7 \end{aligned}\right\}$$

סה״כ פעולות חישוב

8

$P_7 = a^{2\times 2}e^{2\times 2} + a^{2\times 2}f^{2\times 2} - c^{2\times 2}e^{2\times 2} - c^{2\times 2}f^{2\times 2}$

$a^{2\times 2}e^{2\times 2} = 8 \ , \ c^{2\times 2}e^{2\times 2} = 8 \ , \ c^{2\times 2}\cdot f^{2\times 2} = 8$

$$\underbrace{\underbrace{(ae + af)}_{8\ \text{פעולות}} + \underbrace{(-ce - cf)}_{8\ \text{פעולות}}}_{8\ \text{פעולות}} = 32$$ פעולות

סה״כ כמות ב פעולות 64 פעולות

שאלה 4:

בשאלה זו נוסיף שדה שנקרא size לרשימת דילוגים. משמעות השדה היא שאם יש שני צמתים עוקבים a_1 ו- a_2 באותה רמה, אז השדה size של a_1 יכיל את מספר האיברים ברמה 0 שנמצאים בין a_1 ו- a_2.

א. תארו איך מתחזקים את ערך השדה בפעולות INSERT ו- DELETE.

ב. הסבירו איך ניתן למצוא את סטטיסטי הסדר ה- i ע"י שימוש בשדות ה- size שבמבנה.

נדבר על כל פעולה בנפרד,

כאשר מכניסים צומת אנחנו נצטרך לעדכן את הפעולות הבאות

Inser -

```python
def insert(t, v, level):
    prevEl , nextEl = search(t, v)
    prevParentPath, distanceFromV = searchPath(t, prevEl)
    distanceToNext = prevParentPath.size
    remainingDistance = distanceToNext - distanceFromV
    prevEl.next = Node(v, prevEl.next)
    v.next = nextEl
    for path in prevParentPath:
        if path.dir != 'UP':
            continue
        if path.v <= v and path.next.v >= v:
            path.size += 1
    ADD_TO_TOP = random.choice([True, False])
    if ADD_TO_TOP:
        insert(t.levels[level + 1])

    return t
```

נדגיש שזה mockup

לא קוד אמיתי

אך הרעיון ברור

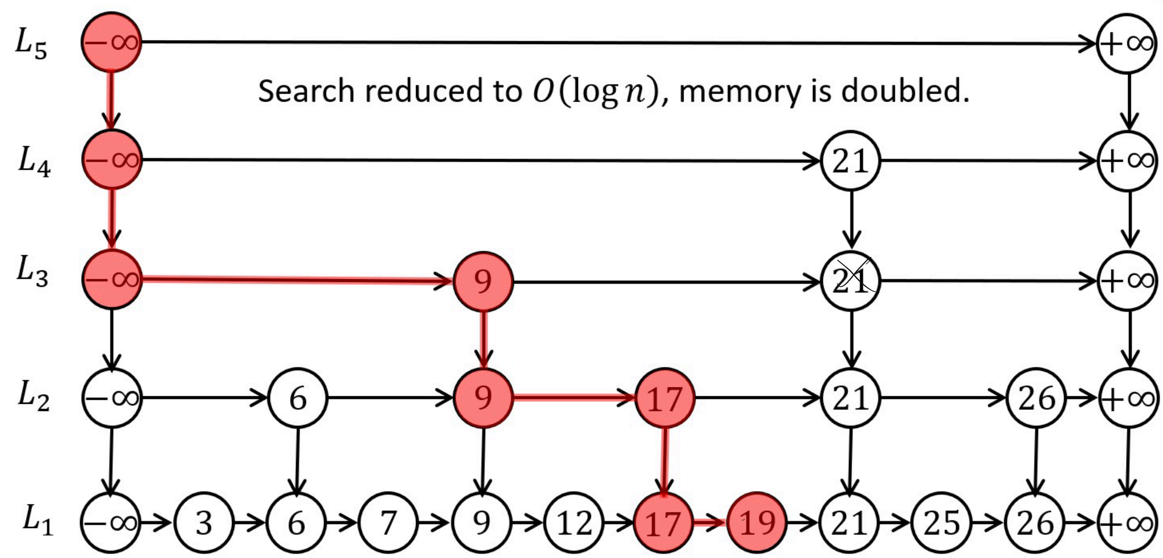
```python
def delete(t, v):
    prevEl , nextEl, didFound = search(t, v)
    prevParentPath, distanceFromV = searchPath(t, prevEl)
    if not didFound:
        prevEl.size - 1
    else:
        prevEl.size += v.size - 1

    prevEl.next = nextEl
    delete(t.levels[1] , v  )
```

כאשר נרצה לעדכן מחיקה

נבצרה רק סיבוב,

גם פה מדובר ב mockup.

```python
def findOrderStatistic(t, k, init):
    start = init
    end = init.next
    if start.size == k:
        return start
    if start.size > k and end.size < k:
        return findOrderStatistic(t, k , start)
    if start.size < k:
        return findOrderStatistic(t, k - start.size, end)
```



L_5
L_4
L_3
L_2
L_1
Search reduced to $O(\log n)$, memory is doubled.
$-\infty$
$+\infty$
3
6
7
9
12
17
19
21
25
26