אלגוריתמים 2

נעם דומוביץ

11 באוגוסט 2019

סיכום זה הוא אוסף של קודים/פסאודו־קודים מהסיכומים של אליזבת, יוסף זוהר, מלא גוגל, *java*, *python* , הקלטות של ואדים, אור (תודה לאביהו), תוספות שלי וכד', מוזמנים לקחת את הכל בערבון מוגבל.

לתיקונים/הערות/הארות - *ndomovich@gmail.com*

בהצלחה לכולנו!

תוכן עניינים

1 הגדרות ומשפטים לגרפים ועצים

תורת הגרפים

- נגדיר $G(V,E)$, כאשר:
  - V - קבוצת קודקודים
  - $E=\{e\}$ - קבוצת צלעות, כאשר:
    - $\{e=(v_i,v_j)\}$ - כלומר כל צלע מחברת שני קודקודים בגרף
  - אם זוג מורכב מזוגות סדורים נאמר שזהו **גרף מכוון**
  - כאשר הצלעות מיצגות זוג לא סדור - נאמר שזהו **גרף לא מכוון**
- **גרף ריק** - אוסף של קודקודים ללא צלעות
- **מסלול בגרף** - סדרת קודקודים בגרף $v_{i_1},v_{i_2},...v_{i_k}$ כך שבין כל זוג של קודקודים סמוכים יש צלע.
- מסלול סגור שמתחיל ומסתיים באותו קודקוד נקרא **מעגל**
- **צלעות מקבילות -** הן מחברות בין אותו זוג של קודקוד הגרף
- **לולאה בגרף** - צלע שמחברת קודקוד עם עצמו
- **גרף פשוט**- גרף לא מכוון ללא צלעות מקבילות, וללא ללואות
- **מסלול פשוט -** מסלול שכל הקודקודים שלו שונים
- **גרף שלם** - גרף שכל קודקוד מחובר לכל קודקוד אחד על ידי צלע
- **גרף קשיר** - בין כל זוג קודקודים קיים מסלול
- **רכיב קשירות** - תת גרף קשיר מקסימאלי
- **גרף לא קשיר -**גרף בו קיים זוג קודקודים שלא קיימת צלע שתחבר אותם.
- **מרחק** - הוא מספר הצלעות המינימאלי (מסלול) בין שני קודקודים
- **אורך מסלול פשוט -** מספר צלעות במסלול
- בגרף לא מכוון **דרגת הקודקוד** הוא מס' הצלעות היוצאות מקדקוד זה נסמן ב $degree\backslash deg$
  - בגרף בעל n קודקודים הדרגה המקסימאלית היא $n-1$
- **קוטר** גרף - המרחק המקסימלי בגרף בין זוג קודקודים כלשהם

עצים

- **עץ** הוא גרף קשיר ללא מעגלים
- הגדרה אלגברית: עץ הוא $\#\left(Com\left(G\right)\right)=1\ and\ |V|=|E|+1$
- **קוטר עץ**: אורך של מסלול ארוך ביותר
- **אקספליסטי** של קודקוד x הוא המרחק הגדול ביותר בין xלכל קודקד אחר: $ex(x)=Max\{dist(v,x),\ v\in V\}$
- **רדיוס** העץ הוא אקספיליסטי המינמלי: $radius\left(T\right)=Min\{ex(v),\ v\in V\}$
- יהי $G=(V,E)$ גרף לא מכוון. **מסלול אוילר** הוא מסלול לא בהכרח פשוט שעובר בכל צלע בגרף בדיוק פעם אחת
- **מעגל אוילר** הוא מעגל לא בהכרח פשוט שעובר בכל צלע בגרף בדיוק פעם אחת.

<u>משפט:</u> בכל גרף (קשיר) יש לפחות שני קודקודים, בעלי אותה דרגה

<u>הוכחה:</u>

- נניח שיש לנו גרף עם n קודקודים
- טווח הדרגות האפשרי הוא מ1 עד $n-1$, לכן אם נגדיר :
  - $n-1$ שובכים - שובך לכל דרגה אפשרית
  - n יונים - יונה לכל קודקוד
- נקבל מעקרון שובך היונים, שיש שני קודקודים בעלי אותה דרגה

<u>משפט:</u> סכום של כל דרגות של קודקודי הגרף שווה לפעמים מס צעלות כלומר:

$$\sum_{i-1}^{n} ged(v_i) = 2|E|$$

הוכחה:

- נספור את הצלעות המחוברות לכל הקודקודים, בכך נקבל את סך הדרגות
  - מהגדרה הצלע, כל קודקוד נספר פעמיים

מסקנה 1: סכם דרגות של כל קדקודי הגרף הוא מס זוגי

מסקנה 2: מס' הקודקודים בעלי דרגה אי־זוגית - זוגי

<u>הוכחה:</u>

- נחלק את כל קדקודי הגרף לשתי קבוצות, כלומר $V = A \cup B$
  - A - קודקודים בעלי דרגה זוגית
    - B - קודקודים בעלי דרגה אי־ זוגית
    - $A \cap B = \phi$
  - מהמשפט:

$$\underbrace{2|E|}_{even} = \sum_{i-1}^{n} deg(v_i) = \underbrace{\sum_{v \in A} deg(v)}_{even} + \sum_{v \in B} deg(v) \Rightarrow \underbrace{\sum_{v \in B} deg(v)}_{even}$$

  - מכיוון B זוהי קבוצת הקודקודים בעל דרגה אי־זוגית, לכן מספר הקודקודים צריך להיות בעצמו זוגי , כנדרש.

<u>עצים</u>

הגדרה: עץ הוא גרף קשיר לא מכוון ללא מעגלים

טענה1 : בעץ לכל שני קודקודים קיים מסלול יחיד

- נניח בשלילה שבין שני קודקודים שונים בעץ יש שני מסלולים שונים
- נמחק את הצלעות המשותפות
- תוצאה: קיבלנו מעגל בסתירה להגדרת העץ

טענה 2: לכל עץ יש לפחות עלה אחד

- נניח בשלילה שלעץ אין עלים $\Leftarrow$ דרגת כל קודקוד בעץ היא לפחות 2
- ונניח שיש לנו n קודקודים
- יהיה קודקוד כלשהו, נסמנו ב v_1.
- מהנחה בשלילה כל קודוקד מחובר ל2 קודקודים נוספים (לפחות) ולכן נוכל לעבור בצלע המשותפת ולהרכיב מסלול $v_1, v_2, v_3, ...$
- אם במהלך הרכבת המסלול , חזרנו על קדוקד שכבר שותף במסלול, אז קיבלנו מעגל - וזו סתירה להגדרת עץ.
- אחרת, אז קיים קודקוד w כך ש $\forall i \in [1,n]\ w \neq v_i$, ולכן הרכבנו מסלול $v_1, ..., v_n, w$ (כי הגרף קשיר), בסתירה לכך שיש רק n קודקודים.

טענה 3 : לכל עץ יש לפחות 2 עלים

- יהיה P מסלול ארוך ביותר בין שני קודקודים, כלומר $P = v_1, v_2, ..., v_k$, P הוא מסלול פשוט (כל הקודקודים שונים זה מזה $\Leftarrow$ אין מעגלים)
- v_1עלה
  - נניח בשלילה שאינו עלה, לכן דרגתו לפחות 2 $\Leftarrow$אז קיים קודקוד נוסף הסמוך אליו נסמנו ב u
  - מהגדרת P כמסלול פשוט נובע ש u אינו שייך לP ולכן ניתן לכתוב: $P' = u, v_1, v_2, ..., v_k$
  - וזו סתירה לכך שאורך P מקסימלי

טענה 4 : לכל עץ בעל n יש בדיוק $n-1$ צלעות

- <u>בסיס:</u> $n=1 \Leftarrow |E| = 0 = 1-1$. $n=2 \Leftarrow |E| = 1 = 2-1$
- <u>צעד - נניח לעץ בעל n ונוכיח ל $n+1$</u>
  - יהיה עץ עם $n+1$ קודקודים. מטענה 2 קיים לו עלה אחד לפחות.
  - נמחק את העלה הזה (צלע + הקודקוד)
  - מהנחת האינדקוציה נקבל עץ עם n קודקדים ו $n-1$צלעות
  - כעת נחשב: $|E| = n-1+1 = n$, כנדרש.

טענה: גרף לא מכוון G הוא עץ $\Longleftrightarrow$ G קשיר מינימלי $\Longleftrightarrow$ G חסר מעגלים

- G קשיר מינימלי - הסבר: קשיר + הורדת צלע כלשהי תפגע בקשירותו
- G חסר מעגלים מקסימלי - הסבר:: כל צלע שנוסיף תסגור מעגל

הוכחה:

1. G גרף לא מכוון הוא עץ $\Longleftrightarrow$ G קשיר מינימלי:

   נניח ש G עץ, G ודאי קשיר, צ"ל להראות שהוא קשיר מינימלי

- נניח בשלילה שאינו קשיר מינימלי, לכן קיימת צלע $e = \{x,y\}$ שאם נוריד אותה מ G הגרף עדיין יהיה קשיר (כלומר הגרף $G \backslash \{e\}$ קשיר)
- בגרף $G \backslash \{e\}$ יש מסלול מ x ל y שכומבן לא כולל את e

- מכאן שאם נוסיף את e נקבל מעגל, בסתירה לכך G הוא עץ, ומהגדרה חסר מעגלים.

נניח ש G קשיר מינמלי צ"ל שהוא עץ, מנתון G קשיר נותר להראות שהוא חסר מעגלים

- נניח בשלילה ש G היה מכיל מעגל, אז אם נסיר צלע ששיכת למעגל, נקבל גרף שהוא עדיין קשיר (הוכחנו בשיעור שעבר)
- סתירה לכך שG קשיר

2. G גרף לא מכוון הוא עץ $\iff$ G חסר מעגלים מקסימלי

ניתן להוכיח מנימוקים דומים

2 בעיית הבקבוקים

נניח ויש לנו 3 מיכלים:

- A בגודל n ונניח שב A יש a ליטרים
- B בגודל m , וננניח ב B יש b ליטרים

<u>מהם כל המצבים האפשרים לכמויות במיכלים?</u>

```
def index(i, j, n):
    return (m + 1) * i + j
def bottle(n, m):
    dim = (m + 1 ) * ( n + 1)
    mat = [[False] * dim for i in range(dim)]
    for i in range(n + 1):
        for j in range(m + 1):
            ind = index(i, j, m)
            mat[ind][index(0, j, m)] = True
            mat[ind][index(i, 0, m)] = True
            mat[ind][index(i, m, m)] = True
            mat[ind][index(n, j, m)] = True
            i1 = index(max(0, i + j - m), min(m, i + j), m)
            mat[ind][i1] = True
            i1 = index(min(n, i + j), max(0, j + i - n), m)
            mat[ind][i1] = True
        for j in range (dim): // diagonal - should decide what path from v to v himslef
            mat[j][j] = False
    return mat
```

סיבוכיות

- על פי שני הfor נקבל : $O\left(nm\right)$ ־ היות ובפנים יש רק פעולות קשיחות

נכונות

המצבים האפשריים מ(a,b):

1. (m,b) $\rightarrow$// למלא את A
2. (o,b) $\rightarrow$// לרוקן את A
3. (a,n) $\rightarrow$// למלא את B
4. $(a,0)$ $\rightarrow$// לרוקן את B

כמו כן, אפשר לשפוך מ A ל B

```
if (a+b <= n)
```

```
→(0,a + b)
else // a+b > n
```
$\rightarrow (a-(n-b,n))$// =(a+b-n,n)

5. כלומר $B \rightarrow A$ אנחנו מבצעים $\langle Max\left(0,a+b-n\right), MIN\left(a+b,n\right)\rangle$ או יותר פשוט: $\langle (a+b)-MIN\left(a+b,n\right), MIN\left(a+b,n\right)\rangle$

ואם נשפוך מ $A \rightarrow B$ אז:

```
if (a+b <= m)
    →(a+b,0)
else // a+b > m
```
$\rightarrow (m,b-(m-a))$// =(m,a+b-m)

6. כלומר $A \rightarrow B$ אנחנו מבצעים $\langle Min(a+b,m),(a+b)-Min(a+b,m)\rangle$

דוגמה:

ניקח $A=2$ ו $B=1$, המצבים האפשריים:

		0	1	2	3	4	5
		$(0,0)$	$(0,1)$	$(1,0)$	$(1,1)$	$(2,0)$	$(2,1)$
0	$(0,0)$	1	1	0	0	1	0
1	$(0,1)$	1	1	1	0	0	1
2	$(1,0)$	1	1	1	1	1	0
3	$(1,1)$	0	1	1	1	1	1
4	$(2,0)$	1	0	0	1	1	1
5	$(2,1)$	0	1	0	0	1	1

3 פלויד־ורשל

שימושים

- האם הגרף קשיר ־ בהנתן מטריצת שכנויות של $0,1$.
- **לכל** קודקוד מה המסלול הקצר ביותר בשבילו **לכל** קודקוד

חסר:

- אלגוריתם לספירת רכיבי קשירות ־ יותר יעיל ב $DFS \backslash BFS$
- דוגמה קומפקטית

```
FW(w[][])
    for each pair of vertices (i,j)
        dist[i][j] = ∞
    for each edge vertices(i,j)
        dist[u][v] = w(u,v) // weight of
    for each vertex v
        dist[v][v]
    for k from 0 to n-1
        for i from 0 to n-1
            for j from 0 to n-1
              dist[i][j] = dist[i][j] ||dist[i][k] && dist[k][j] // case of bool - check Connectivity
              if(dist[i][k] != ∞ && dist [k][j] != ∞)
                dist[i][j] = min (dist [i,j],dist[i][k] + dist[k][j]

Fw_Get_Pathes()
    for i from 1 to n
        for j from 1 to n
            if (dost[i][j] != ∞  path[i][j] = i→ j
            else path[i][j] = ""
    for k from 1 to n
        for i from 1 to n
            for j from 1 to n
              if (dist[i][k] != ∞ && dist[k][j])
                if(dist[i][j] > dist[i][k] + dist[k][j])
                   dist[i][j] = dist[i][k] + dist[k][j]
                   path[i]j] = path[i][k] + path [k][j]
```

נכונות

למה: יהיה $p=(v_1,v_2,.....v_k)$ המסלול הקצר ביותר בין קדוקוד v_1 ל v_k , ויהיה $p'_{ij}=(v_1,...v_j)$ תת מסלול של p , אם p הוא המסלול הקצר ביתר בין קודקוד v_1 ל v_k אז p'_{ij} הוא המסלול הקצר ביותר בין קדקודים v_i ל v_j

<u>הוכחה:</u>

- נניח בשלילה ש p'_{ij} אינו המסלול הקצר ביותר בין v_i ל v_j , לכן קיים מסלול $f"_{i,j}$ שיכול להחליף אותו.
- נפרק את המסלול p לשלושה חלקים: $p^1=(v_1,....,v_{i-1}),\ p^2=(v_i,...v_j)\ ,\ p^3=(v_{j+1},....v_k)$
- מתקיים ש: $p=p^1\cup p^2\cup p^3$
- מההנחה $|f|<|p^2|$, ולכן נחליף את המסלול p^2 ב f
- נקבל מסלול חדש $t=p^1\cup f\cup p^3$ הקצר יותר מ מ p בסתירה לכך ש p הוא המסלול הקצר ביותר

הגדרה (טווח חיפוש המסלול): יהיו $v_1,...,v_n$ קודוקודים, ומסלול $p_{i,j}$, בין v_i ל u_j אז נסמן ב k את אורך המסלול בין v_i ל u_j ונאמר $\{v_1,...,v_k\}$ כקודקודים לאורך המסלול, הם ה**קודקודים המורשים**.

דוגמאות:

- $k=0$, הקודקודים המורשים
- $k=1$ מסלול העובר דרך קוקוד מורשה אחד
- ....

נכונות FW **: בין כל קודקוד לכל קודקוד מתקבל המסלול הקצר ביותר**

נתבונן במסלול קצר ביותר $F_{i,j}$ כאשר :

- הקודקדים המורשים הם $\{v_1,...,v_{k-1}\}$,
- נסמן את המרחק ב $d_{i,j}$
- נוסיף את v_k לרשימת הקודקודים המורשים (נגדול את המסלול האפשרי) ישנן 2 אפשרויות

1. <u>הקוקדקוד v_k לא נמצא על המסלול הקצר ביותר</u> בין v_i לבין v_j $\Leftarrow$ הוספת הקודקוד החדש לא שיפרה את המרחק בין v_i,v_j $\Leftarrow$ $d^k_{i,j}=d^{k-1}_{i,j}$
2. <u>הקודקוד v_kנמצא על המסלול הקצר ביותר</u> , מהו המרחק החדש בין v_i ל v_j ?

- נחלק את המסלול החדש לשנים p_{ik},p_{kj} - תתי מסלולים של המסלול הקצר ביותר מהטענה הם גם הקצרים ביותר $\Leftarrow$ $d^k_{i,j}=d^k_{ik}+d^k_{k,j}$
- בכל אחד מתת המסלולים v_k הוא קודקוד אחרון/ראשון , ולכן לא שייך לרשימת המורשים שהם קודקודים אמצעים $\Leftarrow$ לכן העלות מושפעת לכל היותר מ $k-1$ קודקודים מורשים

$$\Leftarrow\ d^k_{i,j}=d^{k-1}_{ik}+d^{k-1}_{k,j}$$

- בשני המקרים המרחק בין v_j ל v_i מורכב ממרחקים של הקבוצה הקודמת של הקודקודים המורשים
- לכן בהינתן מטריצת שכנות המתאימה ל $k=0$ אנו יכולים לחשב את המרחקים הקצרים ביותר לכל $k=1,2,...$
- כאשר $k=n$ מקבלים מסלול אופטימלים לכל זוגות קדקודי הגרף. כאשר עוברים מ $k-1$ ל k מתקבל ערך מינמלאי שמתקבל בשני המקרים:

$$d^k_{i,j}=\min\left(d^{k-1}_{i,j},d^{k-1}_{ik}+d^{k-1}_{kj}\right)$$

3.1 משקלים על הקודקודים וצלעות

משקלים רק על הקודקודים

פסאודו־קוד

קלט:

- $f[]$ - מערך של משקלים המגודרים על קודקודי הגרף
- $adj[][]$ -מטריצה בוליאנית המגדירה את צלעות הגרף

פלט: מטריצה המייצגת את המרחקים הקצרים ביותר בין קדקודי הגרף

אלגוריתם

1. בונים מטריצה המייצגת את המשקלים על צלעות הגרף, על פי הנוסחה: $weight[a][b] = f[a] + f[b]$
2. $h[][] = Fw(weight)$
3. ממירים כל תא בh על פי הנוסחה הבאה: $d[i][j] = \frac{h[i][j]+f[i]+f[j]}{2}$

נכונות:

- למה: בסוף האלגוריתם כל תא מייצג
  - נסמן ב $h(i,j)$ את עלות המעבר בין קוקוד i לקודקוד j לפי צלעות
  - נסמן ב $d(i,j)$ עלות המעבר בין קודקוד i לקודקוד j לפי הקודקודים אז מהגדרת המחיר
  - $d(i,j) = f_1 + f_{i+1} + ... + f_{j-1} + f_j \Rightarrow 2 \times d(i,j) = 2\left(f_1 + f_{i+1} + ... + f_{j-1} + f_j\right)$
  - $h(i,j) = (f_i + f_{i+1}) + (f_{i+1} + f_{i+2}) + ... + (f_{j-1} + f_j) = f_i + 2\left(f_{i+1} + f_{i+2} + ... + f_{j-1}\right) + f_j$
  - לכן:

$$\underbrace{2 \times d(i,j) - h(i,j) = f_i + f_j}_{1} \iff \underbrace{h(i,j) = 2 \times d(i,j) - f_i + f_j}_{2} \iff \underbrace{d(i,j) = \frac{h(i,j) + f_i + f_j}{2}}_{3}$$

- לכן מנוכנות fw המטריצה h היא מטריצת המסלולים הקצרים בין כל קודקוד לכל קודקוד +
  מהלמה את ההמרה מהמטריצה h (משקלים על צלעות) למטריצה d (משקלים על הקודקודים), כנדרש.

משקלים על צלעות + קודקודים

פתרון:

- נגדיר עלות צלע: $p(a,b) = f(a) + 2w(a,b) + f(b)$
- בהנתן מסלול $h(i,j)$:

$$\begin{aligned} h(i,j) &= \overbrace{f(i) + 2w(i,i+1) + f(i+1)}^{p_1} + \overbrace{f(i+1) + 2w(i+1,i+2) + f(i+2)}^{p_2} + ... + f(j) \\ &= f(i) + 2\left(f(i+1 + w(i+1) + ...f(j-1) + w(j-1,j)\right) + f(j) \end{aligned}$$

- הפעם d :

$$
\begin{gathered}
d(i,j) = f(i) + w(i+1) + f(i+1) + ... + f(j-1) + w(j-1,j) + f(j) \\
\Downarrow \\
2d(i,j) = 2\left(f(i) + w(i+1) + f(i+1) + ... + f(j-1) + w(j-1,j) + f(j)\right) \\
= f(i) + 2\left(w(i+1) + f(i+1) + ... + f(j-1) + w(j-1,j)\right) + f(j)
\end{gathered}
$$

- מתקיים ש:

$$2d(i,j) - h(i,j) = f(i) + f(j)$$

- ולכן נמיר כל תא במטריצה

$$d(i,j) = \frac{h(i,j)+f(i)+f(j)}{2}$$

4 דיקסטרה

שימושים

- מציאת המסלול הקצר ביותר מקודקוד מקור לכל קודקודי הגרף (בגרף עם משקלים אי־שליליים)

האלגוריתם:

1. מסמנים את X הקודקוד הנוכחי, כקודקוד שביקרו בו
2. נסמן ב Y קודקוד שכן של X
   (א) אם לא ביקרנו כבר ב Y , נסמן שמרחקו שווה ל $Min\left(Y.dist, X.dist + wieght(X, Y)\right)$
3. ממשיכים כך, עד שנעבור על כל הקודקדים

פסאודו

```
G  - array of vertices, S  - source vertrex
Dijkstra(G,S)
    //init
    Q ←MinHeap()
    Adj - list of adjancy // egdes with wieght
    pred ←array() // previous vertex of vertex v
    dist ←array() // min distance from source
    visited ←array() // boolean array
    for each v ∈ G
        dist[v]= ∞
        pred[v] = NIL
        visited[v] = false
    dist[s] = 0
    Q.push(S)
    while (Q not empty)
        u = Q.pop() // get vertrex
        for each(vertex v∈ Adj(u) // get neighbour
            if (not visited[v]) // "new path found"
              if(dist[v] > dist[u] + weight(v,u)) // if better update
                dist[v] = dist[u] + weight(v,u)
                pred[v]
                Q.decreaseKey(v)
        visited[u] = true // sign this path
```

קבלת כל המסלולים:

```
getPath(G,u,v)
    Dijkstra(G,u)
    t = v
    String path = t
    while(t!=u)
```

```
        t = pred[t]
        path = t + path
    return path
```

סיבוכיות

- עוברים על כל הקודקודים, דרך כל הצלעות ולכן $O(|V|+|E|)$
- כל הוצאה מערימת מינימום $O(\log|V|)$
- לכן $O\left((|E+|V|)*\log|V|\right)$

נכונות

נגדיר:

G - גרף מכוון משוקלל, משקלי הצלעות מספרים לא שלילים

S - קודקוד מקור

$\delta\left(S,v\right)$ = המרחק המינמלי בין S לבין כל $v\in V(G)$

משפט: לאחר הרצת האל' מתקיים $dist[v]=\delta\left(S,v\right)$

הוכחה באינדוקציה - נראה שלכל קודקוד v כאשר מוצאים אותו מתור עדיפויות Q מתקיים השיוויון $dist[v]=\delta\left(S,v\right)$

<u>בסיס:</u> בשלב הראשון $dist[S]=\delta\left(S,S\right)=0$

<u>צעד:</u> נניח עבור n שלבים ונוכיח לשלב $n+1$

- יהיה u הקודקוד שנבחר בשלב ה $n+1$ צ"ל ש $dist[u]=\delta\left(s,u\right)$

  $(*)$נשים לב כי לכל $x\in V$ מתקיים ש $dist[x]\geq\delta(S,x)$
- יהי P המסלול הקצר ביותר מ s ל u ,
- נסמן ב v את הקודקוד הראשון הלא מטופל במסלול P ויהיה z קודקוד קודם ל v ב P , לכן קודקוד z טופל כבר ומהנחת האינדוקציה מתקיים $dist[z]=\delta\left(S,z\right)$
- כיון ש P הוא מסלול קצר ביותר מ s ל u התת־מסלול שלו מ s ל v הוא גם קצר ביותר לכן:

$$\delta\left(S,v\right)=\delta\left(S,z\right)+wight(v,z)=dist[z]+wight(v,z)$$

$$\Downarrow$$

$$\delta\left(S,v\right)=dist[z]+weight(v,z)$$

- <u>על פי פעולת האלגוריתם </u>נבחר את המינימלית ולכן במסלול שהתקבל לפי דייקטסרה קדקוד v קיבל מרחק מינמלי , ולכן:

$$dist[v]\leq dist[z]+weight(z,v)=\delta\left(S,v\right)$$

- כיון ש $\delta\left(S,v\right)$ הוא המרחק הקצר ביותר מ S ל v אז $dist[v]=\delta\left(S,v\right)$
- אבל קדקוד v לא מטופל ואנו בחרנו בקודקוד u ולכן:

$$dist[u]\leq dist[v]=\delta\left(S,v\right)\leq\delta\left(S,u\right)\overset{*}{\leq}dist[u]$$
$$\Downarrow$$
$$dist[u]=\delta\left(s,u\right)$$

- האלגוריתם מסתיים כאשר כל הקודקדים מטופלים לכן לכל קדקוד מתקיים $dist[u]=\delta\left(S,u\right)$, כנדרש.

5 גרפים עם משקלים שלילים - קטע מקסימלי במערך

הבעיה: מציאת הקטע המקסימלי

- חיפוש שלם -
  - מספר הקטעים הרצופים במערך $(n-1)+(n-2)+...+1=\frac{n(n-1)}{2}=O(n^2)$
  - חישוב סכום כל קטע $O(n)$
  - סה"כ: $O(n)*O(n^2)=O(n^3)$
- חמדני - לא עובד
- דינאמי - נחשב את סכומם של כל $\frac{n(n-1)}{2}$
  - על ידי: פתרון ראשון: $S[i,j]=S[i,j-1]+s[j,j]$ (לוודא מול התרגול של אור)
  - $O(n^2)$

5.1 *Best*

1. מתחילים לסכום את איברי המערך מאיבר חיובי ראשון
2. סוכמים את האיברים ושומרים על הסכום המקסימאלי עד שנקבל סכום שלילי. כלומר בכל תת־קטע $[a_i,...,a_k]$ הסכומים $a_i+a_{i+1}+..+a_j>0$ לכל $j<k$
3. מאפסים את הסכום וחוזרים ל2

```
best(int[] arr) {
    int n = arr.length; int helper[] = new int[n];
    int tmp_sum = 0, t_l = 0, t_r = 0; max_sum  = 0, m_l = 0, m_r =0 ;
    for (int i = 0; i < n; ++i) :
        tmp_sum += arr[i];
        if (tmp_sum < 0): // count to negative -> define new segment
            tmp_sum = 0;
            helper[i] = -1;
            t_l = i + 1; // next segment
        else:
            helper[i] = tmp_sum;
            if ( max_sum < helper[i]) :  // found greater sub-array
              max_sum = helper[i];
              m_l = t_l;
              m_r = i;
    System.out.println("max_sum: " + max_sum);
    System.out.println("m_l: " + m_l);
    System.out.println("m_r: " + m_r);
    return max_sum;
}
```

נכונות

טענה 1 : קטע בכל סכום מקסימאלי הוא אחד מהקטעים הנ"ל

- המספר האחרון של כל תת־מרוח הוא שלילי בגלל שלפניו הוא סכום חיובי
- המספר הראשון של כל תת־מרוח הוא חיובי אחרת הוא בעצמו היה שלילי ־ ואנחנו מדלגים עליו.
- נניח בשלילה שבחלוקה זו יש תת קטע עם סכום גדול יותר.
  - על כן יהיו $a_i, ..., a_{j-1}a_j, ...a_k,$ ובה"כ נניח כי הקטע $a_j....a_k$ הוא המקסימלי
  - נסכום מ a_j אז הסכום $a_i + ... + a_{j-1} > 0$ ולכן:

$$a_j + ... + a_k < a_i + ... + a_{j-1} + a_j + ...a_k$$

בסתירה למקסימיליות של $[a_j, ...a_k]$

  - על כן הסכום המקסימלי מתקבל מסכימת כל אברי הקטע
- כעת נניח בשלילה, שקיימת סכימה מקסמלית עם אברי מהקטע הבא
  - על כן יהיו $a_i, ..., a_{j-1}a_j, a_{j+1}...a_k,$ ובה"כ נניח כי האיבר הa_j גורם לסכום להיות קטן מאפס
  - אז באופן מיידי נקבל ש:

$$a_i + ... + a_{j-1} > a_i + ... + a_j$$

כי $a_j < 0$

מסקנה: בין כל $\frac{n^2}{n}$ כדאי להתבונן ב n קטעים בלבד.

5.2 קטע מקסימלי במערך מעגלי

הרחבות

- תת קטע עם סכום מקסימלי, הקצר/הארוך ביותר
- כמה תת קטעים יש עם סכום מקסימלי

פתרונות

- חיפוש שלם:
  - נצור את כל תת המערכים האפשריים $O(n^2)$
  - נפעיל $best$ על כל אחד מהם $O(n)$
  - סה"כ $O(n^3)$
  - בתכנון דינאמי $O(n^2)$
- שרשור המערך לעצמו + $best$
  - דוגמה נגדית: $[1, 2, -1, 5]$
  - $max = 1 + 2 + 5 = 8$
  - בפתרון $[1, 2, -1, 5]\,[1, 2, -1, 5] \Leftarrow m\_sum = 14$

- שימוש חכם ב$Best$
  - ישנן 2 אפשרויות:
    * לגרום ל $Best$ למצוא את ה min ואז $comp=\sum(A)-min$
    * להשאר את $Best$ אותו דבר , ואז: $comp=\sum(A)+Best(-A)$
  - ולהחזיר את $Max\left(Best(A),\ comp\right)$

<u>נכונות:</u>

$$max\left(Best(A)\right)=\sum(A)-\left(-Best(-A)\right)=\sum(A)+Best(-A)$$

  - דוגמה: $[10,2,-5,8,-30,12]$

5.3 בעיות תחנות הדלק

הבעיה: נתון מעגל עם מחירים על הצלעות (=ק"מ) ומחיר בקודקודים (=דלק שניתן למלא) - הרכב נמצא (עם מכל ריק) בנקודה 1 ורוצים שהוא ישלים סיבוב - דרך n הקודקודים והצלעות - האם יצליח?

- יש לבדוק בכל שלב שיהיה יותר דלק a_i= מק"מ = b_i
- נצור מערך שלישי $c_i=b_i-a_i$
- נפעיל $best$
- נתחיל מתחילת הקטע המקסימלי אותו מחזיר $best$

<u>נכונות</u>

- לוודא:

$$\begin{gathered}a_1\geq b_1\\ a_1+a_2\geq b_1+b_2\iff a_2\geq(b_1-a_1)+b_2\\ \vdots\\ a_1+a_2+...+a_n\geq b_1+b_2+..+b_n\end{gathered}$$

  - סיבוכיות (באופן דינאמי) - $O(n)$

- <u>משפט לובס: אם $\sum a_i=\sum b_i$ $\Leftarrow$ קיימת נקודה שממנה ניתן להתחיל ולסגור נקודה.</u>
  - נגדיר מערך חדש $c_i=b_i-a_i$
  - נשים לב ש: $\sum a_i=\sum b_i\Leftrightarrow\sum b_i-\sum a_i=0=\sum c_i$
  - ונניח בשלילה שישנו k כך ש $sum\left(C\left[i,j\right]\right)+sum\left(C\left[j+1,k\right]\right)<0$
  - אז מתכונת המשלים יתקיים ש:

$$\underbrace{sum\left(C\left[i,j\right]\right)+sum\left(C\left[j+1,k\right]\right)}_{<0}+\underbrace{sum\left(C\left[k+1,i-1\right]\right)}_{>0}=0$$

  - ולכן על פי הרעיון של $best$ נחבר את $[i,j]$ עם $[k+1,i-1]$$\Leftarrow$ נתחיל מנקודה k $\Leftarrow$ תמיד נהיה חיוביים (= מספיק דלק) $\Leftarrow$ נסגור את המעגל

6 בעיית המטריצה (הרחבת קטע מקסימלי במערך)

6.1 חיפוש שלם

```
private void whole_search() {
    int max = 0, tmp, t_i = 0, t_j = 0, b_i = 0, b_j = 0;
    for (int i = 0; i < n; ++i)
     for (int j = 0; j < m; ++j)
      for (int k = i; k < n; ++k)
       for (int l = j; l < m; ++l)
        tmp = sumSubMat(k, l, i, j)
          if (max < tmp) {
           max = tmp;
           t_i = i; t_j = j;
           b_i = k; b_j = l;
    return max,t_i,t_j,b_i,b_j
private int sumSubMat(int rows, int cols, int s_i, int s_j):
    int sum = 0;
    for (int i = s_i; i <= rows; ++i) :
        for (int j = s_j; j <= cols; ++j):
            sum += mat[i][j];
    return sum;
```

סיבוכיות

- אפשרויות לבנות רצועות של שורות = $\frac{m(m-1)}{2}$, אפשרויות לבנות פסים של עמודות = $\frac{n(n-1)}{2}$
- חישוב סכום תת מטריצה : $O(m \cdot n)$, סהכ האפשרויות לתת מטריצות : $\frac{n(n-1)}{1} * \frac{m(m-1)}{2} = O\left(n^2 m^2\right)$
- סה"כ : $O\left(n^3 m^3\right)$

6.2 חישוב רצועות עם $best$

```
private void strip_search():
    int max = 0, tmp, t_i = 0, t_j = 0, b_i = 0, b_j = 0;
    int[] arr = new int[m];
    for (int i = 0; i < n; ++i) { // idx row - start range
        for (int j = i; j < n; ++j) { // idx row - end range
            Arrays.fill(arr, 0);
            for (int k = i; k <= j; ++k) { // idx row start in range
              for (int l = 0; l < m; ++l) { // idx row end in range
                arr[l] += mat[k][l];
            tmp = best(arr);
            max = max < tmp ? tmp : max;
    return max
```

סיבוכיות

- יש לנו $O\left(\frac{n(n+1)}{2}\right)$ רצועות, $O(n\cdot m)$ חישוב שטח מלבן, $O(m)$ מציאת מקס'
- סה"כ: $O\left(n^2\right)\left(O\left(n\cdot m\right)O\left(m\right)\right)=O\left(n^3\cdot m\right)$ (עבור מלבן עומד)
- שיפור היעילות: $O\left(Min^3\left(n,m\right)*Max(n,m)\right)$.

6.3 הכלה והדחה

```
private void inc_exp_princple() {
    int max = 0, tmp;
    // fill helper mat by counting sub-mats from (0,0)
    int[][] helpMat = new int[n][m];
    helpMat[0][0] = mat[0][0];
    for (int j = 1; j < m; ++j) { helpMat[0][j] += helpMat[0][j - 1] + mat[0][j];}
    for (int i = 1; i < n; ++i) { helpMat[i][0] += helpMat[i - 1][0] + mat[i][0];}
    for (int i = 1; i < n; ++i) {
        for (int j = 1; j < m; ++j) {
            helpMat[i][j] = mat[i][j] + helpMat[i - 1][j] + helpMat[i][j - 1] - helpMat[i - 1][j - 1];
    //create all rectangle
    int t_i = 0, t_j = 0, b_i = 0, b_j = 0;
    for (int i = 0; i < n; ++i) :
     for (int j = 0; j < m; ++j) :
      for (int k = i; k < n; ++k) :
       for (int l = j; l < m; ++l) :
        // find max sub-mat
         tmp = sumByHelper(helpMat, i, j, k, l); //O(1)
           if (max < tmp) {
             max = tmp;
             t_i = i; t_j = j;
             b_i = k; b_j = l;
private int sumByHelper(int[][] helpMat, int s_i, int s_j, int rows, int cols)
    // case on frame
    if (s_i == 0 && s_j == 0) return helpMat[rows][cols];
    else if (s_i == 0 && s_j > 0) return (helpMat[rows][cols] - helpMat[rows][s_j - 1]);
    else if (s_i > 0 && s_j == 0) return (helpMat[rows][cols] - helpMat[s_i - 1][cols]);
    // all inner cases - by Inclusion-exclusion principle
    else { // (s_i > 0 && s_j > 0 )
        return (helpMat[rows][cols] - helpMat[s_i - 1][cols]
        - helpMat[rows][s_j - 1] + helpMat[s_i - 1][s_j - 1]);
```

סיבוכיות

- בניית $helper$ - $O(m\cdot n)$, מעבר על כל המלבנים האפשריית $O(n^2m^2)$, חישוב מלבן $O(1)\cdot 3$
- סה"כ: $O(n\cdot m)+O\left(n^2\cdot m^2\right)\cdot O(1)=O\left(n^2m^2\right)$

6.4 פתרון משופר וסופי

```
private void integrated_sol() {
    int max = 0, tmp;    int[] arr = new int[n];
    for (int i = 0; i < m; ++i) {
        Arrays.fill(arr, 0);
        for (int j = i; j < m; ++j)
            for (int k = 0; k < n; ++k) {
              arr[k] += mat[k][j]; // count vertical strips
            tmp = best(arr); // inc_exp_principle is inside best
            max = max < tmp ? tmp : max;
```

סיבוכיות

- מספר הרצועות $\times$ חישוב כל רצועה לכן : $O\left(\frac{n(n+1)}{2}\right) \cdot O\left(m\right) = O\left(\min^2(n,m), \max(n,m)\right)$

7 שרפת עצים

```
private void fireTree(adjLost) {
    int[] deg = new int[size];
    LinkedList<Integer> leafQ = new LinkedList();
    LinkedList<Integer> newLeafQ = new LinkedList();
    int count = size;  steps = 0;
    // count degrees
    for (int i = 0; i < adjList.length; ++i):
        int d = adjList[i].size();
        deg[i] = d;
        if (d == 1) { leafQ.add(i); } // keep only leaves
    while (count > 2) {
        // burn leave
        steps++;
        while (!leafQ.isEmpty()) {
            int u = leafQ.remove();
            deg[u]--;
            if (deg[u] == 0) count--; // undirected?
            // update his father - u is leaf ==> get(0) == his parent
            int p = (int) adjList[u].get(0);
            deg[p]--;
            // check if father is new leave
            if (deg[p] == 1) { newLeafQ.add(p); }
        }
        // free newLeafQ by pass the new leaves to LeafQ
        leafQ = (LinkedList<Integer>) newLeafQ.clone();
        newLeafQ.clear();
    }
    if (count == 2) {
        System.out.println("r: " +  steps+1);
        System.out.println("d: " +  (2*steps -1));
        return (leafQ.remove(),leafQ,remove());
    } else {
        System.out.println("r: " +  steps);
        System.out.println("d: " +  (2*steps));
        return leafQ.remove();
    }
}
```

סיבוכיות

- סכימת הדרגות : $2|E| = 2(|V| - 1)$

- על צלע אנחנו עוברים פעמיים, 1. כאשר קודקוד שואל את אביו אם הוא הופך לעלה. 2. כאשר שורפים עלה, לכן גם $2|E|=2(|V|-1)$
- סה"כ $4|V|=O(|V|)$

טענה 5 : יהיה T עץ ו $a\in T$ קודקוד כלשהו , ויהיה b הקדקוד הרחוק ביותר ממנו (כלומר $ex(a)=max\left\{dist(v,a),v\in V\right\}=dist(a,b)$ **) אז המסלול שבין a ל b עובר דרך מרכז העץ c**

הוכחה

טענה 6 (מציאת קוטר): יהיה $x\in T$, ויהיה y הקודקוד הרחוק ביותר ב x ,אז קוטר העץ הוא המרחק בין y ל z , כאשר z הוא הקודקוד הרחוק ביותר מ z

הוכחה:

- נניח בשלילה שקיים מסלול אחר בין a ל b כך ש $dist(a,b)>dist(y,z)$,
- ממשפט שני המסלולים עוברים דרך מרכז העץ c , ולכן:

$$dsit(y,c)+dist(c,z)=dist(y,z)<dist(a,b)=dist(a,c)+dist(b,c)$$

- קודקוד y רחוק ביותר מ x לכן $dist(b,c)\leq dist(y,c)$, אחרת הוא b היה נבחר
- באופן דומה z הרחוק ביותר מ y $dist(c,a)\leq dist(c,z)$, אחרת a היה נבחר
- לכן:

$$dsit(y,c)+dist(c,z)<dist(a,c)+dist(b,c)\leq dsit(y,c)+dist(c,z)$$
$$\Updownarrow$$
$$dsit(y,c)+dist(c,z)<dsit(y,c)+dist(c,z)$$

וזו סתירה

טענה 7 : (מציאת מרכז) : לכל עץ יש מרכז אחד או שניים

- המרחק המרבי מקודקוד v לכל v_i אחר, הוא כאשר v_i עלה
- נתבונן ב T בעל יותר משני קודקודים, ממשפט ב T יש לפחות 2 עלים
- (שרפה ראשונה) מוחקים את כל העלים ב T מתקיים ש:
- $(\#)$:
  - בגרף T_1 שהתקבל הוא שוב עץ, כי:
    - מחקנו עלים, ולכן הגרף עדיין קשיר
    - מחיקת צלעות, אינה סוגרת מעגל
  - ל T ו T_1 יש את אותו מרכז, כי:
    - מחיקה כזו מורידה **לכל הקודקודים בגרף** את מדד האקסצנטריות, ולכן המקס' נשאר אותו מקסימום, ולכן המרכז אותו מרכז
- שרפה שניה - נמחק את כל העלים מT_1 , ונקבל עץ T_2 מ $\#$, המרכז אותו מרכז
- כיון שהגרף סופי, בשלב כלשהו ניוותר עם קודקוד 1 או צלע 1 , ומכאן נובעת הטענה.

מסקנות:

1. <u>מספר השרפות:</u>

- לעץ עם מרכז אחד ־ הרדיוס שווה למספר השרפות
- לעץ עם שני מרכזים ־ הרדיוס של העץ שווה למפסר השרפות + 1

2. <u>הקוטר מקיים:</u>

$$2 \cdot r - 1 \leq diameter \leq 2 \cdot r$$

הערה: גרף שאינו עץ יכול להכיל יותר מרכזים

הוכחה ־ להשלים

7.0.1 הרחבה: האם 2 עצים איזומורפים

פסאודו־ קוד

```
isIsom(T1,T1):
    centers1 = fireTree(T1)
    centers2 = frieTree(T2)
    if (centers1 != centers2):
        return false;
    if centers1 == 1 :
        retrun helpIsom(T1.root) && helpIsom(T2.root)
    else :
        retrun helpIsom(T1.root1,T2.root1) && helpIsom(T1.root1,T1.root2)
def helpIsom(s1,s2):
    return tree_code(s1) == tree_code(s2)
def tree_code(v)
    if v is leaf:
        return "01"
    else:
        for i in range(v.childs.length)
            codes[i] = tree_code(v.childs[i])
        sort(codes)
        for code in codes
            ans += code
        ans = "0" + ans "1"
        return ans;
```

סיבוכיות:

- $O(|V|) = fireTree \times 2$

- $tree\_code$ - עובד ברקרוסיה, ועובר על כל העץ פעמיים פעם בירידה ופעם בעליה, לכן עבורים על כל קודקוד (למעט העלים) ועל כל צלע פעמיים, לכן: $O(|V|+|E|)=O(|V|)$ (כי זה עץ)

- סה"כ : $O(|V|)$

<u>נכונות:</u>

(לא מספיק פורמאלי - חסר המון טענות עזר והגדרות)

טענה: שני עצים איזומורפים אם ורק אם בכל תת עץ השם הקנוני של העצים זהה

נוכיח רק צד אחד - שמות זהים $\Leftarrow$ עצים איזו':

<u>הוכחה באינדוקציה:</u> על גובה העץ ("מלמטה")

<u>בסיס:</u>

- נניח עבור $h=0 \leftarrow v$ הוא עלה, ומהגדרת האלגוריתם כל עלה בשני העצים הוא "01" ומכאן שכל עלה איזומורפי לכל עלה

<u>צעד:</u> נניח לעצים בגובה $k<h$ ונוכיח ל h

- יהיו T_1,T_2 תתי עץ בגובה h , נסמן את שורש העץ ב s_1,s_2
- ממשפט בדיסקרטית כל אחד מהילדים שלהם הוא גם עץ, וגובהו $h-1$
- מהנחת האינדוקציה אם השמות הקנונים של העצים ב T_1זהים לשמות העצים ב T_2 העצים איזומורפים
- ע"פ פעולת האלגוריתם נמיין את הרצפים הבינארים מקטן לגדול + ונסדר את הילדים של s_1 ע"פ סדר זה
- באותו אופן נבצע עבור s_2
- נקבל שאם השמות זהות זהים $\Leftarrow$ השמות הקנונים זהים $\Leftarrow$ הילדים מסודרים באותו סדר ב T_1,T_2 $\Leftarrow$ העצים זהים $\Leftarrow$ העצים איזומורפים

ע"פ פעולת האלגוריתם נוסיף 0 בהתחלה ו 1 בסוף - מכין שהעץ סופי, בשלב כלשהו נגיע לשורשים ונקבל שכל T_1,T_2 איזומורפים זה לזה.

8 BFS

קלט: רשימת שכנויות (הגרף), קודקוד מקור

```
private void bfs(LinketList[] adjList, int s) {
    //init1
    int[] color = new int[size] , distance = new int[size] , parents = new int[size];
    LinkedList<Integer> Q = new LinkedList();
    int u;
    // init2
    for (int i = 0; i < size; ++i)
        color[i] = WHITE; distance[i] = -1; parents[i] = -1;
    }
    // init3 - take source
    color[s] = GREY; distance[s] = 0; parents[s] = 0;
    Q.add(s);
    while (!Q.isEmpty()) {
        u = Q.remove();
        // look at u neighbour's <==> down tree level
        for (int i = 0; i < adjList[u].size(); ++i) :
            int v = (int) adjList[u].get(i); // get neighbour
            if (color[v] == WHITE) : // if new - sign it + update distance + keep it
              color[v] = GREY;
              distance[v] = distance[u] + 1;
              parents[v] = u;
              Q.add(v);
        color[u] = BLACK; // forget u
    return d,p
```

סיבוכיות

- $init2$ - מעבר על כל הקודקודים לכן $O(|V|)$
- כל קודקוד נכנס ויוצא פעם אחת מהתור , ב $O(1)$
- ב$while$ אנחנו עוברים על כל הקודקודים הסמוכים לקודקוד כלשהו, לכן עוברים פעמיים על כל קודקוד ופעמיים על כל צלע (צד אחד מכל קודקוד) לכן $O(2|E|+|V|)$
- סה"כ: $O(|V|+|E|)$

נכונות

למה 1 : סריקת הגרף BFS בונה עץ כתת־גרף של G בעל הקודקודים שניתן להגיע אליהם מ s

הוכחה באינדוקציה

<u>בסיס:</u>

- בשלב הראשון הגרף מכיל קודקוד אחד, s , כאשר אנו מוספים קודקודים הם בהכרח לבנים (נפגשים איתם פעם ראשונה) לכן אנו מוסיפים גם הצלע שבעזרתה הגענו לקודקוד $\Leftarrow$ יש לנו תת ־גרף קשיר וחסר מעגלים כנדרש לעץ

<u>צעד: נניח לשלב i ונוכיח לשלב $i+1$</u>

- עד השלב ה i גילינו קודקוד לכל היותר פעם אחד, ולכן מהנחת האינדוקציה יש לו אב אחד
- נוציא קודקוד מהתור נמסמנו ב u, ונעבור על שכניו, אם כל שכניו אפורים/שחורים, אז מהנחת האינדוקציה יש לנו עץ
- אחרת מצאנו קודקוד חדש v , ונחבר אותו כבן של u , בפרט יש לנו גרף קשיר חסר מעגלים (כי התעלמנו מקודקודים אותם פגשנו) , ולכן יש לנו עץ כתת גרף של G

למה 2 (BFS עובד לפי רמות העץ): התור $Q=\{v_1,....v_r\}$ מקיים אחת משתי האפשרויות:

1. לכל $i\in[1,r]$ מתקיים ש$dist[v]=k$ עבור $k\in\mathbb{N}$
2. קיים $t\in[1,r]$ כך שלכל $i\in[1,t]$ $dist[v_i]=k$ ולכל $j\in[t+1,r]$ $dist[v_j]=k+1$

<u>בסיס:</u>

- ב$init$ מכניסים s וב $while$ מוסיפים את כל שכניו ־ כולם חדשים ולכן $dist=1$

<u>צעד:</u> נניח ל איטרציה ה i ונוכיח ל $i+1$ ־ כלומר נניח $dist$ בכל התור הוא k או $k+1$ (על פי סדר)

- נשלוף קודקוד u מהנחה $dist[u]=k$
- נבדוק את שכניו. אם השכן חדש אז נכניס אותו לתור עם $dist[u]+1=k+1$
- אחרת, v הוא קודקוד שכבר פגשנו, ולא יכנס לתור.

נכונות BFS : המרחקים של קדקודי הגרף עד המקור המתקבלים בעזרת אלגוריתם BFS הם המרחקים הקצרים ביותר

<u>הוכחה</u>

- נגדיר $\delta(u,v)$ כאורך המסלול המינימלי מ u ל v, ונאמר ש $\delta(u,v)=\infty$ אם אין מסלול בין u ל v
- נפעיל את האלגוריתם על קודקוד s , נקבל בחזרה נקבל ערך כלשהו ב $dist[u]$ כאשר $u\in V$
- נניח בשלילה שקיימת קבוצת קדקודי הגרף לא ריקה $P=\{p_1,p_2,...,p_m\}$ כך שלכל $p\in P$ מתקיים ש: $\delta\left(p,s\right)<dist[p]$ כלומר קבוצה של קודקודים P שבעבורה ה BFS לא מחשב נכון
- יהיה $u\in P$ שמרחקו עד המקור הוא $\delta\left(u,s\right)$ הוא מינימאלי כלומר $\delta(u,s)=\min\left\{\delta\left(p,s\right)\right)\}$
- ויהיה w אביו של u במסלול <u>הקצר ביותר</u> אז: $\delta(u,s)=\delta(w,s)+1 \iff \delta\left(w,s\right)<\delta\left(u,s\right)$ לכן $w\notin P$ ולכן $dist[w]=\delta(w,s)$
- יהיה v קודקוד האב של u השונה מ w על פי ה BFS אז $dist[u]=dist[v]+1$ אבל $u\in P$ לכן
  - $\delta(u,s)<dist[u]$
  - $\Leftarrow \delta(w,s)+1<dist[v]+1$
  - $\Leftarrow dist[w]+1<dist[v]+1$
  - $\Leftarrow dist[w]<dist[v]$
- במצב זה לפי למה 2 w נכנס לתור לפני v , אבל מהנחה w סמוך ל u , ולכן v לא יכול להיות אביו של u (אא"כ $v=w$) בעץ הנפרש על ידי BFS (למה 1), ולכן קבלנו סתירה.

8.1 שימושים

8.1.1 <u>בדיקה האם גרף קשיר:</u>

- נעבור על $dist$ החוזר מ BFS אם קיים קודקוד שהמרחק שלו הוא $-1/\infty$, סימן שהגרף אינו קשיר

8.1.2 <u>הדפסת המסלול הקצר ביותר בין שני קודקדים:</u>

```
print_path(s,v, parents)
    if (v == s)
        print s
    else if (parents[v] == −1
        print "no path"
        return
    else
        print_path(s,v,parents[v])
        print v
```

8.1.3 <u>חישוב מספר רכיבי קישרות, ושורשי הרכיבים</u>

```
countComp()
    int v = 0 , int count = 1
    int [] dist
    boolean partOf = true;
    Set sources;
    while (partOf)
        dist = bfs(G, v);
        sources.add(v);
        partOf = false;
        for( int  i = 0 ; i < dist.length ; ++i )
            if( dist[i] == -1)
                v = i;
                count++;
                partOf = true
                break;
    return count or sources;
```

8.1.4 <u>חישוב קוטר עץ</u>

8.1.5 בדיקה האם גרף צדדי

- נוסיף משתנה בוליאני, $is2sides = True$
- נוסיף מערך $group$, שיחלק את את הגרף לקבוצות $1,2$ כאשר 0 יהיה קודקוד שעדיין לא בוקר
- בתוך הfor נוסיף:

```
while (!Q.isEmpty()) {
    u = Q.remove();
    // look at u neighbour's <==> down tree level
    for (int i = 0; i < adjList[u].size(); ++i) :
        int v = (int) adjList[u].get(i); // get neighbour
        if group[v] == group[u]:
            is2sides = False
        if (color[v] == WHITE) : // if new - sign it + update distance + keep it
            color[v] = GREY;
            distance[v] = distance[u] + 1;
            parents[v] = u;
            group[v] = 3 - group[u]
            Q.add(v);
    color[u] = BLACK; // forget u
return d,p,is2sides, group
```

גרף $G=(V,E)$ הוא דו־צדדי $\iff$ כל המעגלים בו (גם לא פשוטים) בעלי אורך זוגי

הוכחה

כיוון ראשון - נניח G דו"צ, נוכיח שאורך המעגלים זוגי

- נתון ש G דו־צדדי, לכן ניתן לחלק את קודקודי הגרף לשתי קבוצות זרות V_1, V_2 כך שכל צלע בגרף מעילה קודקוד מ V_1 וקודקוד מ V_2
- נבחר מעגל כלשהו בגרף, נניח ש u קודקוד כלשהו במעגל, בה"כ $u \in V_1$ נטייל על המעגל, כלומר נעבור בצלע ב V_1 ל V_2
- מכיון ש G גרף דו"צ, כל צלע מעבירה אותנו לקבוצה אחרת בגרף
- לכן ע"מ להיות חזרה ב $u \in V_1$, חובה עלינו "לחזור" ל V_1 , ולכן מספר המעברים זוגי

כיון שני - נניח שאורך המעגלים זוגי נוכיח ש G דו"צ

- נניח ש G קשיר, אחרת נפעיל את ההוכחה על כל רכיב קשירות בנפרד.
- נבחר קודקוד כלשהו, $u \in V_1$ ונגדיר:
  - $V_1 = \{x \in V | \text{distance between u and x is even}\}$
  - $V_2 = \{x \in V | \text{distance between u and x is odd}\}$
- נניח בשלילה שיש בגרף צלע ששני קודקודיה ב V_1 (בה"כ) , נסמנה ב $\{x, y\}$
  - לפי הגדרת V_1 קיים מעגל בגרף שמתחיל ב u מגיע דרך מסלול באורך זוגי ל x
  - אחר כך ממשיך ל y ע"י מסלול של צלע יחידה
  - מ y חוזר ל u דרך מסלול באורך זוגי
- סה"כ קיבלנו מסלול באורך אי־זוגי
- וזוהי סתירה כי קיבלנו מעגל באורך אי זוגי בסתירה להנחה שאין מעגלים באורך אי־זוגי

```
public void dfs(int s){
    for (int i = 0; i < size; i++) {
        color[i] = WHITE;
        pred[i] = NIL;
        first[i] = 0;
        last[i] = 0;
    }
    hasCycle = false;
    startCycle = NIL;
    endCycle = NIL;
    time = 0;
    visit(s);
    }
private void visit(int u){
    color[u] = GRAY; first[u] = ++time;
    for(int v : graph[u]){
        if (!hasCycle && color[v] == GRAY && pred[u] != v){
            hasCycle = true;
            startCycle = u;
            endCycle = v;
        }
        if (color[v] == WHITE){
            color[v] = GRAY;
            pred[v] = u;
            visit(v);
        }
    }
    color[u] = BLACK;
    last[u] = ++time;
}
```

נכונות

1. dfs עובר על כל קדקוד פעם אחת בלבד

הוכחה: הפונקציה dfs_visit נקראת על קדקד V רק עם הצבע שלו לבן, והצבע משתנה מיידית

2. dfs עובר על כל צלע פעם אחת בלבד.

הוכחה: הפונקציה dfs_visit נקרא על קדקוד פעם אחת בלבד + בלולאה עוברים על השכנים של אותו קודקוד $\Longleftarrow$ עוברים על כל צלע שיוצאת מהקודקוד פעם אחת

3. <u>dfs עובר כל קודקדי הגרף (ברכיב קשירות אחד)</u>

הוכחה ־ אינדוקציה על המרחק k של קדקוד כלשהו מהמקור

<u>בסיס:</u> $k = 0$,

- האלגוריתם מתחיל מקודקוד המקור

<u>צעד:</u> נניח ל $t < k$ ונוכיח ל k

- נתבונן בקדקוד v כשלהו שמרחקו עד המקור הוא k
- בגלל שקיים מסלול מ v עד המקור, ישנו קודקוד w המחובר בצלע ל v
- לכן, מרחקו של $w = k - 1$ מקודוקד המקור, ומהנחת האינדוקציה w האלגוריתם יעבור על w בדרכו ל v

סיבוכיות

- אתחול: $O(|V|)$
- (עוברים על כל הקודקדוים) dfs_visit נקראת פעם אחת על כל קדוקד ב V
- (עוברים על כל הצלעות) ה for בפונקציה מתבצעה סה"כ $\sum_{v \in V} Adj(v) = O(|E|)$,
- סה"כ $(O(|E|) + O(|V|)$

10 אוילר

```
asssum all degrees is even > 0:
def Euler_Cycle(G):
    st = Stack()
    Cyc = []
    st.myPush(4)
    while not st.isEmpty():
        v = st.myPeek()
        if len(G[v]) == 0:  # count deg
            Cyc.append(v)
            st.myPop()
        else:
            u = G[v][0]  # first
            st.myPush(u)
            G[v].remove(u)
            G[u].remove(v) # delete 1 edge <==> deg(G) -= 2
return Cyc
```

נכונות

משפט: היה $G=(V,E)$ גרף קשיר לא מכוון. ב G יש מעגל אוילר $\iff$ כל הדרגות של הקודקודים בגרף זוגיות

<u>כיון ראשון: בG יש מעגל אוילר $\Leftarrow$ כל הדרגות זוגיות</u>

- יהי $u \in V$ קודקוד כלשהו במעגל.
- כל מעבר של המעגל דרך הקודקוד u הוא דרך שתי צלעות שונות זו מזו, ומהצלעות של המעברים האחרים.
  - לכן דרגתו של הקודקוד u זוגית.
- אם u הוא הקודקוד הראשון במעגל אז סופרים 1 ביציאה הראשונה ממנו, 2 בכל פעם שעוברים דרכו, ולבסוף כשחוזרים אליו בסיום.
- בכל מקרה קיבלנו שהדרגה של u זוגית, כנדרש.

טענת עזר:

יהי $G=(V,E)$ גרף קשיר לא מכוון שהדרגות של הקודקודים בו זוגיות גדולות מ 0 אז כל קודקוד ב G שייך למעגל כלשהו

הוכחה:

- נצא מקודקוד u ונמשיך להתקדם מקודקוד לקודקוד בלי לעבור באותה צלע פעמיים.
- מכיון שמספר הצלעות סופי, לאחר מספר שלבים סופי לא נוכל להמשיך.
- הקודקוד בו נעצור הוא u
- אחרת לקודקוד האחרון דרגה אי־זוגית מכיון שכל מעבר בו תורם 2 צלעות וכשעצרנו הוספנו צלע אחת (לכניסה), ואלו כל הצלעות, וזוהי סתירה.

כיון שני ־ כל הדרגות זוגיות $\Leftarrow$ יש מעגל אוילר

- יהי $u_1 \in V$ קודקוד כלשהו
- לפי טענת העזר הוא חלק ממעגל $c_1 = (u_1, ...u_k, u_1)$ אם המעגל עובר דרך כל הצלעות סיימנו.
- אחרת, אז c_1 לא מכסה את כל הצלעות.
- נמחק את כל הצלעות המשתתפות ב c_1 ונסמן את הגרף החדש ב G_1
- נשים לב שכל הדרגות ב G_1 גם זוגיות ־ כיון שהורדנו מספר זוגי של דרגות
- בהכרח קיים קודקוד u_i במעגל שדרגתו שונה מאפס (אחרת c_1 כיסה את כל הגרף כפי שציינו)
- כיון ש G המקורי קשיר, קיים קודקוד x לא ב c_1 שמחובר על ידי צלע לקודקוד y ב c_1 ל y יש דרגה חיובית ב c_1 כי הצלע $\{x, y\}$ לא נמצאת ב c_1
- לכן נסמן $c_2 = (u_i, v_2, ...v_k, u_i)$ מעגל המתחיל ב u_i
- כעת נגדיר מעגל שלישי: $c_3 = (u_1, ...u_i, v_2, ...v_k, u_i, ...u_k, u_1)$ (הרכבת המעגלים)
- אם קיבלנו מעגל של כל הקודקודים, סיימנו, אחרת נמשיך בתהליך עד שנכסה את כל הצלעות
- כיון שיש מספר סופי של צלעות, בשלב כלשהו יגמר התהליך, ונקבל מעגל אוילר כנדרש.

משפט: יהי $G = (V, E)$ גרף קשיר לא מכוון. ב G יש מסלול אוילר $\iff$ כל הדרגות של הקודקודים בגרף זוגיות או שיש בדיוק שני קודקודים בעלי דרגה אי־זוגית.

הוכחה:

כיוון ראשון ־ מסלול אוילר $\Leftarrow$ כל הדרגות זוגיות או שיש בדיוק 2 קודקודים בעלי דרגה אי־זוגית:

- כמו ההוכחה בבטענה קודמת ללא הדרישה על זוגיות של הקודקוד הראשון והאחרון במסלול שהן אי־זוגיות (רק כניסה/יציאה בהתאמה)

כיון שני: כל הדרגות זוגיות/יש בדיוק 2 קודקודים בעלי דרגה אי־זוגית $\Leftarrow$ מסלול אוילר

- אם כל הדרגות זוגית אז לפי משפט קודם יש מעגל אוילר, וסיימנו
- לכן נניח שיש קודקודים a, b שדרגם אי־זוגית.
- נוסיף קודקוד חדש z וצלעות $\{a, z\}, \{z, b\}$. קיבלנו גרף חדש קשיר שכל דרגותיו זוגיות
- ממשפט קודם יש בו מעגל אוילר שמתחיל ומסתיים ב a
- כעת נשמט מהמעגל את z וצלעותיו, ונקבל מסלול אוילר שמתחיל ב a ונגמר ב b , כנדרש.

10.0.1 הוספת צלעות מינימלית לאוילר

הרחבה: יהי G גרף כלשהו, מהו מספר הצלעות **המינימלי** שצריך להוסיף על מנת שיהיה בו מעגל אוילר:
(ניסיון שלי לקראת המבחן)

1. הרץ BFS שמחזיר:
   - רשימת קודקודים בעלי דרגה אי־זוגית
   - מערך $color$

2. בפונקציה $countComp$ עבור על קדקודי הגרף:

- כל עוד יש קודקודי לבנים, קרא לBFS עם קודקוד לבן
- ע"פ הערך המוחזר , הכנס לערימת מקסימום את השלישיה $(source, stack\_of\_odd, key = stack\_of\_odd.size())$

3. פונקציית $connect\_comp$:

- $count = 0$
- שלוף שני רכיבים $C1, C2$
- אם שניהם שונים מאפס

(א) שלוף קודוקד אי־זוגי מ $v = C1.stack\_of\_odd.pop()$,

(ב) שלוף קודוקד אי־זוגי מ $u = C2.stack\_of\_odd.pop()$

(ג) בצע $G[u].add(v)$

(ד) $G[v].add(u)$

(ה) $maxHeap.push(C1.source, C1.stuack \cup C2.stack, key = C1.size + C2.size - 1 - 1,$

(ו) $count + +$

- אם אחד שונה מאפס, עבור $C1$ סימטרי ל $C2$

(א) $v = C1.stack\_of\_odd.po()$

(ב) $u.C2.s$

(ג) בצע $G[u].add(v)$

(ד) $G[v].add(u)$

(ה) $maxHeap.push(C1.source, C1.stuack \cup C2.stack, key = C1.size + C2.size - 1 + 1,$

(ו) $count + +$

- שניהם 0 :

(א) $u = C2.s$ $v = C1.s$

(ב) בצע $2\times$ $G[u].add(v)$

(ג) $2\times$ $G[v].add(u)$

(ד) $maxHeap.push(C1.source, C1.\{\}, 0,$

(ה) $count+ = 2$

4. פונקציית חיבור $connect\_odd|\_v$ (לרכיב קשירות 1)

- $C = maxQ.pop()$
- כל עוד $C.stack.size() > 0$

(א) בצע $u = C.stack.pop()$

(ב) בצע $v = C.stack.pop()$

(ג) בצע $G[u].add(v)$

(ד) בצע $G[v].add(u)$

(ה) $count + +$

11 קרוסקל

11.1 "לא יהיו מעגלים"

```
def kruskal (graph):
    MST = []
    n = len(graph) , numE = 0
    edges = ascending_sort_edges_Q_by price(graph)
    while numE < n-1:
        min_e = edges.enqueue() // Node {v,e,price}
        T = MST + min_e
        has_cyc = bfs(T, min_e.v) // improve: stop bfs if v is GRAY
            if  not has_cyc:
              MST = T
              numE++
```

11.2 הפוך - "נוודא שהגרף קשיר"

```
def kruskal_oppsite(G):
    MST = G , n = len(graph)
    edges = descending_sort_edges_Q_by price(graph)
    numE = edges,size()
    while numE > n-1:
        max_e = edges.enqueue() // Node {v,e,price}
        T = MST - max_e
        num_of_comp = bfs(T, max_e.v, max_e.u) // improve: stop bfs if it met v on the road
            if  num_of_comp == 1:
              MST = T
              numE--
```

סיבכויות:

- עוברים על $O(m-(n-1))=O(m)$ צלעות
- bfs הוא $O(m+n)$, לכן: $O(m^2)$

11.3 $disjoint-set$ - שיפור ל "לא יהיו מעגלים"

<u>פסאודו:</u>

```
KRUSKAL(G):
    T = empty
    foreach v ∈ G.V:
        MAKE-SET(v)
    foreach (u, v) in G.E ordered by weight(u, v), increasing:
```

```
        if FIND-SET(u) ≠ FIND-SET(v):
            T = T ⋃ {(u, v)}
            UNION(FIND-SET(u), FIND-SET(v))
    return A
```

מימוש:

```
class MySet:
    def __init__(self, x):
        self.parent = x
        self.rank = 0
def find_set(x):
    if x.parent != x:
        x.parent = find_set(x)
    return x.parent
def union(x, y):
    x_root = find_set(x)
    y_root = find_set(y)
    if x_root == y_root:  # connected
        return
    if x_root.rank < y_root.rank:
        x_root.parent = y_root
    elif x_root.rank > y_root.rank:
        y_root.parent = x_root
    else:  # rank equals
        y_root.parent = x_root
        x_root.rank += 1
def kruskal_set(G):
    T = [] , n = len(G) , numE = 0
    for v in G:
        MySet(v)
    edges = edges_list(G)
    while len(edges) > 0 and numE < n - 1:
        min_e = get_min(edges) // Node{price,v,e}
        v = min_e.v, u = min_e.u

        if find_set(v) != find_set(v):
            T.append(min_e)
            union(v, u)
            numE += 1
    return T
```

נכונות

- יהיה G גרף, נסמן $W(G)$ כמשקל הכולל של הגרף G
- , יהי T עץ שנבנה בעזרת אלג' קרסקל, ויהיה S עץ פורש מינמלאלי נניח בשלילה שמתקיים כי $W(S) < W(T)$
- בגרף יכולים להיות מספר עצים פורשים מינימאלים, ניקח את S כך שמספר הצלעות המשותפות שלו עם T הוא מקסימאלי ונסמן את מספר הצלעות ב k
- נניח כי $T = \{e_1, e_2, ...e_n\}$ היא סדרת הצלעות המתקבלות על ידי קרסקל (ממוינת עולה חלש)
- יהי $e_i = (a,b)$ עבור $i \in [i,n]$, הצלע הראשונה המקיימת $e_i \notin S$ ($1,....,i-1$ כן שייכות..)
- נוסיף אותה לעץ S , נקבל גרף חדש $S_1 = S \cup e_i$, בעל n צלעות $\Leftarrow$ S_1 מכיל מעגל נסמנו ב C
- היות ו T עץ, במעגל C ישנה e_p המקיימת $e_p \notin T$,
- נשווה את משקלי הצלעות e_i ו e_p
  - נזכר כי $e_1, ...e_{i-1} \in S$ $\Leftarrow$ e_p לא סוגרת איתן מעגל,
  - מפעולת האלגוריתם בשלב ה i בחרנו את הצלע המינימלית ולכן בחרנו ב e_i
  - ולכן

$$weight(e_i) \leq weight(e_p) \iff weight(e_i) - weight(e_p) \leq 0$$

- נסיר את צלע e_p מגרף S_1 נקבל עץ: $S_2 = S_1 \backslash \{e_p\} = S \cup \{e_i\} \backslash \{e_p\}$
- העץ S_2 התקבל מ S , ויש בו $k+1$ צלעות משותפות עם T , נשים לב כי :

$$W(S_2) = W(S) + \underbrace{wight(e_i) - weight(e_p)} \leq S(S)$$

- מהנחה S עץ פורש מינמאלי ולכן המשקל שלו קטן ביותר בגרף G $\Leftarrow$ $W(S_2) = W(S)$
- והרי גם S_2 עץ פורש מינמאלי $\Leftarrow$ סתירה.

סיבוכיות:

- פתרון נאיבי:$O\left(|E|\log_2|V| + |V|^2\right)$
- פתרון משופר ע"י $disjoint-set$ + איחוד חכם על פי הדרגות =$O(|E|\log_2|V| + |V|\log_2||V|) = O(|E|\log_2|V|)$
- עם מערך הצלעות ממוין$O(|V|\log_2|V|)$

12 $Prim$

פסאודו - קוד

```
def Prim(tree, root):
    T = dict()
    n = len(tree)
    numE = 0
    visited = dict(), key = dict(), parent = dict()
    for v in tree:
        visited[v] = False
        key[v] = ∞
        parent[v] = None
    key[root] = 0
    Q = min_heap(tree, key)
    while len(Q) != 0 and numE < n - 1:
        u = extract_min(Q)
        for v in tree[u1]:
            if visited[v.neb] == False and v.key < key[v_neb]:
                key[v.neb] = v.price
                    parent[v.neb] = u1
                    decrese_key(Q, v_price, v_neb)
        visited[u.name] = True
        x = get_min(Q)
        T[numE] = (parent[x.name], x.name)
        numE += 1
    return T
```

סיבוכיות

- הלולאה תרוץ תעבור על כל צלעות הגרף, ולכן $O(|E|)$
- מלבד כמה פעולות של עדכונים ב $O(1)$, ישנה הוצאה/עדכון של הצלע המינמלית ב $O(\log|V|)$ היות והערמה וזו ערמת מינימום של הקודקדים.
- סה"כ: $O\left(|E|\log|V|\right)$

נכונות

טענה: בכל שלב בו אנו מוסיפים צלע חדשה ל T , אנו מקבלים תת עץ של עץ פורש מינמאלי כלשהו M

הוכחה באינדוקציה

בסיס:

בשלב הראשון אנו מוצאים מ Q - ערמת מינמום את ה $root$ שהוא שייך לכל עץ פורש מינמאלי

צעד: על האיטרצות

- נניח שקיבלנו T תת עץ פורש של M

- אנו מוסיפים את הצלע e לT , אם $e \in M$ אז $T \cup \{e=(a,b)\} \subseteq M$ וסיימנו
- אחרת, אז $e \notin M \Leftarrow$ שב $T \cup \{e\}$ יש מעגל, היות ו T כבר עץ פורש.
- מהגדרה רק קודקוד אחד של e שייך ל T, בה"כ נניח שזה a
- היות ויש מעגל, ישנו מסלול כלשהו בין a ל b,
- תהיה הצלע $g=(x,y)$ צלע שמחברת בין רכיבי הקשירות ובכך יוצרת את המסלול בין a ל b
- האלגוריתם של פרים היה יכול להוסיף g ל T , אבל הוא בחר ב e ומכאן ש $w(e) \leq w(g)$.
- נבנה עץ חדש $M' = M \cup \{e\} \setminus \{g\}$ שמשקלו קטן או שווה למשקל של M
- אבל M הוא בעל משקל מינימאלי לכן $W(M) = W(M')$ ו M' הוא גם עץ פורש מינימאלי שמכיל את T
- לכן $T \cup \{e\} \subseteq M$, מש"ל

ואדים המלצות

- עם הגרף קטן - עדיף מחיקות
- עם הגרף גדול - קרוסקל/פרים
- עם הגרף עצום - פרים עם ערמה בינארית פיבונאצי

13 הופמן

פסאודו־קוד

```
Huffman(C)
    n = |C|
    Q ←c  // minHeap
    for i =1 to n -1 {
        create node z
        x = Q.extractMin()
        y = Q.extractMin()
        // update father
        z.left = x ; z.right = y
        z.freq = x.freq + y.freq
        Q.insert(z)
        // update sons
        x.parent = z ; y. parent =z
    }
return Q.extractMin()
```

סיבוכיות

- 2 הוצאות + הכנסה + עדכונים = $O(\log n)) = 3 \cdot O(\log n) + O(1)$
- עבור $n-1$ קודקודים = $O(n)$
- סה"כ : $O(n \log n)$

נכונות

טענה 1 : אם קידוד הוא $prefix\ free \Leftarrow uniquely\ decodable$

- נניח בשלילה שישנו קידוד דו־משמעי, נסמן:

$$x_1, x_2, ...x_k, ...x_r$$

- בה"כ נוכל לסמן:

$$\underbrace{x_1, x_2,}_{a} \underbrace{.....x_k, ...x_r}_{b} \qquad \overbrace{x_1, x_2, ...}^{f} \overbrace{x_k, ...x_r}^{g}$$

- כעת נשווה את הקידוד ל a ל f נקבל סתירה לכך ש $prefix\,free$

טענה 2: אם בהנתן קידוד נבנה עץ בו העלים הם מייצגים אות $\Leftrightarrow$ הקידוד $pfrefix\ free$

$\Rightarrow$:

- נניח בשלילה שבהנתן קידוק $prefix - free$ בנינו עץ בו קיים קודקוד המייצג אותו שאינו עלה נסמנו ב u
- נטייל בגרף מהשורש עד אותו צומת u, ונאסוף את הקידוד בגרף, בה"כ $x_1, x_2...x_r$ ומייצג את האות a

- כעת נמשיך בטיול ונגיע לעלה בה"כ תהיה זו האות b ועל פי הטיול שלנו נקבל ש: $b = x_1, ..., x_r, ...x_s$
- בסתירה לכך ש $prefix - free$

$\Leftarrow$:

- סימטרי

<u>טענה 3 : $hufmman$ אופטימלי</u>

רעיון (לא הוכחה)

- מכיון שאנחנו מתחילים לבנות את העץ "מלמטה" כלומר דואגים שאותיות בתדירות נמוכה יקבלו קידוד הכי ארוך , נקבל שאותיות בדירוג גבוה, יהיו קרובים לשורש, ולכן הקידוד שלהם יהיה קצר. ונקבל שהימוש בזכרון אופטימלי

13.1 שיפור - האפמן עם שני תורים

קלט: מערך ממוין של תדירויות מהקטן לגדול

```
def huff_code_2q(abc_freq):
    // init
    Queue q1,q2
    q1  <-- abc_freq  // by oncresing order (smellest in front)
    while q1.size + q2.size > 1:
        x = get_min(q1, q2) // handle case of one queue empty
        y = get_min(q1, q2)
        z = Node(x.name + y.name, x.freq + y.freq)
        z.left = x , z.right = y
        q2.enqueue(z)
        x.parent = z , y.parent = z
    return get_min(q1,q2)
```

סיבוכיות:

- במקרה הגרוע כל הערכים שווים $\Leftarrow$ הלולאה תתבצע ..... $+ \frac{n}{8} + \frac{n}{4} + \frac{n}{2} \Leftarrow O(n)$

14 אוסף בעיות מתמטיות

הגדרה ־ זמן ריצה NP :

נניח ויש לנו בעיה בגודל k , אם קיים אלגוריתם בוליאני הפותר אותה בזמן פולינמיאלי נאמר שהוא $''Non - Deterministic -$ $Polynomialtime''$

<u>שאלת מליון הדולר:</u> האם קיים לבעיה פתרון פולינמיאלי המתאים לכל k (מעין חיפוש שלם)

14.1 בעיית החתול והכלב

ישנו מעגל שבמרכזו עומד חתול, ועל ההיקף עומד כלב

- לכלב אסור להכנס למעגל
- הכלב רוצה לתפוס את החתול על ההיקף
- מהירות החתול היא v ומהירות הכלב היא $4v$
  - הרעיון : היקף מעגל הוא $2\pi r$, לכן כאשר החתול הולך r , הכלב רץ חצי מעגל πr ומתקיים ש: $4r > \pi r$ ולכן הכלב תופס

כיצד החתול יצליח לברוח ־ "חמדן חכם"

אלגוריתם

- החתול לוקח $\frac{1}{4}r - \varepsilon$ מהמרכז , ורץ במעגל (מעגל קטן)
- ברגע שהוא מגיע לנקודה הנגדית ביחס לכלב, מתחיל לצאת החוצה

הסבר, מתקיים: מהירות $\times$ זמן = דרך

$$\frac{\frac{3r}{4}}{V_c} = \frac{3r}{4 \cdot V_c} < \frac{\pi r}{4 \cdot V_c} \iff 3 < \pi$$

14.2 בעיית שוקולד

צריך לפרק את השוקולד <u>ליחידות</u> עם עלות מינמלית:

1	2	3	4	5	6	7	8

עלות חלוקה ל $k, n-k$ היא $k(n-k)$

ניסיונות:

- חלוקה לחצאים $4*4+2*2+2*2+4*1*1 = 16+8+4 = 28$
- חלוקה לבודדים משמאל: $1*7+1*6+1*5.....1*1 = 28$
- פיבונאצי: $3*5+1*2+1+2*3+1+2*1+1 = 28$

משפט: עלות חלוקה ליחדות של n יחידות: $1+2+...(n-1) = \frac{n(n-1)}{2}$

הוכחה ־ באינדוקציה:

<u>בסיס: $n=2$</u> אז: $1*1 = 1 = \frac{2(2-1)}{2}$

<u>צעד:</u> נניח ל n נוכיח ל $n+1$ כלומר שהעלות שווה ל $\frac{(n+1)n}{2}$

- נבצע את החלוקה ל n ול 1 לכן מהגדרת העלות + הנחת האינדקוציה: $\frac{n(n+1)}{2} = ... = n*1+\frac{n(n-1)}{2}$, כנדרש.

14.3 דוור סיני

- הבעיה: גרף ממושקל (לא בהכרח גרף מלא), צריך לצור מעגל אוילר בצורה אופטימלית
- הרעיון: להוסיף לקודקודים עם דרגות אי זוגיות, צלעות (מקבילות) מינמליות ליצירת מעגל אוילר בעלות מינמלית

<u>אלגוריתם (סקיצה)</u>

1. לוקחים את כל הדרגות האי זוגיות
2. מכינים ממנו גרף שלם (המטריצה) - ממשפט זה יהיה מספר זוגי של קודקודים ולכן אפשר לבצע שידוך.
3. מבצעים $matching$ = ההתאמה הכי זולה (לא בהכרח של צלע אחת) בין כל זוג קודקודים
   - (את ה $matching$ נייצג במטריצה)
4. מוסיפים את הצלעות לגרף המקורי

- קיבלנו שכל הדרגות זוגיות $\Leftrightarrow$ יש מעגל אוילר.

<u>סיבוכיות:</u> $NP < O\left(n^3\right)$

14.4 בעיית העוגה

הבעיה: ישנה עוגה, ושני ילדים. על העוגה יש כל מיני תוספות (שוקולוד, קצפת וכו'), צריך לחלק את העוגה לשניים כך ששני הילדים יהיו מרוצים

פרומלית:ישנה קבוצה A , ופונקציות:

- $\mu_1 : 2^A \to [0,1]$ (2^A תתי הקבוצות של A) , $\mu_2 : 2^A \to [0,1]$, כך ש : $\mu_1(A) = \mu_2(A) = 1$

  צריך שיתקיים: $\begin{Bmatrix} \mu_1(A_1) \geq \frac{1}{2} \\ \mu_2(A_2) \geq \frac{1}{2} \end{Bmatrix}$ עבור $A_1 \cap A_2 = \phi$,

<u>הנחה:</u> הפונקציות אדדטיביות $\mu_1(A_1 \cup A_2) = \mu_2(A_1) + \mu_2(A_2)$ (אין מוצרים שמקיימים הגדול שלם מסך חלקים - דוגמת המפתח והמנעול)

<u>פתרון:</u>

- הראשון מחלק את העוגה לשני חצאים, כלומר $\mu_1(A_1) = \mu_2(A_2)$
- השני בוחר את החצי המועדף אליו, כלומר $Max\left(\mu_1(A_2), \mu_2(A_2)\right) \geq \frac{1}{2}$
- השני בוחר את החצי הנותר, ולכן $\mu_1(A_?) = \frac{1}{2}$

<u>הכללה</u> - חלוקת העוגה לשלוש

- $\mu_1, \mu_2\mu_3 : 2^A \to [0,1]$

$$\mu_1(A_1) \geq \frac{1}{3} \qquad \mu_2(A_2) \geq \frac{1}{3} \qquad \mu_3(A_3) \geq \frac{1}{3}$$
$$\bigcup A_i = A \qquad A_i \cap A_j = \phi$$

- הראשון מחלק : $A = A_1 \cup A_2$, השני בוחר, על פי $\mu_2(A_2)$, והראשון בוחר את שנותר על פי $\mu_1(A_1)$, השלישי מבקש $\frac{1}{3}$ מכל אחד .
- נסמן את החלוקה $A_1 = A_{11} \cup A_{13}$ מתקיים ש:

- הראשון מרגיש: $\mu_1(A_{11}) \geq \frac{2}{3}\mu_1(A_1) \geq \frac{2}{3}\cdot\frac{1}{2} = \frac{1}{3}$

- נסמן את החלוקה$A_2 = A_{22} \cup A_{23}$, מתקיים ש:
  - השני מרגיש:$\mu_2(A_2) \geq \frac{2}{3}\mu_2(A_2) \geq \frac{2}{3}\cdot\frac{1}{2} = \frac{1}{3}$
- השלישי מרגיש, (מהאדדטיביות):= $\mu_3(A_{13}\cup A_{23}) = \mu_3(A_{13}) + \mu_3(A_{23}) \geq \frac{1}{3}\mu_3(A_1) + \frac{1}{3}\mu_3(A_2) = \frac{1}{3}\mu_3(A_1\cup A_2) = \frac{1}{3}\mu_3(A) = \frac{1}{3}$

<u>הכללה k - הוכחה באינדוקציה:</u>

- $\mu_1(A) = \mu_2(A) = ... = A(\mu_3) = 1$

$$\forall i\in[1,k] \quad \mu_i(A_i) \geq \frac{1}{k}$$
$$\bigcup_{i=1}^{k} A_i = A \quad A_i\cap A_j = \phi$$

- נניח ל$t<k$ ונוכיח ל k
- מהנחת האינדוקציה נבצע את החלוקה בין $k-1$ הילדים, ונתבונן במשא ומתן של הילד ה k
  - הראשון מרגיש: $\mu_1(A_{11}) \geq \frac{k-1}{k}\mu_1(A_1) \geq \frac{k-1}{k}\cdot\frac{1}{k} = \frac{1}{k}$
  - הילד ה k הולך לשאר הילדים, ולוקח:
    * $\mu_k(A_{1,k}) \geq \frac{1}{k}\mu_k(A_1)$
    * $\mu_k(A_{2,k}) \geq \frac{1}{k}\mu_k(A_2)$
    * ...
    * $\mu_k(A_{k-1,k}) \geq \frac{1}{k}\mu_k(A_{k-1})$

    סה"כ:

$$\mu_k(A_{1,k}\cup A_{2,k}\cup ....A_{k-1,k}) \geq \frac{1}{k}\mu_k(A_1\cup A_2\cup ....A_k) = \frac{1}{k}\mu_k(A) = \frac{1}{k}$$

<u>הערות</u>

1. <u>המרה למדעי המחשב:</u>
   - נניח ש $A=\{a_1,....,a_n\}$ ע"פ ההסתכלות של <u>הראשון</u>, ונניח ש $B=\{b_1,b_2,...b_n\}$ ההסתכלות של <u>השני</u>
   - צריך להתקיים $\sum_{i=1}^{n} a_i \cdot \frac{1}{2} \geq \sum_{i\in n} a_i$, באופן דומה ל B .
2. <u>סיבוכיות:</u>
   - עבור k אנחנו יודעים להגיד מהי החלוקה ב $kn=O(n)$ (k קבוע)
   - מהי החלוקה ? לא ידועה וזה זמן ריצה NP ולא בP