

□

# אלגוריתמים קומבינטוריים | 104291 | הרצאות

6 ביולי 2022

# 1 שיעור ראשון

## 1.1 הקדמה מינימלית

1. לא מתוכנן בוון אמצע
2. ניתן לפנות במיל ובהפסכות בכל עניין - מיל : [chaime@technion.ac.il](mailto:chaime@technion.ac.il)
3. שעת קבלה: יום שני 30 : 17 – 30 אמדו 901
4. קורס של מדעי המחשב לסטודנטים למתמטיקה
5. הקורס יהיה במתכונת של שנים קודמות - חיים מתכוון להיזכר לחומר המוצג במודול
6. דרישת קדם: קומבינטוריקה (נושאים מבבנה נתונים יהיו ניקרים בקורס זה - בעיות מيون וכו', למרות שלא קדם)
7. שיעורי בית: או כל שבוע או כל שבועיים (עוד לא הוחלט)
8. הקלטות יعلו למודול, השיעורים יהיו היברידים

## 1.2 מيون

הגדרה: תמורה  $S_n$  היא פונקציה חד-עומדת  $[n] \rightarrow [n]$  :  $\pi$  כאשר  $\{n, \dots, n\} \rightarrow [n]$

מיון הוא "פונקציה" שמקיימת את הבא:

קלט: סדרה \ רשימה של מספרים  $a = (a_1, a_2, \dots, a_n)$

פלט: רשימה ממוקנת של אותם מספרים  $a' = (a'_1, a'_2, \dots, a'_n)$  כאשר  $a'_1 < a'_2 < \dots < a'_n$  (כמובן ש  $a = a'$  כקבוצות)

הערה: אנחנו מניחים שהמספרים שונים זה מזה (הנחה של הקורס, במטרה להקל).

דוגמה: עבור הקלט הבא:  $(9, 10, 18, 19, 29, 35)$  נרצה להחזיר את הפלט הממויין  $(19, 35, 9, 10, 29, 18)$

ניסוח שקול: צריך למצוא תמורה  $\pi \in S_n$  כך ש-  $a_{\pi(1)} < a_{\pi(2)} < \dots < a_{\pi(n)}$

התשובה: תהיה  $\pi = 3, 4, 6, 1, 5, 2$  (כאשר המספרים הם האינדקסים של המספרים בסדרה המקורית - נשים לב שנוצרן עבור פעם אחת על כל המספרים כדי למספר,  $n$  צעדים)

הערה: לא משנה אם ממיינים מספרים או אובייקטים, אנחנו נסתכל על איברים של קבוצה סדורה (כל 2 איברים בקבוצה ניתנים להשוואה ע"י היחס)

## 1.2.1 מיון הכנסה

נסתכל על המספרים הבאים : 19, 35, 9, 10, 29, 18  
בכל פעם נקלוט איבר אחד ונכניס אותו למקום המתאים ע"י השוואת איבר איבר משמאלי, עד שנגיע ל 2 מספרים שהקלט נמצא ביניהם (ע"י היחס).  
לאחר שמצאנו מקום כזה, וקיים מספרים שהוכנסו קודם לכן אבל גדולים מהקלט, נזיז אותם למקום אחד ימינה - לדוגמה :

```
19
19 35
9 19 35
9 10 19 35
9 10 18 19 29
9 10 18 19 29 35
```

הערה: בקורס נוכל להציג אלגוריתמים בצורה pseudo code, ואף לעיתים נרשום אותם בשיעור ב כדי שאלגוריתם יהיה ברור יותר.

האלגוריתם שלנו ב pseudo code

```
1 def Insertion_Sort(List list):
2     for i=2 in list.length: #every time we look in first i nodes in list
3         k=list[i]
4         j=i-1
5         while j>=1 and k< list[j]: #every time we move the new node to the
6             right place in the underlist we look at
7             list[j+1]=a[j]
8             j=j-1
9         list[j+1]=k
10    return list
```

למה האלגוריתם נכון?

לעתים האלגוריתם ברור, אבל לעיתים נctrן להוכיח זו.  
באינדוקציה על  $i$ .

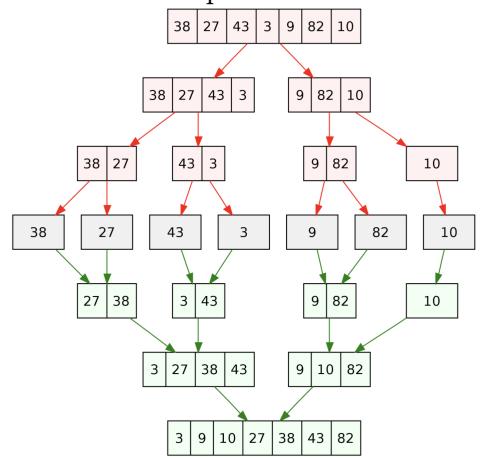
אחרי שלב  $i$ :  $a_1 < a_2 < \dots < a_i$

בשלב  $i + 1$ : האלגוריתם מזיז את  $a_{i+1}$  למקום הנכון, כך ש-

הערה: בהמשך נראה הוכחות מסוימות יותר כאשר האלגוריתם יהיה קצר יותר מסובך

## מיזוג מ有序 1.2.2

הפרד ומשול - "divide and conquer"



האלגוריתם שלנו ב pseudo code :

```

13 def merge_sort(List a):
14     n = a.size
15     if n<=1: return a
16     b = merge_sort(a[1],...,a[n/2])
17     c = merge_sort(a[n/2],...,a[n])
18     return merge (b,c)
  
```

באנדרוקציה על  $n$ , רשימות וממינות וביחד כוללות בדיקות את  $(a_1, a_2, \dots, a_n)$   
בהתה של  $merge$  נכוונה גם הכל ממויין ומכליל ביחד בדיקות את  $(a_1, a_2, \dots, a_n)$

pseudo code to "merge"

```

1 def merge (List a,List b):
2     mergedlist = createListInSize(a.size+b.size)#make empty list size a+b
3     i,j=1
4     for m=1 to a.size+b.size:
5         if j<=a.size and (i>b.size or a[j]<b[i]):|
6             mergedlist[m] = a[j]
7             j++
8         else:
9             mergedlist[m]=b[i]
10            i++
11     return mergelist
  
```

הערה: נשים לב שרוב הקומפיילרים שבודקים if מרובה תנאים, מקיימים את הסדר הבא:

1. אם  $x$  לא נכון, לא בודקים את  $y$   $x \text{ and } y$

2. נכון  $x$ , לא בודקים את  $y$   $x \text{ or } y$

### 1.3 סיבוכיות של אלגוריתם מיוון

הגדירה: סיבוכיות של אלגוריתם מיוון  $A$  היא פונקציה  $\mathbb{N} \rightarrow \mathbb{N}$  כasher  $C_A(n)$  הוא מספר השוואות  $a[i] <? a[j]$  שמבצע את האלגוריתם  $A$  על קלט באורך  $n$ .  
המירבי השוואות גם אם הפונקציה מקבלת יותר מפרמטר אחד, פשוט יתבצעו יותר השוואות.

#### למה סופרים רק השוואות?

- כל אלגוריתם מיוון חייב לבצע השוואות
- כל אלגוריתם מיוון סביר, השוואות יהיו דומיננטיות, לכן מספר הפעולות יהיה בערך מספר השוואות.
- נראה בהמשך שהזהה יפשט את הניתוה. ואפשר לקבל תוצאה די מדויקת
- לעיתים השוואת היא הפעולה הכח' יקרה

#### סיבוכיות של המוניים שראינו:

1. מיוון הכנסה - נשים לב שבכל פעם אנחנו עושים  $1 - i$  השוואות

$$C_{IS}(n) \leq \sum_{i=2}^n (i-1) = 1 + 2 + 3 \dots + (n-1) = \binom{n}{2} = \frac{n(n-1)}{2} = O(n^2)$$

נשים לב: שאם  $a = (n, n-1, \dots, 2, 1)$  או בשלב ה- $i$  מתבצעות בדיקת השוואות, ולכן  $C_{IS}(n) \geq \binom{n}{2} = \Omega(n^2)$   
(לפעמים אנחנו רוצים לוודא שהחסם זהה באמת הדוק)

-  $k, l \geq 1$  עבור  $merge$  .2

$$C_{merge}(k, l) \leq k + l - 1 = O(k + l)$$

הערה: נחסיר 1 מכיוון שאנחנו משווים בין כל האיברים חוץ מהאיבר האחרון.  
טענה: אם זוג האיברים הכי גדולים או יתבצעו בדיקת  $k + l$  השוואות

-  $merge sort$  .3

$$C_{MS}(n) \leq C_{MS}(\lfloor \frac{n}{2} \rfloor) + C_{MS}(\lceil \frac{n}{2} \rceil) +_{merge} \lfloor \frac{n}{2} \rfloor + \lceil \frac{n}{2} \rceil - 1 = C_{MS}(\lfloor \frac{n}{2} \rfloor) + C_{MS}(\lceil \frac{n}{2} \rceil) + n - 1$$

נוכחה עבורה:  $n = 2^k$

טענה:  $C_{MS}(n) \leq n \log_2 n$

הוכחה: באינדוקציה על  $C_{MS}(1) = 0$

$$C_{MS}(n) = C_{MS}(2^k) \leq C_{MS}(2^{k-1}) \cdot 2 + 2^k - 1 \leq_{assumption} 2^{k-1}(k-1) \cdot 2 + 2^k - 1 = k \cdot 2^k - 1 \leq k \cdot 2^k = n \log_2 n$$

תרגיל: אם  $n < m$  או  $C_{MS}(n) < C_{MS}(m)$   
מסקנה מהתרגיל: לכל  $n$ ,  $C_{MS}(n) \leq C_{MS}(2^{\lceil \log_2 n \rceil})$ , כלומר שמדובר בחזקה של 2 אנחנו יכולים להשתמש בטענה לעיל.  
 $C_{MS}(n) \leq C_{MS}(2^{\lceil \log_2 n \rceil}) \leq 2^{\lceil \log_2 n \rceil} \cdot \lceil \log_2 n \rceil \leq 2n(\log_2 n + 1) = O(n \log n) \Leftarrow$

תרגיל: האם אפשר להוכיח באינדוקציה שלכל  $n$

$$C_{MS}(n) \leq ? (\lceil \log n \rceil - 1)n + 1 \leq ? n \log n$$

## עצים 1.4

דוגמיה:  $a = (a_1, a_2)$  (רשימה באורך 2)

$$\begin{array}{ccc} a_1 : a_2 & & \\ < \cdot \cdot & \cdot \cdot > & \\ (a_1, a_2) & & (a_2, a_1) \end{array}$$

דוגמיה:  $a = (a_1, a_2, a_3)$  (רשימה באורך 3)

להשלים....

$$n \text{ים לבש } C_{IS}(3) = \binom{3}{2} = 3^3$$

דוגמה: עץ שמתאר לאלגוריתם לא נכון

$$\begin{array}{ccc} a_1 : a_2 & & \\ < \cdot \cdot & \cdot \cdot > & \\ (a_1, a_2) & & (a_1, a_2) \end{array}$$

דוגמה: עץ שמתאר אלגוריתם בלתי יעיל - (השוואות שאין בהן צורך וכו')

הגדרה: עץ בינארי הוא :

- גוף (אוסף של קודקודים וצלעות - שאלת זוגות של קודקודים)
- גוף שהוא עץ (קשר וחסר מעגלים)
- ומקיים:
  - קיים שורש - קודקוד מיוחד
  - הצלעות מכונות הרחק מהשורש
  - דרגת יציאה של כל קודקוד היא לכל היותר 2

טרמינולוגיה:

- עלה - קודקוד ללא בניים
  - גובה העץ -  $= \text{מספר הצלעות המירבי במסילה מכוונת מהשורש לקודקוד כלשהו}$
- הגדרה:  $T$  הוא עץ השוואות למין  $n$  איברים אם:

1. קודקודים פנימיים מתיוגים ע"י  $a_i : a_j$
2. עליים מתיוגים ע"י תמורות  $a_{\pi(1)} < a_{\pi(2)} < \dots < a_{\pi(n)}$
3. כל וקטור  $(a_1, \dots, a_n)$  מוביל למין שע"י המסילה:

$$V_0 = \text{root}$$

$$V_{n+1} = \begin{cases} \text{right}(V_n) & a_i > a_j \\ \text{left}(V_n) & a_i < a_j \end{cases}$$

עד שנגיע לעלה...

אבחנה\טענה: לכל אלגוריתם מין ניתן לבנות עץ

אבחנה: בהינתן עץ  $T$  למין  $n$  איברים שמתאים לאלגוריתם  $A$ ,

$$C_A(n) = \text{height}(T)$$

טענה:

$$\text{leaves}(T) \leq 2^{\text{height}(T)}$$

נשים לב: גובה השורש הוא 0  
טענה: בעץ  $T$  למינן  $n$  איברים,

$$n! \leq \text{leaves}(T)$$

$$C_A(n) = \text{height}(T) \geq \log_2 \text{leaves}(T) \geq \log_2(n!) = \Omega(n \log n)$$

טענה: לכל אלגוריתם מיון  $A$  ומספר  $n$ , קיים עץ השוואות  $T$  כך שלכל קלט  $(a_1, \dots, a_n)$  המסלול המתkeletal ב- $T$  עושה את אותן השוואות כמו  $A$  ומהזיר את אותו הפלט.

רעיון: נעבור על כל הקלטים האפשריים ו"נגלה" מסלולים בעץ (בעזרת  $A$ ). אם נותרו קודקודים שלא הגיעו אליהם, נזהיר באופן שדרורתי  $(a_1, \dots, a_n)$ .

טענה: יהיו  $T$  עץ השוואות למילון  $n$  איברים המתאים לאלגוריתם  $A$ , אז  $\{$  מס' הצלעות המרבי במשילה מהשורש לעלה  $\}$  (מהגדירה)

הוכחה: נסתכל על ההגדירות:  $=C_A(n)$  המקסימום על קלט  $a$  (מס' השוואות של  $A$  עשו על  $a$ ) ו-  $=height(T)$  המקסימום על המסלולים (אורך המסלול), وكل להשכנע שהם שוים.

טענה: לכל עץ בינארי  $T$ , מתקיים  $leaves(T) \leq 2^{height(T)}$

הוכחה: באינדוקציה על  $height(T)$

$$\bullet \text{ בסיס: } n = 0 \text{ עץ רק עם שורש, אז } leaves = 1 = 2^0$$

$$\bullet \text{ מקרה בסיס: } n = 1 \text{ (עד גובה 1) - זה כולל את הבסיס אבל האופציות: רק בן ימני, רק בן שמאלי, שורש ו2 בניים}$$

$$leaves \leq 2 = 2^1 \Leftarrow$$

$$\bullet \text{ הוכחה א': עבר עץ } T, \text{ נסמן תחת עץ שמאלי } T_L \text{ וימני } T_R$$

$$\text{if } height(T) \neq 0 \text{ so } height(T) = max(height(T_L), height(T_R)) + 1$$

$$leaves(T) = leaves(T_L) + leaves(T_R) \leq_{indoc..} 2^{height(T_L)} + 2^{height(T_R)} \leq 2^{height(T)-1} + 2^{height(T)-1} = 2^{height(T)}$$

טענה: בעץ  $T$  השוואות למילון  $n$  איברים  $\leftarrow$  יש לפחות  $n$  עלים.

הוכחה: לכל אחד מהקלטים  $(\pi(1), \pi(2), \dots, \pi(n))$  ( $\pi \in S_n$ ) עבר  $T$  (לדוגמה  $\pi = 3, 1, 2, 3$  או  $1, 3, 2$  או  $1, 2, 3$  וכו') יש פلت שונה,

$$\text{ולכן לכל אחד מהם מתאים עלה שומר, אז: } leaves(T) \geq n!$$

מסקנה: לכל אלגוריתם מיון  $A$ , מתקיים  $C_A(n) \geq log_2(n!)$

הוכחה:  $T$  עץ המתאים לאלגוריתם  $A$

$$C_A(n) = \text{height}(T) \geq \log_2(\text{leaves}(T)) \geq \log_2(n!)$$

מסקנה: חסם תחתון  $C_A(n) = \Omega(n \log(n))$

$$C_{\text{sort}}(n) = \min_A C_A(n) = \Theta(n \log n)$$

## 2.1 דיוון על החסם התחתון

ראינו -  $C_{MS}(n) \leq n \log_2(n)$

בתראיל -  $C_{\text{sort}}(n) \geq \log_2(n!) \geq n \log(n) - O(n)$

כל לראות -  $\log_2(n!) \leq \log_2(1 \cdot 2 \cdot 3 \cdots \cdot n) \leq \log_2(n^n) = n \log_2(n)$

$$\log_2(n!) \geq \log_2(1 \cdot 2 \cdots \cdot n) \geq \log_2(\lfloor \frac{n}{2} \rfloor \cdots \cdot n) \geq \log(\frac{n}{2})^{\frac{n}{2}} = \frac{n}{2} \log(\frac{n}{2}) = \frac{n}{2} \log(n) - \frac{n}{2} \log(2) = \frac{1}{2} n \log(n) - O(n)$$

### 2.1.1 קירוב סטרלינג

$\frac{f(n)}{g(n)} \xrightarrow[n \rightarrow \infty]{} 1$  אם  $f(n) \sim g(n)$ .  $n! \sim \sqrt{2\pi n} \left(\frac{n}{e}\right)^n$   
לפי סדר חישיבות  $n^n \sim \sqrt{n} \sim e^n \sim e^{n \log(n)}$  קבוע  $\sqrt{2\pi}$ .

כעת ניקח את המוחברים לפי סדר החישיבות:

$$\log_2(n!) = n \log_2(n) - n \log_2(e) + \frac{1}{2} \log_2(n) + \frac{1}{2} \log_2(2\pi) + o(1)$$

$$= \log_e 2 \approx 0.7$$

הערה: ההגדרה של החסם הוא **ספציפי** לאלגוריתמים מבוססים השוואות - בעזרת הדיווק זהה יכולנו למצוא את החסם התחתון

## 2.2 בעית המיזוג

קלט: 2 רשימות ממוגנות  $(a_1, \dots, a_k)$  ו  $(b_1, \dots, b_l)$  (בנחתה שכל האיברים שונים זה מזה)

פלט: המיזוג של האיחוד שלhn

$$C_{\text{merge}}(k, l) = k + l - 1$$

טענה: לכל אלגוריתם מיזוג  $A$ ,  $C_A \geq \lceil \log_2 \binom{k+l}{k} \rceil$

הוכחה: יהי  $T$  עץ השוואות למיזוג המתאים ל- $A$ .

אבחנה: צריכים להיות לפחות  $\binom{k+l}{k}$  עליים, כי כל מילה ב- $a^k b^{l-k}$  עם  $a$ -דים ו- $b$ -דים מתאימה לעלה.

נזכור על אותו טיעון כמו קודם עם הגובה של העץ ונקל ש-

## 2.2.1 פירוט מקרים

1. במקרה  $k$  כלשהו, ר-1 = 1

מיזוג הוא "חיפוש" קלט:  $i \in \{0, \dots, k\}$   $a_i < b < a_{i+1}$  ו- $b$ , פלט: נמצא  $i$  כך ש-  $a_1 < a_2 < \dots < a_k$  כאשר  $a_i < b < a_{i+1}$

חסם האינפורמציה:  $C_A(k) \geq \lceil \log_2(k+1) \rceil$

חסם חיפוש ביןארי:  $C_{BS}(k) = \lceil \log_2(k+1) \rceil$  (סיבוכיות השוואות)

תרגיל: כתבו אלגו' חיפוש ביןארי, והראו שהוא עושה בדיקות לכל  $k$  ולכל קלט.

2. במקרה ש-1 : k = 1

$C_{merge}(k, k) = 2k - 1 \Leftarrow b_1 < b_2 < \dots < b_k$ ,  $a_1 < a_2 < \dots < a_k$

חסם האינפורמציה: לכל  $A$  מתקיים

$$C_A(k) \geq \lceil \log_2 \binom{2k}{k} \rceil \geq \log_2 \binom{2k}{k} = \log(2k)! - 2\log(k!)$$

$$= 2k\log(2k) - 2k\log_2 e + \frac{1}{2}\log 2k + O(1) - 2[k\log_2 k - k\log_2 e + \frac{1}{2}\log k + O(1)]$$

$$= 2k - \frac{1}{2}\log_2 k \pm O(1)$$

טענה: לכל אלגוריתם מיזוג  $A$  מתקיים  $C_A(k, k) \geq 2k - 1$

הוכחה (דוגמה להוכחת חסם תחתון): יהי  $T$  עץ המתאים ל- $A$ , נסתכל על עלה מסוים שיראה בצורה הבא :  $L := (a_1, b_1, a_2, b_2, \dots, a_k, b_k)$

טענה (בתוך ההוכחה) כל השוואות הבאות מופיעות במסילה המובילה לעלה  $L$  :  $a_k < b_k, \dots, b_1 < a_2, a_1 < b_1$  (סה"כ  $2k-1$  השוואות)

נניח בשילולו שלא מופיעה ההשוואה  $a_2 < b_2$  ( נגד שם נחליף בין 2 איברים, בה"כ  $(a_2, b_2)$  ונתבונן בקלט, ונראה שלא קיבל רשימה ממויינת, ולכן זו סתירה). נקבל שכמות ההשואות היא בוודאות לכל הפחות  $2k-1$ .

## 2.3.1 תוכנות

1. מחזיקה  $n$  איברים הניתנים להשוואה

2. הכנסת איבר ב- $O(\log n)$  השוואות

3. למצוא את המינימלי  $O(1)$  זמן , אפס השוואות

4. להוציא איבר ממקום נתון ב- $O(\log n)$  השוואות

5. בניית ב- $O(n)$  השוואות (בניגוד ל- $O(n \log n)$  שנובע מהן "ל")

6. עירימה נייצג בעזרת מערך באורך  $n$  - אבל נדרש עלייה כע"ז ביןאי.

הערה: 2,3,4 נתונים לנו מין ב-

הגדרה: עץ ביןари מושלם אם כל עלייו באותו מרחק מהשורש  $\leftarrow 2^k - 1$  קודקודים.

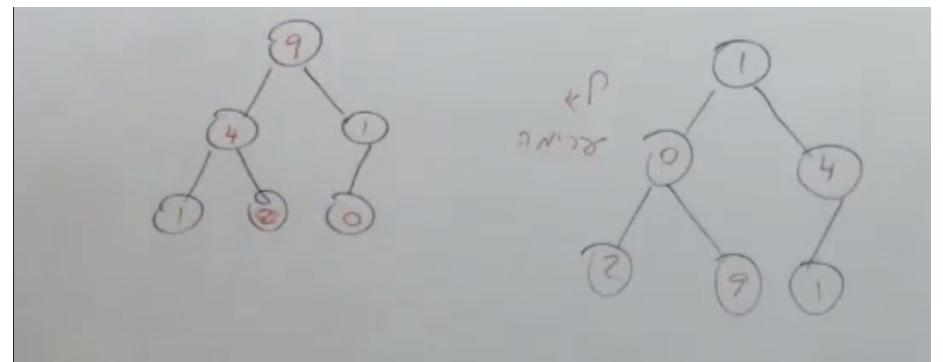
נוסחאות לקבלת צמתים :  $parent(v) = \lfloor \frac{v}{2} \rfloor$  ,  $right-son(v) = 2v + 1$  ,  $left-son(v) = 2v$

הגדרה: עץ הוא מאוזן בעלי  $n$  עליים, אם הוא מושRNA מהעץ המושלם עם  $\lceil \log_2 n \rceil$  על העלים  $\{1, \dots, n\}$

הגדרה: מערך  $A$  הוא ערמה אם לכל קודקוד

$A[v] \geq (A[left(v)], A[right(v)])$  (נקבע \ שקול)

דוגמה:



## 2.3.2 תחזוקת עירימה

- נניח מישחו שינה את  $A[v]$

- נניח מישחו הוציא את  $A[v]$  ורשם במקומו את  $A[n]$  וקטין את  $n$  ב-1

- נניח מישחו רשם משחו ב- $A[1 + n]$  והגדיל את  $n$  ב-1

↳ יש מקום אחד שմפר את תוכנת העירימה.

### החלפות - *heapify* 2.3.3

מקרה א' - אלגוריתם *heapify up* - *v* גדול מידי:

```
20 def heapify_up (Heap h,Node v):
21     while v>1 and v.parent>v:
22         swap (h[v],h[v.parent])
23         v=parent(v) #update the pointer who need to heap up
24
```

מקרה א' - אלגוריתם *H[v]* - *heapify up* - *v* קטן מידי:

```
25 def heapify_down (Heap h,Node v):
26     while left(v)<=n:
27         if right(v)<= h.size and (h[right(v)]>h[left(v)]):
28             u=right(v)
29         else:
30             if h(u)>h(v):
31                 swap(h(u),h(v))
32                 v=u
33             else return
```

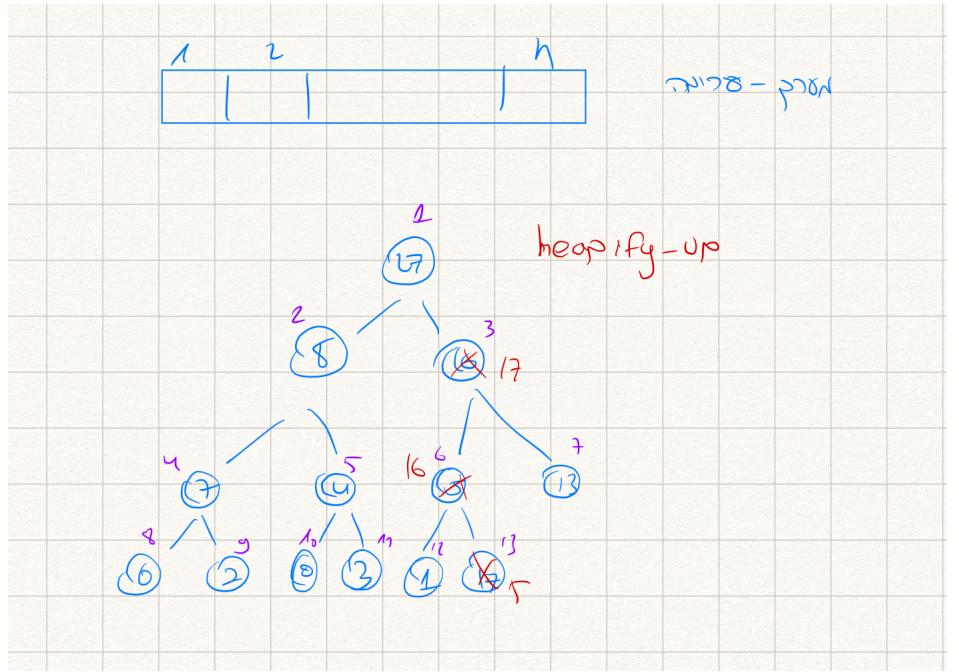
טענה: מתכונת את הערימה אם *H[v]* קטן \ גדול מדי.

## 3 הרצאה 3

תזכורת ערימה זה מערך (בגודל  $n$ ) ובעצם נסתכל עליו כעץ בינארי

- האלגוריתם  $\text{heapify\_up}(H[1, \dots, n], v) \leq \text{lon}_2 v$  שמטרתו:

- סיבוכיות -  $\text{Cheapify\_up}(n, v) \leq \text{lon}_2 v$  -



- האלגוריתם  $\text{heapify\_down}(H[1, \dots, n], v) \leq \log_2 n$  שמתארתו:

– סיבוכיות  $\text{heapify\_down}(n, v) \leq 2[\log_2(n) - \log_2(v)] \leq 2 \cdot \text{height}(v)$   
תוכורת  $\text{height}(v)$  מס הצלעות המירבי בمسילה מכונה מ- $V$  לעלה.

### 3.1 איך השתמש בערימה למין

נתן מערך מבולגן  $H[1, \dots, n]$  (לא ממויין ואפילו לא ערימה)  
מטרה: למין את  $H$ .  
 אנחנו יודעים שבעורת merge sort ניתן למין ב- $n \log_2 n$  מטרת ביניינים: לסדר את  $H$  בערימה.

1. תמיד אפשר למין  $O(\log_2 n)$

2. עבור כל  $v$  מ- $2$  עד  $n$  אפשר לעשות  $\text{heapify\_up}(H[1, \dots, n], v)$  ולבסוף לקבל ערימה.

(א) נוכנות: כי באינדוקציה, אחרי  $v$ ,  $H[1, \dots, v]$  ערימה.

(ב) סיבוכיות :  $O(n \log_2(n))$

(ג) עצירה: כי  $\text{heapify\_up}$  עצוץ

3. נróż על לולאה  $[1, \dots, n] = [n, \dots, 1]$  (הפוך) ונבצע בכל פעם  $\text{heapify\_down}(H[1, \dots, n], v)$  (הפוך) ונקבע בכל פעם  $v = n$  (הפוך).

(א) נוכנות: כי באינדוקציה (בכל פעם נבודק את  $v$  שנוסיף, ורק הוא זה שיכول לפגוע בסדר, שהוא לא יהיה גדול מבניו.) אחרי השלב הד- $v$  לכל  $v > u$ : תחת העץ יש  $u$  השורש שלו הוא ערימה.

**(ב) סיבוכיות:**  $2 \sum_{v=1}^n height(v) \leq O(n \log_2(n))$  וגם  $2n \geq 2 \sum_{v=1}^n height(v)$  נוכיח באינדוקציה  
**טענה:**  $Cstart\_heap \leq 2n$  מספיק להראות בסיס האינדוקציה  $1 = n$  קל לראות שעובי באינדוקציה על  $n$

$$2 \sum_{v=1}^n height_n(v) = 2 \sum_{v=1}^{\lfloor \frac{n}{2} \rfloor} (height_{\lfloor \frac{n}{2} \rfloor}(v) + 1) = 2 \left[ \sum_{v=1}^{\lfloor \frac{n}{2} \rfloor} (height_{\lfloor \frac{n}{2} \rfloor}(v)) \right] + 2 \cdot \lfloor \frac{n}{2} \rfloor \leq 2 \cdot \lfloor \frac{n}{2} \rfloor + 2 \cdot \lfloor \frac{n}{2} \rfloor \leq 2n$$

**טענה עזרה:** כאשר מורידים את כל העליים, הגובה של כל קודקוד אחר קטן ב-1 (נשים לב שהגובה נקבע ע"י המרחק המירבי של קודקוד מעלה)

**אבחנה:** מה הגובה הממוצע (בערך) של קודקוד בעץ בינארי מאוזן?  
 נחשב את הגבהים, כך שיש חצי עליים שהגובה שלהם הוא 0, ואו רביע שהגובה שלהם הוא 1 ושמינית שהגובה שלהם הוא ... 2

$$\frac{1}{2} \cdot 0 + \frac{1}{4} \cdot 1 + \frac{1}{8} \cdot 2 + \frac{1}{16} \cdot 3 \dots \leq \sum \frac{h}{2^{h+1}} < 1$$

אנחנו משתמשים כमובן על עץ מושלם.

## 3.2 מיזן בעוררת עריםה

pseudo code to "heap sort"

```
def Heap_sort (Heap h):
    start_heap(h)
    for v in (n,n-1,...,2):
        swap (h[v],h[1])
        heapify_down(h[1],...,v-1),h[1])
```

**נכונות:** באינדוקציה, אחרי השלב ה-

.1.  $H[v, \dots, n]$  ממויינים וגם גדולים מהשאר

**סיבוכיות:**  $2n + O(n \log_2 n)$

## 3.3 בעיית הבחירה Select

**קלט:** רשימה לא ממויינת  $A[1], \dots, A[n]$  ומספר  $k \in \{1, \dots, n\}$

**פלט:** איבר  $A[i]$  שהוא ה- $k$  ברשימה הממוינת עם אותו איברים, במיללים אחרות  $A[i]$  כך ש-

**דוגמה לקלט לפונקציית Select (נניח שהמספרים שונים):**

1.  $k = 1$  למצוא את המינימום

2.  $k = n$  למצוא את המקסימום

3.  $\lceil \frac{n}{2} \rceil$  למצוא את החציון (יכול להיות מצב של 2 איברים כאשר מספר האיברים הוא זוגי, ניקח העליון מבין השתיים)

**טענה:** חסם תחתון  $c_{min}(n) \geq n - 1$   
לכל  $i \neq j$  חייב להופיע בז' השוואת -  $A[i] < A[j]$  כאשר  $\{<, >\} \in \{A[k], A[j]\}$  (שוו השוואה שאומרת ש- $j$  גדול ממיشه)

כיוון אחר (דומה):

- לטעון שיש בכל שלב רשות מועמדים למינימום ובכל השוואה הרשימה קטנה ב-1 לכל היותר.
- טורניר (לפרט)
  - לעיתים נתקל בגרף לא קשור, נצורך לטפל בו.
- משחק - יש 2 שחנים, אחד שואל שאלות והשני עונה לו תשובות.
  - השחקן הראשון באחד שלביים, אומר ש- $a_7$  הוא המינימום, אז השחקן השני ייתן רשותה של איברים.... (לא ברור)

פתרון לבעה

1. נמצא את האיבר  $a_k$  בגודלו ע"י  $k$  מעברים על המערך  $\leq n \cdot k$
2. נמצא את האיבר  $a_k$  בגודלו ע"י מיוון וגיישה למקום  $\leq n \log_2 n$ ,  $k$
3. נמצא את האיבר  $a_k$  בגודלו ע"י שליפה  $k$  איברים מערימה  $\leq 2n + k \log_2 n$
4. נמצא את האיבר  $a_k$  בגודלו ע"י אלגוריתם בזמן לינארי (נראה)

pseudo code to "simplistic select"  
 $A$  מערך,  $k$  הוא המיקום (בגודל) של האיבר שנרצה להחזיר (נניח שהאיברים במערך שונים אחד מהשני)=  
guess\_median פונקציית ניחוש תיאורטית

```
def simplistic_select (A[1, ..., n], k) :  
    m(index, value) ← guess_median(A)  
    L ← {A[i]|A[i] < m}  
    R ← {A[i]|A[i] > m} (alternative is for loop size n)  
    if |L| = k - 1 : return m;  
    if |L| > k - 1 : return simplistic_select (L, k);  
    if |L| < k - 1 : return simplistic_select (R, k - |L| - 1);
```

טענה: האלגוריתם נכון גם אם  $\text{guess\_median} \leq$  לא.

הוכחה: באינדוקציה על  $n$ ...

טענה: אם הניחושים נכונים הסיבוכיות  $\leq 2n$

טענה: אם הניחושים לא נכונים, יכול להיות  $\binom{n}{2} = O(n^2)$  המקרה הכיו גורע זה שהוא תמיד ייכן את השמאלי  $L$ .

טענה: אם מוחשים איבר אקראי או ממוצע  $(O(n))$

טענה: נגיד שהניחושים תמיד קולעים בשני הרביעונים המרכזיים (נניח שналץ את המערך ל 4 תתי מערכים)

מאמר Blum,Floyd,practt...

1. נחלק את  $A$  ל-  $\lfloor \frac{n}{5} \rfloor$  חמישיות

2. נמצא את החזיון של כל חמישיה ללא רקורסיה

3. הניחוש הוא החזיון (בעזרת רקורסיה) של רשימה  $\lfloor \frac{n}{5} \rfloor$  החזיונים

pseudo code to "select"

$A$  מערך,  $k$  הוא המיקום (בגודלו) של האיבר שנרצה להציג (נניח שהאיברים במערך שונים אחד מהשני)

```

def Select(Array A,k):
    n=A.size
    if n <=5:m=A[1]
    else:
        F=(A[1,...,5],A[6,...,10],...,A[...,\lfloor n\5\rfloor*5])
        M=(Select(F[1],3),Select(F[2],3),...,Select(F[\lfloor n\5\rfloor],3))
        m=Select (M,\lfloor n\10\rfloor)
        L={A[i] | A[i]<m}
        R={A[i] | A[i]>m}
        if |L|=k-1:return m
        if |L|>k-1:return Select(L,k)
        if |L|<k-1:return Select(R,k-|L|-1)
    
```

נשים לב  $C_{\text{select}}$  ש-

סיבוכיות:

ראשית נספר השוואות:

• מציאת  $M$ :  $M \cdot \lfloor \frac{n}{5} \rfloor \cdot 10$

• מציאת  $m$ :  $m \in C_{\text{select}}(\lfloor \frac{n}{5} \rfloor)$

• בחישוב  $R, L$ :  $n - 1$

• קריאה הרקורסיבית בסוף:  $(\text{חישוביים}: \dots \leq C(\max(|L|, |R|)))$

לסיכום:

$$C < C(\lfloor \frac{7n}{10} \rfloor + 4) + (\lfloor \frac{n}{5} \rfloor) + 3n - 1$$

כאשר נסuum על החישובים לפי עז :

$$n + (\frac{7}{10} + \frac{1}{5})n + (\frac{7}{10} + \frac{1}{5})^2 n + \dots = \frac{1}{1 - (\frac{7}{10} + \frac{1}{5})} n$$

## 4 הרצאה 4

### 4.1 המשך Select

נוסחה רקורסיבית: לכל  $n > 5$

$$C(n) \leq C\left(\frac{1}{5}(n - n\%5)\right) + maxC(t) + 2(n - n\%5) + n - 1$$

כאשר  $t < \frac{9}{10}(n - n\%5) + (n\%5)$   
להשלים....

### 4.2 קידוד

נסתכל על טבלה

A	00
B	01
C	10
D	11

ונרשום את המילה  $ABDACADABDA \leftarrow 0001110010001100011100$ , שזה **11 אותיות** שנן **22** ביטים.  
נסתכל על מקרה גדול יותר ,  $n$  אותיות שנן  $2n$  ביטים, האם ניתן לחסוך בביטים? באינטואציה לא, אבל נראה שכן.

A	0
B	111
C	110
D	10

ובמקרה הכללי , עבור  $n$  אותיות נצטרך  $\frac{5}{11}n \cdot 1 + \frac{2}{11} \cdot 3 + \frac{1}{11}n \cdot 3 + \frac{2}{11}n \cdot 2 = \frac{20}{11}n \approx 1.82n$  הראינו והוא שבחרנו עבר אותיות עם שכיחות יותר גדולה, קידוד יותר קצר וכן הלאה.

הגדרה: קידוד הוא אוסף מילים ב- $\{0, 1\}^*$  -  $\{w_1, w_2, \dots, w_m\}$

הגדרה: קידוד נקרא **קידוד רישא** אם אין 2 מילים שונות  $w_i, w_j$  כך ש- $w_i$  רישא של  $w_j$

הגדרה: לקידוד יש פענוח ייחיד אם לכל מילה  $w$  כאשר מתקיים  $k = l$  וגם  $w = w_{i_1}, w_{i_2}, \dots, w_{i_k} = w_{j_1}, w_{j_2}, \dots, w_{j_l}$  וכן הלאה...  $w_{i_1} = w_{j_1}$

טענה: קידוד רישא ניתן לפענוח ייחיד

הוכחה: נניח שלא  $\leftarrow$  או יש מילה  $w$  עם שני פענוחים שונים.

נניח את  $j_t \neq i_t$  הראשון שבו הם שונים:

אם  $|w_{j_t}| = |w_{i_t}|$  אז בהכרח  $w_{i_t} = w_{j_t}$  כי קוראים ב- $w$  מאותו מקום, סתירה !

אם  $|w_{j_t}| \neq |w_{i_t}|$  אז בהכרח אחד רישא של השני, סתירה !

הערה: לכל קידוד שנייה לפענוח יחיד קיים בקידוד רישא "טוב" כמו זה.

#### 4.2.1 אלגוריתם פענוח לקידוד רישא

טענה: לכל קידוד רישא ניתן להתאים עץ בינהר שעליו הם -  $\{1, \dots, n\}$  אותיות ולהיפך

תרגיל: מה ההתאמה ולמה היא חח"ע ועל.

lolala של האלגוריתם: כל פעם קוראים ביט, כל עוד יש ונווז בעץ ימינה או שמאלה.

אם הגיענו לעלה נרשום את האות שמופיעה בו, ונקפוץ חורה לשורש.

שגיאות: נגדיר שגיאות:

- אם הגיענו לסוף הקלט לא בעלה

- אם אין בן שמתאים לביט

הגדרה: עץ מלא אם כל קודקוד הוא או עלה או עם שני בניים.

תרגיל: לכל קוד רישא  $(w)$  יש קוד רישא  $(\hat{w})$  עם עץ מלא, שטוב לפחות כמו זה.

"טוב" = אם מתקיים  $|\hat{w}_i| \leq |w_i|$

סימון: נסמן ב- $f(c)$  את השכיחות של תו  $c$  כלומר, מספר הפעמים ש- $c$  מופיע בקלט.

$$B_{bits}(T, f) = \sum_{c \in \Sigma} f(c) \cdot |w_c| = \sum_{c \in \Sigma} f_{freq}(c) \cdot d_{depth}(T, c)$$

הבעיה: בהינתן  $f$  למצוא קוד רישא  $T$  כך ש- $B(T, f)$  מינימלי.

וריאציה:

נסמן ב- $p(c)$  את הסתברות של תו  $c$  להופיע

length  $L(T, p) = \sum_{c \in \Sigma} p(c) \cdot |w_c| = \sum_{c \in \Sigma} p(c) \cdot d(T, c) = E[d(T, c)]$  ונגדיר

טענה: יהיו  $T$  עץ המתאים לקידוד רישא ו- $T'$  עץ המתkeletal ע"י החלפת אותיות  $x, y$  (עליהם).

או מתקיימים :

$$L(T, p) - L(T', p) = \sum_{c \in \Sigma} p(c) \cdot d(T, c) - \sum_{c \in \Sigma} p(c) \cdot d(T', c) =$$

$$= p(x) \cdot d(T, x) + p(y) \cdot d(T, y) - p(x) \cdot d(T', x) - p(y) \cdot d(T', y) = [p(x) - p(y)] \cdot [d(T, x) - d(T, y)]$$

**נשים לב ש-**  $d(T', y) = d(T, x)$  ו-  $d(T', x) = d(T, y)$

**מסקנה בטענה:** אם  $p(x) < p(y)$  ו-  $d(T, x) \leq d(T, y)$

**5.1 אלגוריתם חמדניים**

דוגמה: בחירת אינטראולים זרים נתונים  $I \ni i \neq j \in I$  ( $a_i, b_i), (a_j, b_j) \in \{1, \dots, n\}$ . **צריך למצוא** **כך שלכל**  $I \subset \{1, \dots, n\}$  **כך ש**  $|I|$  מקסימלי **לדוגמה:**

1. הרצאה 9:00-11:00

2. הרצאה 10:30-13:30

3. הרצאה 12:00-14:00

4. תרגול 13:00-17:00

5. טיול 8:00-17:00

6. ספורט 16:00-18:00

7. סרט 16:00-19:00

8. מסיבה 18:00 - 21:00

או :

• פתרון לא אופטימלי :  $I = \{1, 3, 8\}$ • פתרון אופטימלי :  $I = \{1, 3, 6, 8\}$ • לא פתרון :  $\{7, 8\}$ **5.2 איך נמצא פתרון אופטימלי?**

טענה: אם  $(a_i, b_i)$  הוא בעל  $b_i$  מינימלי או  $i$  שיך לפתרון אופטימלי (בדוגמה זה השעה שנגמר האירוע)

הוכחה: יהי  $I$  פתרון אופטימלי .

אם  $i \in I$  או סיימנו.

אחרת, יהי  $I \ni j \in I$  כך  $b_j$  מינימלי (עבור  $I$ ). נחליפה:  $I' = I \setminus \{j\} \cup \{i\}$  (להחסיר את  $j$  ולהוסיף את  $i$ )  $\Leftarrow$  מתקיים  $|I'| = |I|$ .

נשיב לב שגם  $I'$  פתרון .

יהי  $(a_i, b_i) \cap (a_k, b_k) = \emptyset$  וגם  $b_i \leq_{known} b_j \leq_{i,k different} b_k < b_k$  אז  $k \in I' \setminus \{i\}$

## 5.2.1 הצעה לאלגוריתם

אולי נמיין בהתחלה ? ← נבחר את זה שמתאים ראשון ← נמתק את כל מי שנחתק אליו ← נמשיך ברקורסיה על הרשימה  
שנשארה

← נותר להראות שהפתרון אופטימלי של הרשימה שנשארה ביחיד עם הראשון הוא הפתרון האופטימלי של הרשימה  
המקורית (הראנו ע"י הטענה למטה)

טענה: אם  $i$  הוא כך ש-  $b_i$  מינימלי, ו-  $J$  פתרון אופטימלי של הבעיה עם האינדקסים  $\{j : (a_i, b_i) \cup (a_j, b_j) \neq \emptyset\}$   
או  $\{i\} \cup J$  פתרון אופטימלי של הבעיה המקורית עם אינדקסים  $\{1, \dots, n\} \setminus \{i\}$

הוכחה: יהי  $I'$  פתרון אופטימלי.  
יהי  $\{i\} \cup I = I' \setminus \{j\}$  (להחסיר את  $j$  ולהוסיף את  $i$ ) גם אופטימלי לפי הטענה הקודמת.  
כעת  $\{i\} \cup I$  פתרון אופטימלי של הבעיה  $K$

$$|J \cup \{i\}| - 1 = |J| \geq_{j \text{ optimal}} |I \setminus \{i\}| = |I| - 1 = |I'| - 1$$

ולכן  $|I'| \geq |J \cup \{i\}| \geq |I|$  ולכן גם  $\{i\} \cup J$  אופטימלי

## 5.3 האלגוריתם של max disjoint intervals

```
pseudo code to "selectmax disjoint intervals"
def Max_Disjoint_Interval((a[1],b[1]),..., (a[n],b[n])):
    sort by b[i]
    I={i}
    i=1
    for j in (2,...,n):
        if a[j] >= b[i]:
            I=I united with {j}
            i=j
    return I
```

סיבוכיות:  $O(n \log n)$  בغالל המيون בהתחלה

## 5.4 המשך קידוד - אופטימליות

תוכנות הגדרות וטענות:

- קוד הוא אוסף של מילימ'ם ביןאריות  $*\{0, 1\}$  ו-  $w_i \in \{0, 1\}$  ו-

- קוד הוא קוד רישא אם אין  $w_i \neq w_j$  כך ש-  $w_i$  רישא של  $w_j$ .

- קוד רישא ניתן לפענוח יחיד

- אלגוריתם קידוד קל - פענווח של קוד רישא בעורת עץ

טענה: יש התאמה חח"ע ועל בין קוד רישא בגודל  $m$  לעצים בינהרים עם  $m$  עליים.  
הערנו ללא הוכחה:

- מספיק לדון בקוד רישא (אם רוצים פענווח יחיד)
- מספיק לדון בקוד עם עץ מלא (לכל קודקוד פנימי יש שני ילדים)

#### 5.4.1 בעיית הקידוד

נתונים אוסף תווים  $C = \{1, \dots, m\}$  וקטור שכיחיות\סתברויות  $(p(1), \dots, p(m))$ .  
 מצא קוד רישא עץ  $T$  כך ש-  $L(T, P) = \sum_{c \in C} p(c) \cdot d(c, T)$  כאשר  $L$  הוא “אורך המילה המומוצעת”

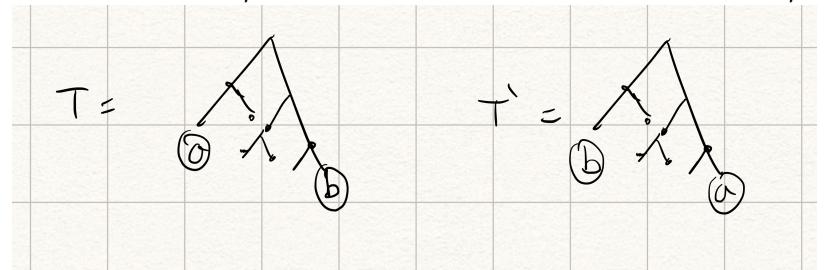
הגדרה:  $T$  אופטימלי אם  $L(T, p)$  שווה למינימום (אין  $L(T', p)$  אחר שקטן ממנו)

חישוב(חזרה): אם  $T'$  הוא העץ שמתקבל מ- $T$  ע”י החלפת  $a, b$  אז :

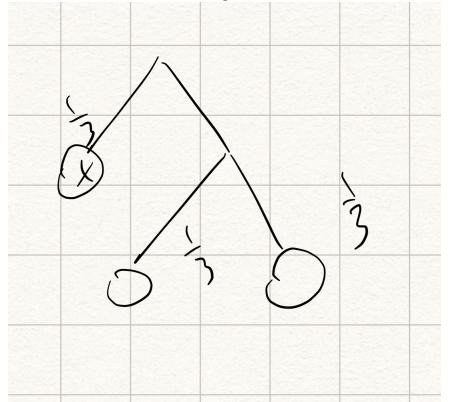
$$L(T, p) - L(T', p) = \sum_{c \in \Sigma} p(c) \cdot d(T, c) - \sum_{c \in \Sigma} p(c) \cdot d(T', c) =$$

$$= p(x) \cdot d(T, x) + p(y) \cdot d(T, y) - p(x) \cdot d(T', x) - p(y) \cdot d(T', y) = [p(x) - p(y)] \cdot [d(T, x) - d(T, y)]$$

(נשים לב ש-  $d(T', y) = d(T, x)$  ו-  $d(T', x) = d(T, y)$ )  
 נקבל הפרשת הסתברויות כפול הפרש העומקים  $\Leftarrow$  דומים בהסתברות או בעומק נקבע שלא היה שינוי בכלל



הערה חשובה: יהי  $x$  בעל  $p(x)$  מינימלי, האם בכל עץ אופטימלי  $(x, p)$  בעומק המירבי? לא! עץ עם 3 עלים שככל אחד מהם בהסתברות  $\frac{1}{3}$  והגדול ביותר נמצא לא בעלה הנמוך ביותר



אבל, מסקנה: קיים עץ אופטימלי עם  $p(x)$  בעומק המירבי. בכל עץ אופטימלי יש בעומק המירבי עלה עם הסתברות  $p(x)$ .

טענה: יהיו שתי אוטיות  $y, x$ , כך ש- $y$  לא  $\neq x$ , או קיים עץ אופטימלי כך ש- $y, x$  אחים (תחת אותו אב).

הוכחה: יהי  $T$  אופטימלי.

יהי  $a$  עלה בעומק המירבי, קלומר  $d(c, T) \leq d(a, T)$  לכל  $c$  ו- $T$  עץ מלא ולכן יש לא- $a$  אח, נקרא לו  $b$ .  
בה"כ  $p(a) \leq p(b)$ , יהי  $T'$  המתקיים מהחלפת  $a, b$ . מהטענה לעלה

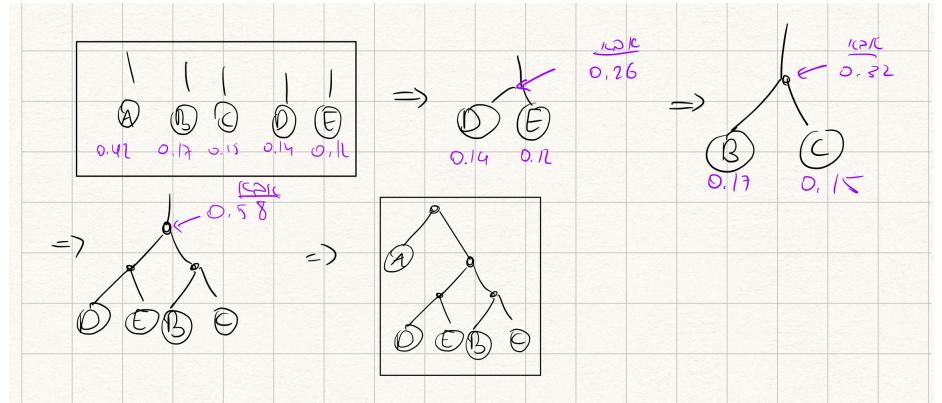
$$L(T, p) = L(T', p) + \sum_{\geq 0} [p(a) - p(x)] \cdot [d(a, T) - d(x, T)] \geq L(T', p) \underset{\text{optimal } T}{\geq} L(T, p)$$

לכן גם  $T'$  אופטימלי (נווצר סוג של סנדוויץ'), כעת סיימנו להציג את  $x$ , עכשו נחליף את  $b$  עם  $y$ .  
יהי  $T''$  מתקיים מהחלפת  $b, y$  עם  $a$ .

$$L(T', p) = L(T'', p) + \sum_{\geq 0 \text{ and } p(a) \leq p(b)} [p(b) - p(y)] \cdot [d(b, T) - d(y, T)] \geq L(T'', p) \underset{\text{optimal } T'}{\geq} L(T', p)$$

(נשים לב - יש סימטריה ולכן לא משנה אם נחליף לא- $y$   $y - p(y) - p(b)$ )  
ולכן גם  $T''$  אופטימלי ■

## 5.4.2 אלגוריתם האפמן



**טענה:** נניח  $c \neq y, x$  **לכל**  $p(c) \geq p(y) \geq p(x)$  **יהי**  $T'$  **המתכבר מ-**  $T$  **ע"י** ה החלפה של  $z$  **עם** הזוג  $x, y$  **האחים.**

$$\begin{cases} C' = C \setminus \{x, y\} \cup \{z\} \\ p(z) = p(x) + p(y) \end{cases}$$

**או**  $T$  **אופטימלי לבעה המקורית (עם**  $C$ )

**הוכחה:** **נתחיל בלהראות שיוויון כללי עבור**  $T$  **עם**  $x, y, z$  **המתאים**

$$L(T, p) = \sum_{c \in \Sigma} p(c) \cdot d(T, c) = p(x) \cdot \sum_{=d(z, T')+1} d(x, T) + p(y) \cdot \sum_{=d(z, T')+1} d(y, T) + \sum_{c \in C \setminus \{x, y\}} p(c) \cdot d(c, T)$$

$$= p(z)[1 + d(z, T')] + \sum_{c \in C' \setminus \{z\}} p(c) \cdot d(c, T') = L(T', p) + p(z)$$

**יהי**  $T_{opt}$  **אופטימלי עבור**  $C$  **בבעה המקורית ובה**  $c$  **אנחנו יודעים ש-**  $y, x$  **אחיהם ב-**  $T_{opt}$  **(לפי הטענה למטה)**

**יהי**  $T'_opt$  **מתכבר מ-**  $T_{opt}$  **ע"י** ה החלפה של זוג האחים  $x, y$  **עם**  $z$ .

**ומתקיים:**

$$L(T, p) \geq_{optimal T_{opt}} L(T_{opt}, p) = L(T'_{opt}, p) + p(z) \geq_{optimal T'} L(T', p) + p(z) = L(T, p)$$

**הערה:** **אי אפשר להניח ש**  $L(T, p) = L(T_{opt}, p)$  **(כי יתכן מספר עצים אופטימליים)**

**לכן גם**  $T$  **אופטימלי** ■

**האלגוריתם (האפמן):**  
 $c \in C$ ,  $C$  **אות**  $p(c)$  **לכל**

1. הכניסו את העלים  $c$  עם  $p(c)$  לעירמת מינימום ממוינת ע"י

2. בצע  $|c| - 1$  פעמים:

(א) הוציאו את  $x, y$  מראש העירימה

(ב) הכנסו לעירמה את  $z$  שהוא עץ עם 2 בניים  $x, y$  וסתירות של  $z$  הוא

3. החזר האיבר שבראש העירימה

סיבוכיות:  $O(n \log n)$  כאשר כמובן

### 5.4.3 אנטרופיה

הגדרה: עבור וקטור הסתברויות  $\mathbf{p}$   $\sum_i p_i = 1$ ,  $i \in [m]$  **לכל**  $p_1 \geq 0 \dots, p_m$  **או האנטרופיה** של  $\mathbf{p}$  היא

$$H(\mathbf{p}) := \sum_i p_i \log\left(\frac{1}{p_i}\right)$$

(אשר  $\log(0) = 0$ )

טענה:  $H(\mathbf{p}) \leq \min_T L(T, \mathbf{p}) < H(P) + 1$

הוכחה:

$$d_i = \lceil \log_2 \frac{1}{p_i} \rceil \text{ י"י: } \min_T L(T, \mathbf{p}) < H(P) + 1$$

$$\sum_i \left(\frac{1}{2}\right)^{d_i} \leq \sum_i \left(\frac{1}{2}\right)^{\log_2(\frac{1}{p_i})} = \sum_i p_i = 1$$

לכן קיימים עץ  $T$  עם עלים בעומק  $d_i$ .

$$L(T, \mathbf{p}) = \sum_i p_i d_i = \sum_i p_i \lceil \log_2 \frac{1}{p_i} \rceil \underset{\lceil x \rceil < x+1}{<} \sum_i p_i (\log \frac{1}{p_i} + 1) = H(\mathbf{p}) + 1$$

$$H(\mathbf{p}) \leq \min_T L(T, \mathbf{p})$$

## 6 הרצאה

### 6.1 קידוד המשך

#### 6.1.1 האלגוריתם (האפקטן) - המשך

נזכיר את השאלה: נתונם  $P = (p(1), \dots, p(m))$  והסתברויות של מילים כך ש- $0 < p_i < 1$

נדריך למצוא קוד רישא  $L(P, W) = \sum_{i=1}^m p(i)|w_i|$  כז  $W = \{w_1, \dots, w_m\}$  מינימלי

**טענה (הוכחנו):** אלגוריתם אופמן מוצא קוד כזה

$$H(p) \underset{\text{log curve}}{\leq} L(P, W) < H(P) + 1$$

**הערה (לא הוכח):** במצב האידלי הוא שהען מאוזן מתקיים  $|w_i| = \log_2(\frac{1}{p_i})$  כאשר  $p_i = (\frac{1}{2})^{|w_i|}$

נשים לב - מבנה העץ לא תמיד ירמז ישרות משחו על הקידוד שלנו ← מכיוון שעבור עץ עם 2 מיילים  $P(A) = 0.01$  ו- $P(B) = 0.99$  אותו עץ כמו כמו  $P(A) = 0.5$  ו- $P(B) = 0.5$  רק עם סיכויים שונים להגיע לכל עלה. (לשניהם יש שורש אחד ו-2 עלים)

## 6.1.2 משפט שני

נניח  $X_n = (p(1), \dots, p(m))$  משתנים מקרים בלתי תלויים המקבלים ערכים  $m, \dots, 1$  בהסתברות  $P$  בהתאם לסדרת פונקציות החה"ע:

$$\Phi_n : \{1 \dots m\}^n \rightarrow \{0, 1\}^*$$

(קלט של מספרים מ-1 עד  $m$  ופלט הוא רצף של 0 ו-1 כלשהו)

**טענה:** לכל סדרה  $\{\Phi_n\}_{n=1}^\infty$  (סדרת של פונקציות שעירות מיילים ל-0 ו-1) מתקיים:

$$E[|\Phi_n(X_1, \dots, X_n)|] \geq n \cdot H(p) \pm o(n)$$

וקיימת סדרה  $\{\Phi_n\}_{n=1}^\infty$  שמקיימת:

$$E[|\Phi_n(X_1, \dots, X_n)|] = n \cdot H(p) \pm o(n)$$

## 6.2 גרפים

### 6.2.1 תזכורת

- הגדרה:** גראף הוא זוג  $G = (V, E)$  כאשר  $V$  הם קדקודים (קבוצה כלשהי) ו-E צלעות (זוגות של קדקודים)

- הגדרה:** גראף פשוט, הצלעות לא מכונות ומתקיים:  $e = (x, y) = (y, x) \subset V$  צלע, ולפעמים נגדיר (ונזכיר קשת בין 2 עיגולים - כמו עץ) ולפעמים  $xy \in E$

- **הגדה:** גרפּ מכוון, הצלעות מכוונות, איברי  $E$  הם זוגות סדריים ואו  $e = (x, y) \neq (y, x)$  (נציין קשר עם חז' בין 2 קודקודים)

- **הגדה:** גרפּ עם לולאות, יתכן  $x \in E$  (נציין קודקוד בודד וקשר לעצמו - מכון או לא מכון)

**דוגמה לграф מכוון:**  $E = \{23, 34, 43, 42, 53\}$ ,  $V = \{2, 3, 4, 5\}$ ,  $G = (V, E)$  (להשלמים):

- **הגדה:** מסילה\מסלול - בדרך כלל כאשר  $V_j \neq V_i$  לכל  $j \neq i$  ואז אפשר להגיד מסילה פשוטה, ואם לא לפחות אומרים מהלך בגרף.

מעגל (פשוט או מהלך מעגלי) בgraf כאשר אנחנו חוזרים לקודקוד שכבר ביקרנו בו במהלך מסלול.

- **הגדה:** גרפּ קשור אם יש מסלול בין כל זוג קודקודים (זהו יחס שקילות בgraf פשוט ← אם אפשר להגיאع מקודקוד

לכלם או גרפּ הוא קשור) אחריה, קיימים מסלול בין  $x$  ל- $y$  הוא יחס שקילות ומחלקות השקילות נקראות רכיבי קשרות, וננסמן את מספר רכיבי הקשרות ב- $c(G)$ .

## 6.2.2 עצים - טענות והגדאות

**הגדה:** עץ הוא graf פשוט, קשור וחסר מעגלים  
טענות ידועות על עצים:

1. כל עץ (עם לפחות 2 קודקודים) מכיל עלה (עליה הוא קודקוד ששייך לצלע אחת בדיק - שזה דרגה 1)

2. ידי  $G$  גרפּ פשוט עם  $n$  קודקודים. אז  $G$  הוא עץ  $\iff$  (התנאים הבאים שקולים):

(א) קשור וחסר מעגלים

(ב) חסר מעגלים מקסימלי (כל צלע נוספת ניצור מעגל)

(ג) קשור מינימלי (כל צלע שנמחק הוא לא יהיה קשור)

(ד) קשור עם  $1 - n$  צלעות

(ה) חסר מעגלים עם  $1 - n$  צלעות

(ו) בין כל שני קודקודים יש מסילה, והוא יחידה.

3. בgraf חסר מעגלים או  $|E| + c(G) = |V|$

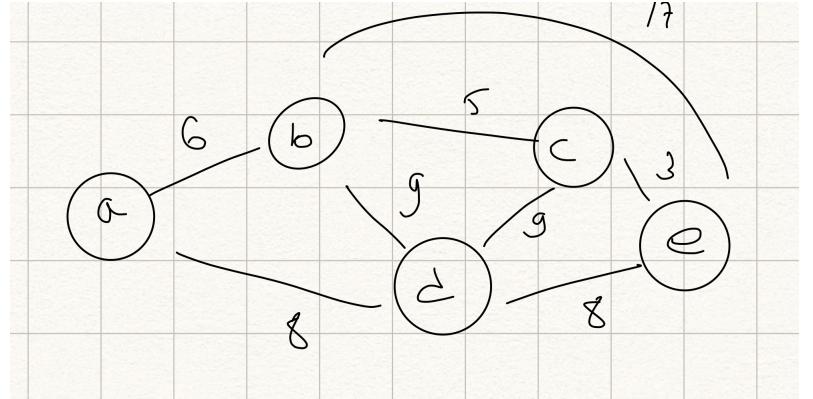
4. כל גרפּ קשור מכיל עץ פורש (=מגיע לכל הקודקודים)

5. מספר העצים על  $n$  קודקודים המסומנים  $\{1, \dots, n\}$  הוא  $n^{n-2}$  (*Cayley משפט*)

### 6.2.3 בעית עץ פורשת מינימלי

נתון גרף פשוט קשור. תהי  $w : E \rightarrow \mathbb{R}$  פונקציית משקל על הצלעות. מטרתנו: למצוא עץ פורש מינימלי ( $F \subset E$ ) כך ש-  $\sum_{e \in F} w(e)$  מינימלי.

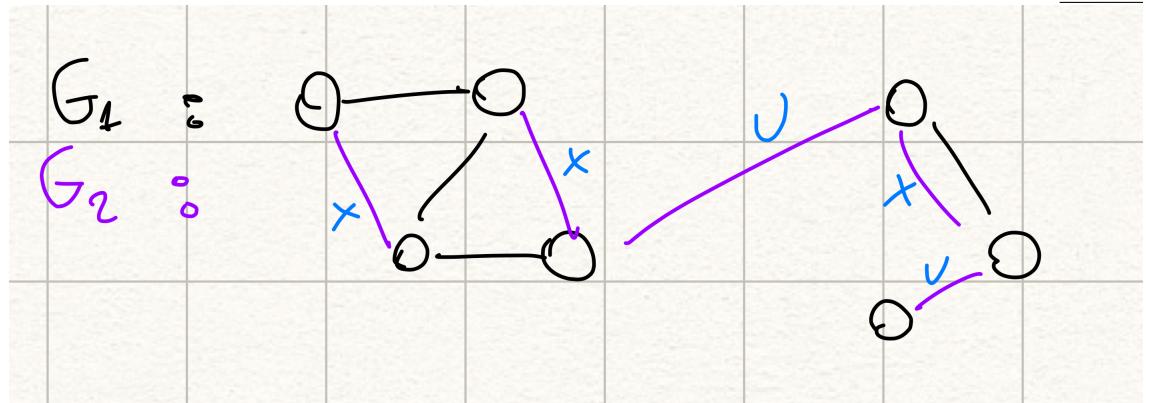
דוגמה:



. ו גם יש עוד עץ  $F' = \{ab, ce, bc, ad\}$   $\Leftarrow$  יתכנו כמה עצים פורשים מינימליים.

טענה: יהיו  $G_1 = (V, E_1)$  ו-  $G_2 = (V, E_2)$  גראפים חסרי מעגלים. אם קיימת צלע  $e \in E_2 \setminus E_1$  חסר מעגלים.

דוגמה:



מסמנים ב- $x$  ו- $\neg x$  מה אפשר להוסיף ומה לא.

הוכחה:

$$c(G_1)_{\text{no loops}} = |V| - |E_1| > |V| - |E_2| = c(G_2)$$

יהיו  $V_1, \dots, V_k$  רכיבי קשריות של  $G_1$ . נבחר קודקוד  $v_i \in V_i$  מכל רכיב קשריות. כל אחד מהקדקים  $V_1, \dots, V_k$  שייך לרכיב קשריות כלשהו מבין  $c(G_2)$  הרכיבים של  $G_2$ .

ואנחנו יודעים ש- $c(G_2) < k$  ומשובך היוונים יש  $v_i \neq v_j$  שניים לאותו רכיב של  $G_2$  (יווניים  $V_1, \dots, V_k$  ושובכים  $(c(G_2))$ ).

$\Leftarrow$  קיימת מסילה ב- $G_2$ , מכיוון שקצבותיה לא באוטו רכיב קשרות של  $G_1 \Leftarrow$  קיימת במסילה צלע  $e = uv \in E_2$  כך ש- $v \in V_i \neq V_j \in V$  כאשר  $v, u$  ברכיבי  $G_1$  שונים (סתירה)

אם היה מעגל ב- $(V, E_1 \cup \{e\})$  ולכן  $(V, E_1 \cup \{e\} \setminus e)$  חסר מעגלים.

### תכונות של עץ פורש מינימלי:

- אם יש צלע  $e$  כך ש- $w(e)$  קטן ממשקל כל צלע אחר  $\Leftarrow$  או  $e$  שייכת לכל עץ פורש מינימלי (אם לא קטן ממש אז יש עץ שכולל את  $e$ )
- אם יש בגרף מעגל צלע שמשקלה מקסימלי במעגל או  $e$  לא שייכת לאף עץ פורש מינימלי.

### 6.2.4 אלגוריתם קורסקל

#### תיאור מקוצר:

נתונים  $E \rightarrow G = (V, E)$  ונ"ל  $w : E \rightarrow \mathbb{R}$ .  
 נתחל מ- $E_0 = \emptyset$  (בלי צלעות), בהינתן  $E_k$  הינה הצלע בעלת המשקל הקטן ביותר  
 כך ש- $E_{k+1} = E_k \cup \{e_{k+1}\}$  נגדי  $e_{k+1}$  ב- $E_k$  (בלי צלעות), נסדר את צלעותיו  $e'_1, \dots, e'_{n-1}$  כך ש- $w(e'_1) \leq w(e'_2) \leq \dots \leq w(e'_{n-1})$  ו- $e'_i \in E'_k \setminus E_{k-1}$   $\forall i$ .  
 והעץ יהיה  $T = (V, E_{n-1})$

טענה: אם  $T'$  עץ פורש מינימלי, נסדר את צלעותיו  $e'_1, \dots, e'_{n-1}$  כך ש- $w(e'_1) \leq w(e'_2) \leq \dots \leq w(e'_{n-1})$ .

מסקנה:  $T$  הוא עץ פורש מינימלי, כי:

$$w(T) = \sum_{e \in T} w(e) \leq \sum_{e' \in T'} w(e') = w(T')$$

#### הוכחה של הטענה:

נתבונן בגרפים חסרי המעגלים:  $(V, E'_k) = (V, \{e'_1, \dots, e'_k\})$  ו- $(V, E_{k-1}) = (V, \{e_1, \dots, e_{k-1}\})$ .  
 ולפי תכונת ההרחבה שהרנו, קיימת צלע  $e'_i \in E'_k \setminus E_{k-1}$  כך ש- $(V, E_{k-1} \cup \{e'_i\})$  חסר מעגלים.

$$\blacksquare w(e'_k) \geq_{\text{we know}} w(e'_i) \geq_{\text{from kruskal}} w(e_k) \Leftarrow$$

מסקנה: לכל עץ פורש מינימלי יש אותה סדרת משקלות (ممויינת)

#### האלגוריתם:

**נדיר**  $E = [(u_1, v_1, w_1), (u_2, v_2, w_2) \dots]$  **כאמור**  $v_i, u_i$  **קודקודים** ו-  $w_i$  **משקל** (לא **חייב להיות** כמובן, רק אינדקסים)

```

1 def kruskal(n,E):
2     T=[] #empty vector
3     E=sorted(E,key=lambda e:e[2]) #sorted by w
4     C=[[v] for v in range(n)]
5     for u,v,w in E:
6         if C[u] != C[v]:
7             T.append((u,v))# and (u,v) to T
8             if len(C[u])>len(C[v]): u,v=u,v #switch names
9             C[v].extend(C[u])# C[v]<--C[v]UC[u] (union by value)
10            for x in C[u]:C[x]=C[v].#change to "pointer"
11    return T

```

### סיבוכיות:

- **המיון :**  $O(|E| \log(|E|))$
- **שורה 4 :**  $|V|$
- **שורה 5 :**  $|E|$
- **שורה 6+5 :**  $|V| - 1$
- **טענה:** לכל קודקוד, שורה 10 נקראת לכל היותר  $\log(|V|)$  פעמים. כי בכל קרייה, הגודל של  $C[x]$  לפחות מכפיל את עצמו. לכן השורה  $C[x] = C[v]$  תקרא לכל היותר  $O(|V| \log(|V|))$  פעמים
- **בגרף קשיר  $\Leftrightarrow$  סיבוכיות האלגוריתם  $O(|E| \log(|E|))$  (בגלל המיון)**

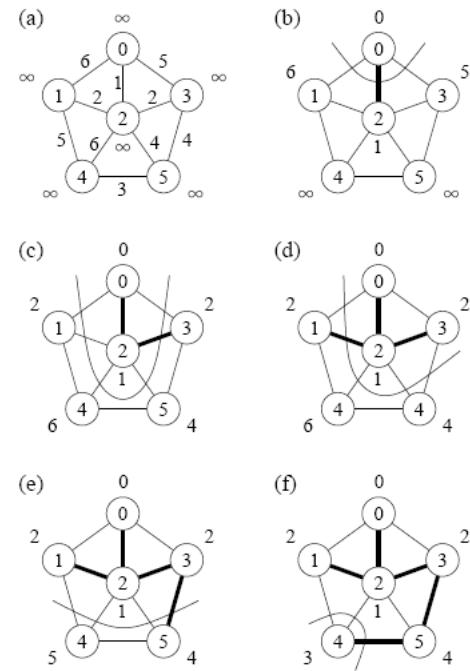
## 7.1 עץ פורש מינימלי המשך

נתון גרף פשוט  $G = (V, E)$  ופונקציית משקל  $w : E \rightarrow \mathbb{R}$

הבעיה: רוצים למצוא עץ פורש  $T$  ב- $G$  בעל משקל  $\sum_{e \in T} w(e)$  מינימלי

פתרון: האלגוריתם שרשנו בהרצאה הקודמת (בהרצאה חיימ הראה את אותו אלגוריתם שוב, עם שינוי קטן בשורה 9, בפועל עושה את אותו הדבר בדיק)

## 7.2 Prim אלגוריתם



### 7.2.1 תכנית

- יהי  $v_1$  קודקוד בgraf או נגידר  $T_1 = (V_1, E_1) = (\{v_1\}, \emptyset)$
- בהינתן  $(G, V \setminus V_k)$  תהי  $(u, v) = e \in (V_k, V \setminus V_k)$  החתך ב- $T_k = (V_k, E_k)$
- נגידר  $v_{k+1} = v$  כאשר  $E_{k+1} = E_k \cup \{e\}$  ו-  $V_{k+1} = V_k \cup \{v\}$
- טענה:  $T_k = (V_k, E_k)$  מוכל בעץ פורש מינימלי. ובפרט  $T_n$  עץ פורש מינימלי.
- הוכחה: באינדוקציה על  $k$ .

יהי  $T_k \subset T$  עץ פורש מינימלי

תהי  $e \notin T$  (בעלת משקל מינימלי בחתך)  $\leftarrow$  אם  $e \in T$  או גם  $e \in T_{k+1} \subset T$  וסימנו, אם אז  $f \in T \cup \{e\}$  מכיל מעגל.

תהי  $f$  צלע במעגל שנמצאת בחתך ( $f \in (V_k, V \setminus V_k)$  או  $f \in (V_k, V \setminus V_k)$ )

מבחן  $e$  מתקיים  $w(T') \leq w(T)$  ו-  $T' = T \setminus \{f\} \cup \{e\}$  מכיל את  $(T_{k+1})$  (שתיים מינימליים,  $T'$  מכיל את  $w(T') \leq w(T)$  יי

## 7.2.2 אלגוריתם prim

הנחות:

- אפשר לקבל שכנים של קודקוד  $v$  ע"י  $G.neighbors(v)$
- הוא מספר הקודקודים  $G.V$
- צורך עירימה  $H$  שאפשר למצוא בה מהר צלע  $(u, v)$  (לא ע"י חיפוש).  
לכל צלע  $(u, v)$  נוסיף שדה  $(u, v).pointer$  שמצוין ב- $H$ , ונעדכן אותו שמשנים צלעות ב- $H$ .

```
1 def prim(graph G, vertex v): #v is random vertex to start with
2     T = graph(V={v}, E=empty)
3     H = start_heap({(u,v): u in G.neighbors(v)}) #order by weight (u,v)
4     for k in {2,3,...,|G.V|}: #G.V num of vertexes
5         (u,v) = H.pop() #get first edge
6         T.add_vertex(u)
7         T.add_edge((u,v))
8         for x in G.neighbors(u):
9             if x in T: H.remove((u,x)) # x already in T, no circle needed
                and we already add the shortest edge with x
10            else : H.add ((x,u)) #x not in T
11
12 return T
```

סיכום:

```
1 def prim(graph G, vertex v):
2     T = graph(V={v}, E=empty) # O(1)
3     H = start_heap({(u,v): u in G.neighbors(v)}) # O(|V|)
4     for k in {2,3,...,|G.V|}: # |V|
5         (u,v) = H.pop() #|V|log|V|
6         T.add_vertex(u) #|V|
7         T.add_edge((u,v))#|V|
8         for x in G.neighbors(u):# 2|E| - every edge will calculate twice
            each time with different side
9             if x in T: H.remove((u,x)) #|E|log|E|
10            else : H.add ((x,u)) #|E|log|E|
11
12 return T
```

סה"כ  $|E|log(|E|)$

### 7.2.3 אלגוריתם prim גירסא ב'

גירסא א' - הערימה  $H$  מכילה את כל הצלעות  $(u, v)$  בחתך כאשר  $u \notin T, v \in T$

גירסא ב' - הערימה  $H$  מכילה לכל קדוקוד  $T \neq u$  את הצלע  $(u, v)$  ממינימלי מבין  $v \in T$ .

הנחות נוספת:

- לכל קדוקוד  $x$  שומרים  $x.pointerToH$  שמצויע למקומות של  $(x, *)$  בערימה (כאשר \* זה איזשהו קדוקוד אחר)

```

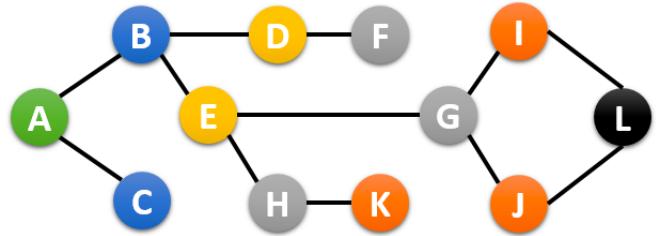
13  def prim(graph G, vertex v): #v is random vertex to start with
14      T = graph(V={v}, E=empty)
15      H = start_heap({(u,v): u in G.neighbors(v)}) #order by weight (u,v)
16      for k in {2,3,.....,|G.V|}: #G.V num of vertexes
17          (u,v) = H.pop() #get first edge
18          T.add_vertex(u)
19          T.add_edge((u,v))
20      for x in G.neighbors(u):
21          if x not in T:
22              (x,y) = H.find((x,*)) #* := any vertex
23              if (x,y).weight > (x,u).weight:
24                  H.replace((x,y),(x,u)) # switch to (x,u)
25      return T

```

סבירויות: מואוד דומה לסבירויות של גירסא א', אבל אנחנו חסכו ניימן טיפה בגודל של הערימה ודברים קוראים קצת פחות פעמים

אבל לבסוף אנחנו עדין  $|E|\log|E|$

טענה: בגרף קשיר  $O(\log(|E|)) = O(\log|V|)$



Breadth-First Search (BFS)

נתון גרף מכוון  $G$  וקודקוד התחלתי  $u$  (אם הגרף לא מכוון נחליף כל צלע ל $2$  צלעות מכוונות)

### 7.3.1 מה *BFS* עושה?

- עובר על כל הקודקודים שאפשר להגעה אליהם מ- $u$ , ואו בכל קודקוד עוזים שהוא שצרי (תלו依 במטרה)
- מטרות אפשריות:
  - למצוא רכיב קשירות (כל הקודקודים הנגישים מ- $u$ )
  - למצוא עץ פורש
  - לחשב מרחקים מקודקוד  $u$  לכל קודקוד  $x$  ( $d(u, x)$  היא המסלילה הקצרה ביותר - מס' הצלעות)
  - למצוא קודקוד שהיפשנו

### 7.3.2 האלגוריתם

נניח שהמטרה למצוא רכיב קשירות

```

1 def BFS (graph G,vertex u):
2     S=[]
3     Q=[u] # queue - first in first out, insert from right, out from left
4     for v in G.V:
5         v.visited = false
6     u.visited = true
7     while Q not empty:
8         v=Q.pop_left()
9         for x in G.neighbors (v):
10            if not x.visited:
11                Q.add_from_right(x)
12                x.visited = true
13            S.append(v)
14    return S
  
```

נניח שהמטרה למצוא לכל קודקוד  $v$  את אורך המסלילה הקצרה ביותר, וגם המסלילה כלשהי קצרה ביותר.

## 8.1 המשך BFS

סיבוכיות:

- שורה 4 : מעבר על כל הקודקודים  $|V|$
  - שורה 6: הגדרות אתחול<sup>1</sup>
  - שורה 7+8 : ללואה והגדרות  $|V'|$  - קודקודים שמחוברים ל- $u$
  - שורה 9 : לולאת  $|E'| for$  - הצלעות הנגישות
  - שורה 10 : תנאי מתקיים  $|V'|$
- $\Leftarrow$  סה"כ :  $O(|E| + |V|)$  (יכול להיות הרבה הרבה צלעות בגרף מכובן)

סימוניים:

$y$  אורך המסלול המכובן הקצר ביותר מ- $x$  ל- $y$   $= d(x, y) = dist(x, y)$  •

$x$  המרחק ל- $x$  (אחרי שנקבע)  $d(x) = x.dist$  •

(אחרי שנקבע)  $p(x) = x.parent$  •

טענה: לכל  $x \in V$  מתקיים :

1.  $d(x) = dist(u, x)$

2. מסלול קצר ביותר מ- $u$  ל- $x$  הוא:

$$u = p^{d(x)}(x) \rightarrow \dots \rightarrow p(p(x)) \rightarrow p(x) \rightarrow x$$

הערה: נזכיר

1. אם אין מסלול  $d(x) = \infty$

2. במקרה זה  $p(x) = null$

הוכחה:  $2 \Leftarrow 1$

אינדוקציה על  $d(x)$ :

בסיס: במקרה  $d(x) = 0$  וזה ברור  $x = u$

צעד: בהינתן  $d(x) = d(u, x)$  מספר סופי, ולפי הנחת האידוקציה יש:

$$u = p^{d(x)}(x) \rightarrow \dots \rightarrow p(p(x)) \rightarrow p(x)$$

אורך המסלול  $1 - d(p(x)) = d(x)$  ונוסיף למסלול את  $x$  ונקבל:

$$u = p^{d(x)}(x) \rightarrow \dots \rightarrow p(p(x)) \rightarrow p(x) \rightarrow x$$

באורך  $d(x)$ .

אם  $\infty$  אז אין מסלול ואו  $\infty$  (הוכחנו בקרה שסעיף 1 הוא  $\infty$ ). אחרת, (לא הבנתי, להשלים)

נסמן  $S = [v_1, \dots, v_n]$  הקודקודים לפי סדר הגילויים, הכנסתם ל- $-Q$ , הוצאתם מ- $-Q$ , טיפול בהם, הכנסתם ל- $S$

נוכיח את (1) - לכל  $d(u, v_j) \leq d(v_j)$ ,  $v_j \in S$  באינדוקציה על  $d(y)$

$$d(v_j) = d(p(v_j)) + 1 \geq_{induction} d(u, p(v_j)) + 1 \geq d(u, v_j)$$

נוכיח את (2) -

$$0 \leq d(v_1) \leq d(v_2) \leq d(v_3) \leq \dots \leq d(v_n)$$

באינדוקציה על  $j$  עבר  $v_j$ .

בסיס:  $0 = d(v_1) \leq d(v_2)$

צעד: נוכחה עבור  $v_j, v_{j+1}$

אם  $p(v_j) = p(v_{j+1})$  (אותו הורה, יכול לקרות) אז :

$$d(v_j) = d(p(v_j)) + 1 = d(v_{j+1})$$

אחרת, נסמן:  $(\text{בגכל שהסדר ב-} S \text{ הוא הסדר של הכנסה ל-} -Q \text{ שהוא})$   
הסדר של מעבר השכנים של ההורה

$$\Rightarrow d(v_j) = d(v_k) \leq d(v_l) + 1 = d(v_j + 1)$$

לכל  $v_j \in S$  מתקיים  $d(v_j) \leq d(u, v_j)$ , באינדוקציה על  $d(u, v_j)$

ניקח מסלול קצר ביותר מ- $v_j$  ו- $v_k = p(v_j)$  הורה אחד שהגיע ראשון ל- $v_j$ , ו-  
הורה שני  $k \leq l$  שיווין שהם אוטו קודקוד

$$d(v_j) = d(p(v_j)) + 1 \leq_{v_k} d(v_l) + 1 \leq_{induction} d(u, v_l) + 1 = d(u, v_j)$$

\* - כי אין מסלול קצר יותר ל- $v_l$ , האחרת היה מסלול קצר יותר ל- $v_j$   
כנדרש  $d(v_j) = d(u, v_j) \stackrel{:}{=} (2) + (1)$

## 8.2 מסלולים קצרים ביותר בגרף ממושקל

נתון גרף מכובן  $G = (V, E)$  ופונקציית משקל  $w : E \rightarrow (0, \infty]$   
 אם  $(x, y) \notin E$  אז  $w((x, y)) = \infty$  (לצורך הנוחות נגדיר  $\infty$  כמשקל של קשת לא קיימת).

הגדרה: (מחדש)

$$d(x, y) = \min \left\{ \sum_{e_i} w(e_i) : x \rightarrow_{e_1} \cdot \rightarrow_{e_2} \cdots \rightarrow y \right\}$$

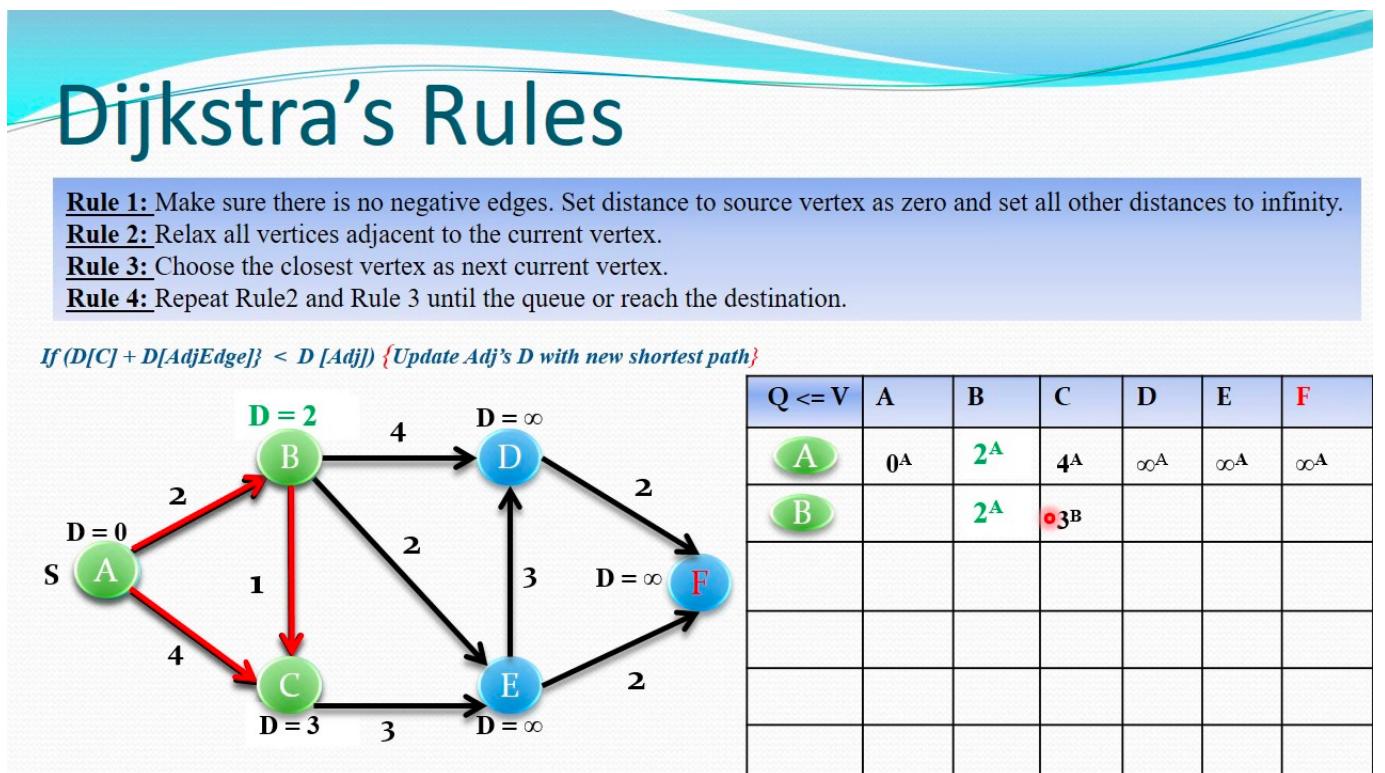
הערה:  $BFS$  הוא המקרה הפרטי  $w(e) = \begin{cases} 1 & e \in E \\ \infty & \text{else} \end{cases}$

בעיה: בהינתן  $V$  מה המרחק  $d(u, v)$ , ומוצא מסלול קצר ביותר.

דוגמאות: ניוטון

ראינו: אלגוריתם תכנון דינמי שנקרא פלויד-זרשל ומתאר את  $d(u, v)$  לכל  $u, v \in V$  בסיבוכיות  $|V|^3$ . (בתרגול)

### אלגוריתם דייקסטרה 8.2.1 Dijkstra



בהינתן כנ"ל מוצא  $d(u, v)$  (וגם מסלול) ל- $u$  הנתון וכל  $v$  בגרף.  
 קיבל השראה מהאלגוריתמים הבאים:

1. פלייד-ורשל : תכנון דינמי

2. BFS : להוסיף לפי הסדר את המרחק  $m_u$

: prim .3

התוכנית: בכל שלב תהיה קבוצה  $S$  של קודקודים שכבר מצאו  $.v \in S$   $d(u, v)$

$$S = \{w\}$$

בשלב ה- $k$ : נזכיר את הקודקוד הכי קרוב ל- $u$  שאינו ב- $S$ .

בכל שלב מבצע את המסלול הקצר ביותר לכל  $v$  דרך קודודי  $S$

האלגוריתם:  $G$  גראף,  $w, u$  קודקודים

```
1 def dijkstra(G,w,u):
2     for x in G.V:
3         x.d=infinity
4         x.p=null
5     u.d=0
6     H=start_heap(G.V,order in minimum heap by x.d)
7     while H not empty:
8         v=H.pop() #add to S
9         for x in G.neighbors(v):
10            if x in H and (v.d+w(v,x)<x.d):
11                x.d=v.d+w(v,x)
12                x.p=v
13                H.heapify(x) #we need to save x place in the heap to make
14                this
15    return G # now include distance and routes
```

טענה: אם נמחק את ההפניות של  $v.d$  בקוד (שורות 10,11 זה ממש דומה ל  $prim$ )

סיבוכיות: סה"כ  $O(|E|\log(|V|))$

```
1 def dijkstra(G,w,u):
2     for x in G.V: # |V|
3         x.d=infinity
4         x.p=null
5     u.d=0
6     H=start_heap(G.V,order in minimum heap by x.d) # O(|V|)
7     while H not empty:
8         v=H.pop() #|V|log(|V|)
9         for x in G.neighbors(v): #|E|
10            if x in H and (v.d+w(v,x)<x.d):
11                x.d=v.d+w(v,x)
12                x.p=v
13                H.heapify(x) #|E|log(|V|)
14    return G # now include distance and routes
```

**נכונות:** נסמן  $(x)$  בשלב ה-  $k$  וגם  $S = \{v_1, \dots, v_n\}$  את הקדדים לפי סדר שליפתם מ-  $H$ .

**משפט:** (עבור הוכחת הנכונות)

$$d_k(x) \leq d(u, x) \quad \text{לכל } k \quad (1)$$

$$d_k(x) = d(u, x) \quad \text{או } x \in S_k = \{v_1, \dots, v_k\} \quad (2)$$

**מסקנה:**  $d_n(x) = d(u, x)$  וגם אפשר למצוא את המסלול בעזרה  $p_n(x)$  כמו ב-  $BFS$

**הוכחה:** (משפט) באינדוקציה על  $k$

**בסיס:** עבור  $1$   $d_1(u) = 0 = d(u, u)$ , ומתקיים  $S_1 = \{u\}$

$$d(u, x) \leq \infty = d_1(x) \quad \text{ברור עבור } u \neq x$$

**שלב האינדוקציה:**

עבור (1): יש מסלול באורך  $d_k(x) = d_{k+1}(x)$  מ-  $u$  ל-  $x$  או  $d_k(x) = d_{k+1}(x)$  וזו זה נובע מהאינדוקציה או ש-  $d_{k+1}(x) = \min_{v_k \in S_k} d_k(v_k) + w(v_k, x)$  ויש את הצלע  $w(v_k, x)$ . וזו יש מסלול מ-  $u$  ל-  $x$  ממשקל  $d_{k+1}(x) \geq d(u, x)$  ולכן  $d_{k+1}(x) \geq d(u, x)$  המסלול המינימלי

עבור (2): ניקח מסלול קצר ביותר מ-  $x$ . מספיק לבדוק את (2) בשלב  $k+1$  עבור  $v_{k+1}$

יהי  $y$  הקודקוד הראשון במסלול ש?? ב-  $S_k$

יהי  $z$  הקודקוד שלפניו במסלול (יתכן  $z = u$ )

$$d(u, y) \underset{\text{from choose shortest path}}{=} d(u, z) + w(z, y) \underset{*}{\geq} d_k(y)$$

\*- בשלב הכנסת  $z$  (לפניהם), מעדכו  $y$  בתורו שכן של  $z$ , כאשר  $d_i(z)$  באותו שלב היה בדיק  $d(u, z)$  באינדוקציה.

$$d_k(x) \geq d(u, x) \underset{\text{under path of shortest path}}{\geq} d(u, y) = d_k(y) \underset{x \text{ was in head of the heap}}{\geq} d_k(x)$$

■

## 9 הרצאה

### 9.1 חיפוש לעומק - $DFS$

שאלה: איך יוצאים מבוך? הולכים ← מגיעים להתפצלות, פונים למקום שלא היינו בו ← אם מגיעים למבי סתום, מתחילה לחזור על עקבותינו עד הצומת האחרון שהייתה בה אופציה שלא ניסינו ← אם מגיעים להתפצלות שאין בה אופציה שלא ניסינו או גם מתחילה לחזור אחרת

נוכיר: תור הוא  $FIFO$  ומחסנית היא  $LIFO$

אלגוריתם: קודם כל נעשה מעבר , כאשר  $G$  הוא גראף מכוון ו $u$  הוא קודקוד התחלתי

```

1 def DFS(G,u):
2     u.visited=True
3     for v in G.out_neighbours(u):
4         if not v.visited:
5             DFS(G,v)

```

סיבוכיות:  $\max\{O(|V|), O(|E|)\}$  וכשאנו יודעים את הגדים זה תאכלס

טענה: בהנחה שבתחלתה היה  $v.visited = True$  אז האלגוריתם יעצור ואז יהיה  $v.visited = False$  לכל  $v \in V$ , או יש מסלול (מכיוון) מ- $u$  ל- $v$ .  $\iff$

הוכחה: מספר הקודדים סופי, לכל קודקוד  $v$ ,  $DFS(G, v)$  נקרא לכל היותר פעם אחת בגלל  $...visited$ .  
 (1)  $u_{i+1}.visited = False$ ,  $u_i.visited = True$ , לא יתכן שבסוף הפעלה,  $u_{i+1}.visited = True = u.visited$  מאופן פעולות האלגוריתם  
 (2) אם  $v.visited = True$  אז קיימת מסילה מ- $u$  ל- $v$

רישימת שימושים ל- $DFS$ :

- לעבר על קודדים נגישי מ- $u$  (אם  $G$  פשוט רכיב קשורות)
- למצוא עץ מסלולים לקודדים אלה (אם  $G$  פשוט עצם פורש לרכיב של  $u$ )
- למצוא יציאה ממבוקש

ניתוח פעילות  $DFS$   
 התוכנית היא להוציא:

- לכל קודקוד משתנה :

$discover = d$  – זמן הגילוי

$finish = f$  – זמן הסיום

- צבעים לקודדים לפי מצבים:

– לבן – לפני גילוי

– אפור – בין גילוי לסיום

– שחור – אחרי סיום

- מספר קודדות התחלה כדי להגיע לכל הקודדים

- עץ מסלולים  $T$  (או יער)

## אלגוריתם הצבעים

```

7 def DFS_main(G,u):
8     T=[] #global varibale
9     time = 0 #time is global variable
10    for v in G.V:
11        v.color = WHITE
12    for u in G.V:
13        if not u.color==WHITE:
14            DFS_visit(G,u)
15    return T
16
17 def DFS_visit(G,u):
18     u.color=GRAY
19     time +=1
20     a.d=time
21     for v in G.outneighbors(u):
22         if v.color==WHITE:
23             DFS_visit(G,v)
24             T.append((u,v))
25     u.color = BLACK
26     time +=1
27     a.f = time

```

**אבחן(1):** לכל  $d(u) < f(u)$ ,  $v$

**אבחן(2):** לכל  $v \in u$  מתקיים  $d(v) < f(u)$  לא יכול לסייע לפני ש- $v$  מתגלה

**אבחן(3):**  $x$  צצא של  $y$  בעץ  $\iff$  קיימת שרשרת קריאות bfs\_visit מ- $y$  ל- $x$  היא אפור כאשר  $x$  מתגלה  
 $d(x) \in [d(y), f(y)] \iff$

**סוגרים:** (תמונה) עבור  $v \neq u$ . נניח בה "כ"  $d(u) < d(v)$  יש 2 אפשרויות:  
 •  $d(u) < f(u) < d(v)$  ו- $v$  לא צצא של  $u$  ו- $v$  לא צצא של  $u$  בעץ \ עיר  $T$ .  
 •  $d(u) < d(v) < f(v) < f(u)$  ו- $v$  צצא של  $u$  בעץ \ עיר  $T$

הוכחה: האם  $d(v) < f(u)$

**אם לא**  $d(u) < f(u) < d(v) < f(v)$  ומרקח (1) נובע  $v$  לא צצא של  $u$  (לפי אבחן(3))  
**אם כן**  $d(u) < f(u) < d(v) < f(v) < f(u)$  לפני אבחן(3) כאשר שמים לב לשרשרת הקריאות הרקורסיביות מ- $u$  ל- $v$  וגם לפי 3  $v$  צצא של  $u$ .

**תכונת המסלול הלבן:**

נתון מסלול בגרף מ- $u_1$  ל- $u_m$  ו- $u_1$  התגלה ראשונה  $d(u_1) = \min(d(u_1), d(u_2), \dots, d(u_m))$  בעץ \ עיר DFS

**הערה:** המסלול הלבן בזמן  $d(u_1)$

**הוכחה:**

$$d(u_1) \underset{known}{<} d(u_2) \underset{number(2)}{<} f(u_1)$$

**מתכונת הסוגרים.**

$$d(u_1) \underset{known}{<} d(u_m) \underset{number(2)}{<} f(u_{m-1}) \underset{induction}{<} f(u_1)$$

**מתכונת הסוגרים**

**10.1 תזוכות**האלגוריתם DFS:

dfs main .1

(א) צובעים את כל הקודקודים לבן

(ב) מאתחיל  $time = 0$ (ג) עוברים על כל קודקוד  $u$ , אם  $u$  לבן אז מרכיבים

dfs\_visit(u) .2

(א) צובעים את  $u$  באפור(ב) רושמים את הזמן ב- $d(u)$ (ג) לכל שכן  $v$  של  $u$ :  $(u \rightarrow v)$ 

dfs\_visit(v)

(ד) צובעים את  $u$  בשחור(ה) רושמים את הזמן ב- $f(u)$ **10.2 המשך DFS**תכונת המסלול השחור:

אם

$$u_1 \rightarrow u_2 \rightarrow u_3 \rightarrow \dots \rightarrow u_{m-1} \rightarrow u_m$$

ונניח ש- $u_m$  הסתיים אחרון, ככלומר בזמן  $f(u_m) \leq i \leq m$  (כלומר בזמן  $f(u_m) > f(u_i)$  שאר המסלול כבר שחור)או  $u_1$  צאצא של  $u_m$  בעץ יער הדיאגרמה

הוכחה: באינדוקציה על אורך המסלול

צעד ראשון: נראה ש- $u_n$  צאצא של  $u_{n-1}$ 

$$d(u_n) \underset{\text{from 1}}{<} d(/.....$$

צעד האינדוקציה: בהנחה ש- $u_n$  צאצא של  $u_n$  נראה ש- $u_1$  צאצא של  $u_n$ .

$$d(u_n) \underset{\text{induction}}{<} d(u_{i+1}) \underset{\text{from 2}}{<} f(u_i) \underset{\text{known}}{<} f(u_n)$$

מתכונת הסוגרים  $u_m \leq u_i \Leftarrow d(u_m) < d(u_i) < f(u_i)$

### 10.3 מיוון טופולוגי

נתון גרפ' מכון חסר מעגלים (חסר מעגלים מכוונים)  
רוצחים למצוא את הקודקודים כלומר לרשום  $V = (v_1, v_2, \dots, v_n)$  כך שאם  $i < j$  אז  $v_i \rightarrow v_j$   
הערה: מיוון טופולוגי לא הכרח יחיד, צריך למצוא מיוון כלשהו  
הערה: להשלים

טענה: יהי  $G$  מכון חסר מעגלים.  
לכל צלע  $v \rightarrow u$ , מתקיים  $f(v) < f(u)$   
הוכחה נניח שלא, אז  $d(v) < f(v) < f(u) < d(u)$ .  
 $\Leftarrow DFS$  בуз' ה- $u$  צאצא של  $v$  ב- $DFS$  יש מעגל.

#### 10.3.1 איך עושים מיוון טופולוגי

בסוף הפוקציה  $dfs\_visit(u)$  מוסיפים את  $u$  לתחילה הרשימה  
תרגיל לזהות אם יש מעגל מכון בעז מכון נתון.  
תרגיל האם יש מיוון טופולוגי של גרפ' כלשהו שלא מתkelig (התשובה אמורה להיות לא)

#### 10.3.2 אלגוריתם למציאת רכיבי קשרות חזקה

הגדרה: יהי  $G = (V, E)$  גרפ' מכון. נגדיר  $v \sim u$  אם קיים מסלול  $u \rightarrow v$ , ולהפנ'.

היחס  $v \sim u$  הוא יחס שיקילות. מחלקות השיקילות נקראות רכיבי קשרות חזקה של  $G$ .

האלגוריתם: בשלבים הבאים:

- (1) - הפעל  $DFS$  על כל הגרף  $G$  ונשמר את זמני הסיום  $f(v)$  של כל קודקוד
  - (2) - הפעיל  $DFS$  על הגרף ההפוך כאשר בחירת הקודקודים היא לפי  $f$  בסדר יורד dfs\_visit( $v_i$ ) כל עוד נותרו קודקודים בגרף  $\leftarrow$  יהי  $v_i$  קודקוד עם  $f(v_i)$  מקסימלי ( $i = 1, 2, \dots$ ) הפעל  $\leftarrow$  הפעל  $f(v_i)$  עם צלעות מכוונות הפוך.
- הסירו את הקודקודים שעברנו עליהם והגדירו אותם בתור רכיבי קשרות חזקה  $V_i$

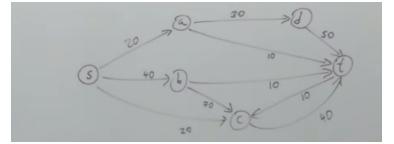
טענה:  $V_i$  הם רכיבי קשרות החזקה

הוכחה: ראשית, לא יתכנו צלעות מקודקוד של  $V_i$  לקודקוד של  $V_j$  עבור  $i < j$  עבורו  $V_i$  שוניות מהאחרת בהרצת  $dfs\_visit(v_j)$  על הגרף ההפוך, הינו מגיעים לקודקוד של  $V_i$   $\Leftarrow$  לכן קודקודים ב- $V_i$  שוניות הם ברכיבי קשרות חזקה שונים  
מצד שני - נראה שכל  $V_i$  נראה שלכל  $v \in V_i$ ,  $v \sim v_i$ . יש מסלול  $m_v$  ל- $v$  צאצא של  $v_i$  ב- $DFS$  עם הגרף ההפוך

$$v \rightarrow \dots \rightarrow v_i$$

$f(v_i) > f(v)$  מבחירה  $v_i$  לפי התכונות המסלול השחור  $v$  צאצא של  $v_i$  לכל  $v$  במסלול

## 10.4 רשותות זרימה



נתון גרף ממכוון  $G = (V, E)$  שני קודקודים מיוחדים.

• **source**  $s$  •

• **בור (target) sink**  $t$  •

**ונתונה פונקציה**  $C : E \rightarrow [0, \infty)$  קיבול (capacity)

**הגדרה:** זרימה (flow) היא פונקציה  $f : E \rightarrow [0, \infty)$  כך ש:

(1) **לכל צלע**  $e \in E$  מתקיים  $0 \leq f(e) \leq C(e)$

(2) **לכל קודקוד**  $u \in V \setminus \{s, t\}$  מתקיים  $\sum_{\substack{\text{output} \\ \text{input}}} f((u, v)) = \sum_{(w, u) \in E} f((w, u))$

**הגדרה:** ערך הזרימה (כמויות החומר הזורמת)

$$val(f) := \sum_{(s, v) \in E} f((s, v)) - \sum_{(w, s) \in E} f((w, s))$$

**מטרה:** ישנן מספר מטרות:

- למצוא מהו  $val(f)$  המקסימלי: (אפשר לשלוח למזה הוא קיים בכלל)

- למצוא זרימה  $f$  שימושית  $val(f) = \text{maximum flow}$

**טענה:** חסם על  $\sum_{(s, v) \in E} C((s, v))$  -  $val(f)$   
בדוגמה - ניתן לראות שמתקיים  $val(f) \leq 80$

ו עוד חסם הוא  $\sum_{(w, t)} c(w, t)$   
בדוגמה -  $val(f) \leq 110$

**טענה:** חסם יותר כללי

בעורת החתק  $(\{s, b, c\}, \{a, b, t\})$

בדוגמה -  $val(f) \leq 70$

**הגדרה:** חתך  $s - t$  הוא חלוקה של הקודקודים  $v$  לשתי קבוצות  $(S, V \setminus S)$  כאשר  $s \in S$  ו-  $t \in V \setminus S$

**סימונים:** בהינתן שתי קבוצות  $A, B$  (לאו דווקא חתך) נסמן

$$f(A, B) = \sum_{(a, b) \in E} f((a, b)) \quad C((A, B)) = \sum_{(a, b) \in E} c((a, b))$$

**טענה:** לכל זרימה  $f$ , ולכל חתך  $(S, V \setminus S)$  מתקיים:

$$val(f) = f(S, V \setminus S) - f(V \setminus S, S) \quad (1)$$

$$val(f) \leq C(S, V \setminus S) \quad (2)$$

capacity of hatah

**הוכחה:** הגדכנו את הזרימה

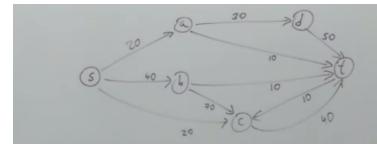
$$val(f) := \sum_{(s,v) \in E} f((s,v)) - \sum_{(w,s) \in E} f((w,s)) \stackrel{\text{Conservation of mass}}{=} \sum_{u \in S} [\sum_{(u,v) \in E} f((u,v)) - \sum_{(w,u) \in E} f((w,u))] \quad (1)$$

$$= f(S, V) - f(V, S) = f(S, S) + f(S, V \setminus S) - [f(S, S) + f(V \setminus S, S)] = f(S, V \setminus S) - f(V \setminus S, S)$$

.(1) **הוכחנו את**

$$f(S, V \setminus S) - f(V \setminus S, S) \leq f(S, V \setminus S) = \sum_{u \in S, v \in V \setminus S, (u,v) \in E} f((u,v)) \leq \sum C((u,v)) = C(S, V \setminus S)$$

(2) **הוכחנו את**

11.1 המשך ורימה

דוגמה חתך קלשו עבור -

$$S = \{s, b, c\} \quad V \setminus S = \{t, a, d\} \Rightarrow C(S, V \setminus S) = 70$$

הגדרה: יהיו  $v, u \in V$  קודקודים. מסלול  $u - v$  (לא מכובן) הוא סדרה של צלעות -

$$u \xrightarrow{e_1} v_1 \xrightarrow{e_2} v_2 \xrightarrow{e_3} v_3 \xrightarrow{e_4} v_4 \xrightarrow{e_5} v$$

כל הקודקודים  $v$  שונים זה מזה, וכל צלע  $e \in E$  שנמצאת במסלול יש כיון, זאת אומרת  
 $v_i \leftarrow v_{i+1}$  או  $v_i \rightarrow v_{i+1}$

הגדרה: בהינתן  $G$ , מסלול  $P$  ורימה  $f$ , העודף של  $P, f$  מוגדר להיות:

$$\text{excess}(f, p) := \min_i(\text{excess}(f, P, e_i))$$

$$\text{excess}(f, P, e) = \begin{cases} c(e_i) - f(e_i) & e_i \text{ with the flow} \\ f(e_i) & e_i \text{ against the flow} \end{cases}$$

דוגמה:

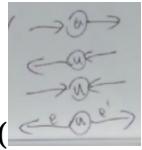
טענה: אם  $f$  ורימה ו- $P$  מסלול  $s - t$  אז קיימת ורימה  $f'$  כך ש-

הוכחה: נגיד:

$$f'(e) = \begin{cases} f(e) & e \notin P \\ f(e) + \text{excess}(f, p) & e \text{ with the flow} \\ f(e) - \text{excess}(f, p) & e \text{ against the flow} \end{cases}$$

זרימה:  $f'$

(מהגדרת העודף (תבדקו את שני המקרים))



(2) **לכל**  $f'$  **מהגדרת**  $f'$  **4 מקרים :**  
**למשל במקרה הרבי עי מתקיים**

$$\sum_{\substack{(u,v) \text{ without } e, e'}} f'(u, v) = \sum_{(u,v)} f(u, v) + (f(e) - x) + (f(e') + x)$$

**ומתקיים** (2) **מקרים:** הורדנו צלע  $S$  **שהיא עם** **כיוון או נגד** **הכוון** (ב-2 המקרים הנוסחא נכונה)

**משפט:** זרימה מקסימלית חתך מינימלי  
**תהי**  $G$  **רשת זרימה,**  $f$  **פונקציה זרימה (**c **קיובל),** **או הדברים הבאים שקולים:**

(1) **-**  $f$  **היא זרימה עם**  $\text{val}(f)$  **מקסימלי**

(2) **-** **קיים חתך**  $(S, V \setminus S)$ ,  $s - t$  **כך ש-**  $\text{val}(f) = c(S, V \setminus S)$

(3) **- לא קיים מסלול**  $s - t$ ,  $P$ , **כך ש-**  $\text{excess}(f, P) > 0$

**מסקנה:** השיוויון הבא:

$$\max \{ \text{val}(f) : f \text{ flow in } G \} = \min \{ c(S, V \setminus S) : (S, V \setminus S) \text{ cut in } s - t \}$$

**הוכחה:** הוכחנו קודם,  $\geq$  ניקח זרימה מקסימלית **לפי** (2) (למעלה) **ערכה לפחות** קיבל החתך המינימלי

**הוכחה (משפט)**

.**3  $\Leftarrow 1$ :** אחרת **לפי** הטענה היינו מקבלים זרימה  $f'$  **עם** **ערך יותר גדול** **בסתירה** **לקסימליות**  $f$ .

**1  $\Leftarrow 2$ :** **כि** **לכל** **זרימה אחרת**  $f'$  **מתקיים**  $\text{val}(f') \leq c(S, V \setminus S) = \text{val}(f)$  **לכן**  $f$  **מקסימלית**  
 $t \notin S, s \in S$ ,  $S := \{u \in V : \text{exist path } s - u, P \text{ with } \text{excess}(f, P) > 0\}$  **כאשר**

**2  $\Leftarrow 3$ :** **נגיד**  $\{(u, v) \in E : v \in V \setminus S \text{ ו- } u \in S\}$  **או**  $\{(u, v) \in E : u \in V \setminus S \text{ ו- } v \in S\}$

**א.** **תהי**  $E$  **נוקט את הצלע**  $(u, v)$  **ונקבל מסלול**  $s - v$  **שהעודף** **הוא**  
**אחרת ניקח מסלול**  $s - u$  **עם** **עודף**  $s - u$  **עם**  $\text{excess}(f, P) > 0$  **ו-** **נוקט מסלול**  $v - s$  **שהעודף** **הוא**

$v \notin S$  **ו- סתירה**  $\text{min}(\text{excess}(f, P), c((u, v)) - f((u, v))) > 0$

**ב.** **תהי**  $f((u, v)) = 0$ . **טענה:**  $v \notin S, u \in S, (u, v) \in E$

**הוכחה:** **אחרת**, **ניקח מסלול**  $u - s$  **עם**  $\text{excess}(f, P) > 0$ . **נוקט את הצלע הפוך**  $v - u$  **ומסלול.**

**נקבל מסלול**  $v - s$  **עם** **עודף**  $v - s$  **עם**  $\text{excess}(f, P) > 0$  **ו- סתירה**  $\text{min}(\text{excess}(f, P), f((v, u))) > 0$

**סיכום - שני המקרים (א ו-ב)** **נקבל שבקיובל של**  $\text{val}(f)$  **הוא**  $c(S, V \setminus S)$  **ונקבל**

$$\text{val}(f) = f(S, V \setminus S) - f(V \setminus S, S) = \sum_{((u,v))} f(u, v) - \sum_{v,u} f(v, u) = \sum_{u,v} c(u, v) = c(S, V \setminus S)$$

**טענה:** אם כל ערכי  $c(u, v)$  שלמים או קיימת זרימה מקסימלית עם ערכי  $f(u, v)$  שלמים

הוכחה: תהי  $f$  זרימה עם  $val(f)$  גדול ביותר מבין הזרימות עם ערכים  $f(u, v)$  שלמים  $excess(f, P) > 0$  עם  $P = s - t$ , ? לא! הגדרת העודף היא מינימום של שאליה האם קיים מסלול  $e$  מ- $s$  ל- $t$  עם  $excess(f, P) < 0$ . אם אין אף גדייר  $f'$  כמו בטענה ונתקבל זרימה שלמה יותר טובה.  $min_e c(e) - f(e) \geq 1$  נובע ש- $f'$  מקסימלית.

### 11.1.1 אלגוריתם פורד פלקרטון

התוכנית: בשלבים כלליים

1. **натхיל מזרימת האפס**  $f(e) = 0$  לכל  $e$

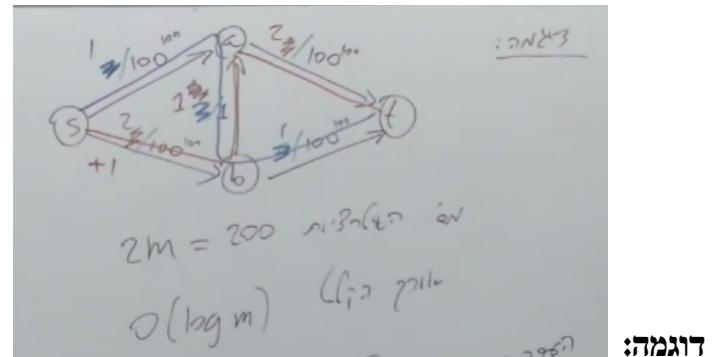
2. **נחפש מסלול  $s - t$  עם עודף חיובי** [ $O(E)$ , פרטיים בהמשך]

3. **נגדייר חדש  $f'$  יותר טובה**  

$$f'(e) = \begin{cases} f(e) & e \notin P \\ f(e) + excess(f, p) & e \text{ with the flow} \\ f(e) - excess(f, p) & e \text{ against the flow} \end{cases}$$

4. אם אין כזה מסלול נחוץ, ולפי המשפט  $f$  זרימה מקסימלית

אם כל הקיבולים  $c(u, v)$  שלמים,  $1 \leq x \leq m$  בכל שלב, האלגוריתם יעצור.  
ולכן יהיו לכל היוטר  $val(f)$  איטרציות והסיבוכיות  $O(|E| \cdot val(f))$



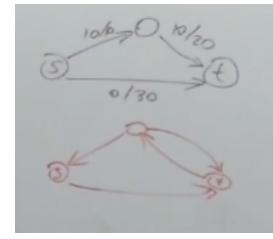
**הערה:** גם אם כל הקיבולים מס' רציונליים, האלגוריתם עוצר ומהזיר זרימה מקסימלית

**הערה:** אם יש קיבולים אי רציונליים אז לא בהכרח.

**נתון:** רשת זרימה  $G$  עם זרימה  $f$ . איך מוצאים מסלול עם עודף חיובי?

$$\text{גדייר גרף מכובן } G_f = (V, E_f) \quad (\text{גרף המסלולים})$$

$$E_f = \{\vec{e} : e = u, v \in E, f(e) < c(e)\} \cup \{\overleftarrow{e} : e = u, v \in E, f(e) > 0\}$$



דוגמה:

טענה: קיים מסלול  $s - t$  ב- $G_f$  עם  $excess(f, P) > 0$  אם ורק אם יש מסלול מכובן מ- $s$  ל- $t$ .

האלגוריתם: דומה למקודם

1. начало  $f(e) = 0$  לכל  $e$

2. нахождение пути  $s - t$  - строим дерево шириной поиска  $BFS$  из  $s$  (корень в  $t$ )

$$f'(e) = \begin{cases} f(e) & e \notin P \\ f(e) + excess(f, p) & e \text{ with the flow} \\ f(e) - excess(f, p) & e \text{ against the flow} \end{cases}$$

3. обновление  $f$  יותר טובה

4. если нет пути  $s - t$ , то  $f$  и  $excess(f)$  являются максимальными

решение: адмондес-карп

נסמן את הורימות במתוך האלגוריתם  $P_1, P_2, \dots$  (כמו  $f$  למעלה רק עם אינדקסים) ואת המסלולים  $f_1, f_2, f_3, \dots$

(1) - לכל  $k$  מתקיים  $|P_{k+1}| \geq |P_k|$

(2) - אם המסלולים  $P_k$  כך ש- $e$   $\in P_k$  ו- $e$  מוכילים צלעות נגדיות  $\leftarrow e, \rightarrow e$ , אז  $|P_l| \geq |P_k| + 2$

(3) - האלגוריתם מסיים אחריו  $O(|V| \cdot |E|^2)$ . ולכן הסיבוכיות  $O(|V| \cdot |E|^2)$

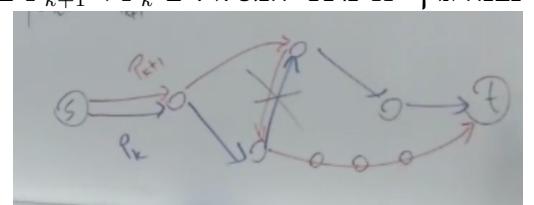
למה: כל צלעות  $\leftarrow e, \rightarrow e$  (אם יש כאלה) ששייכות ל- $P_k$  (לא בהכרח בהתאם)  $\Rightarrow$   $f_{k+1}(e) < c(e)$  ו- $f_k(e) = c(e)$  (הורימה יורדת)  $\Leftarrow$  הופעה ב- $P_k$

הוכחה: אם  $f_{k+1}(e) < c(e)$  אז  $e \in P_{k+1} \setminus G_{f_k}$  (הורימה יורדת)  $\Leftarrow$  הופעה ב- $P_k$

נשאר כתרגיל את הצד השני אבל  $e \in P_k$  עם  $\leftarrow e, \rightarrow e$  .....

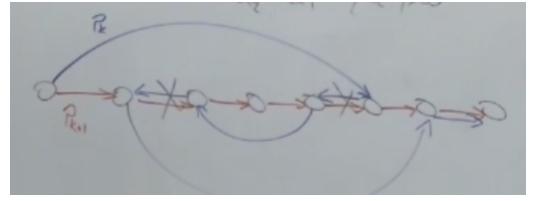
הוכחה: (של המשפט, בפועל הוא הוכיח את הוכחה עם הלמה למעלה)

נבנה גרף  $H$  משני המסלולים  $P_k$  ו- $P_{k+1}$  באופן זר



מוחקים צלעות נגדיות  $\leftarrow e, \rightarrow e$

טענה:  $H$  מכילה שני מסלולים זרים מ- $s$  ל- $t$ , נסמן אותם ב- $Q_1$  ו- $Q_2$ .



הוכחה: תרגיל  
וקיבלנו

$$2|P_k| \leq |Q_1| + |Q_2| \underset{H \text{ definition}}{\leq} |P_k| + |P_{k+1}|$$

$$|P_k| \leq |P_{k+1}|$$

הוכחנו את (1)  
נוכיח את (2): מאותו טיעונים של (1) (מה שהוא עד עכשו) אם היו צלעות נגדיות אז:

$$2|P_k| \leq |P_{k+1}| + |P_k| - 2$$

$$|P_{k+1}| \geq |P_k| + 2$$

עבור  $k, l$  כלליים מסתכלים על  $k < i \leq l$  המינימלי כך ש-  $P_k, P_i, P_l$  מכילים צלעות נגדיות  $\vec{e}, \overleftarrow{e}$ .  
(צריך למה דומה ל-  $P_i \cap P_k = \emptyset$ ).

נוכיח את (3): בכל איטרציה יש לפחות אחת שמוסרת מהגרף, מופיעה במסלול  $G_{f_k} P_k$  אבל לא בכ-  $G_{f_{k+1}}$ .  
אם  $\vec{e} \in P_{j_1} \cap P_{i_2} \cap \dots \cap P_{i_n}$  בין  $i_1 < i_2 < \dots < i_n$  כך ש-

$$|P_{i_2}| \geq |P_{j_1}| + 2 \geq |P_{i_1}| + 4$$

לכל מסלול  $\vec{e}$  הוספה מהגרף לכל היתר  $1 + \frac{|V|}{4}$  פעמים, וכך  
או יש לכל היתר  $2|E| \cdot (\frac{|V|}{4} + 1)$  איטרציות

**סיבוכיות** ( $O(|E| \cdot |V|)$ )

**12.1 אלגוריתמים ארכיטמטיים**

- חיבור חיסור כפל וחילוק (עם ו בלי שארית)

• אלגוריתם אוקלידר לחישוב  $\text{gcd}$

• בדיקת ראשוניות או פירוק לגורמים (ארטוסטנס)

• הוצאת שורש ריבועי

• פתרון משוואות לינאריות ריבועיות דיאפנטיות

הערה: אם אנחנו כותבים  $O((\log n)^c)$  אנחנו מתחווים ל-  $O(\log(\log n))$  כאשר  $c$  הוא איזשהו קבוע

2. חוג המספרים השלמים.

יהי  $n \in \mathbb{Z}$  נציג את  $n$  בספרות עשרוניות ביןarieות. אורך הייצוג  $\Theta(\log(n))$  כך ש-

- חיבור: סיבוכיות חיבור שני מספרים שהם קטנים מ- $n$ ?  $O(\log(n))$

• כפל: שזה בעצם חישוב של כפל ארוך (בשיטת הרגילה) הסיבוכיות היא  $O(\log^2 n)$  אבל בעזרה התמורה פוריה הם הראו שאפשר לבצע כפל בסיבוכיות  $O(\log n \log \log n)$  כאשר המקורי היה בסיבוכיות גדולה יותר.

- חילוק: נראה שניתן לעשות ב-  $O(\log^2 m)$

נוכיר:  $b|a$  זה יחס שאומר  $a$  מחלק את  $b$  או  $b$  מחלק ב- $a$   
לדוגמה:  $6|2$  אבל  $2|6$

(\*) המחלקים של  $b$  הם מוגדרים  $\{a : a|b, a > 0\}$ , והגורמים הם  $a \neq 1, b$  שמוכלים בקבוצת המחלקים (לדוגמה:  
המחלקים של 12 הם  $\{1, 2, 3, 4, 6, 12\}$  והוא  $a > 1$ )

משפט: לכל  $a, b \in \mathbb{Z}$  קיימת הצגה ייחודית  $b = q \cdot a + r$  כך ש-

$$\begin{aligned} q &= \lfloor \frac{b}{a} \rfloor && q \text{ quotient} \\ r &= b - a \cdot \lfloor \frac{b}{a} \rfloor && r \text{ remainder} \end{aligned}$$

**12.1.1 פריקות ייחודית של השלמים**

משפט לכל  $a > 0$  קיימת הצגה ייחודית  $a = p_1^{e_1} \cdot p_2^{e_2} \cdots p_k^{e_k}$  עבור  $p_1, \dots, p_k$  ראשוניים שונים  $< p_1 < \cdots < p_k$  ו  $0 < e_i < 1$

## 12.1.2 המחלק המשותף המקסימלי

הגדעה:

$$gcd(a, b) = \max \{d \in \mathbb{Z} : d|a, d|b\}$$

דוגמאות:

$$gcd(7, 8) = 1$$

$$gcd(400, 1000) = 200$$

$$gcd(0, 20) = 20$$

חלוקת מספר חילק את 0 ולכון לוקחים את המקסימום של המחלקים של 20.

הגדעה:  $a, b$  נקראים זרים אם  $gcd(a, b) = 1$

טענה: מתקיים השוויון (אמרנו שמשנous זה בסדר)

$$gcd(a, b) = gcd(b, a) = gcd(-b, a) = gcd(|a|, |b|)$$

טענה: אם ידוע לפירוק גורמים ראשוניים  $a = p_1^{e_1} \cdot p_2^{e_2} \cdots p_k^{e_k}$  ו  $b = p_1^{f_1} \cdot p_2^{f_2} \cdots p_k^{f_k}$  אז  $gcd(a, b) = p_1^{\min(e_1, f_1)} \cdots p_k^{\min(e_k, f_k)}$

(אחרת זה אותו מספר)

ומתקיים

$$gcd(a, b) = p_1^{\min(e_1, f_1)} \cdots p_k^{\min(e_k, f_k)}$$

דוגמה: לטענה למעלה

$$gcd(300, 72) = gcd(3 \cdot 2^2 \cdot 5^2, 3^2 \cdot 2^3 \cdot 5^0) = 3^1 \cdot 2^2 \cdot 5^0 = 3 \cdot 4 = 12$$

**טענה:** הטענות הבאות נכונות עבור  $\gcd$

$$d|a \wedge d|b \implies d|\gcd(a, b)$$

$$\gcd(a \cdot c, b \cdot c) = c \cdot \gcd(a, b)$$

$$a|b \cdot c, \gcd(a, b) = 1 \implies a|c$$

$$\gcd(a, b) = \gcd(a, c) = 1 \implies \gcd(a, bc) = 1$$

$$\text{lcm}(a, b) := \min \{c : a|c, b|c\} = \frac{a \cdot b}{\gcd(a, b)}$$

**טענה:** ו-0. מחלק עם שארית . $a \neq 0 \wedge a, b \in \mathbb{Z}$  ואז  $b = qa + r$

$$\gcd(a, b) = \gcd(r, a)$$

**הוכחה: נסמן**  $g' = \gcd(a, r)$  ו-  $g = \gcd(a, b)$  או מתקיים לפני הגדרה

$$g|a, g|b \Rightarrow g|(b - q \cdot a) = g|r \leftarrow \text{linear combination}$$

$$\Rightarrow g|a, g|r \Rightarrow g|\gcd(a, r) = g|g'$$

$$\Rightarrow g'|a, g'|r \Rightarrow g'|(aq + r)$$

$$\Rightarrow g'|a, g'|b \Rightarrow g'|\mathbf{g}$$

$$\implies g = g'$$

### 12.1.3 *algoritm来找最大公约数*

**דוגמה:** נפעיל את הטענה שוב ושוב ונקבל

$$gcd(51, 81) = gcd(30, 51) = gcd(21, 30) = gcd(9, 21) = gcd(3, 9) = gcd(0, 3) \underset{\text{stop cause we get zero}}{=} 3$$

**האלגוריתם:** נרשום את זה בpython

```

1 def gcd(a,b):
2     while a!=0:
3         a,b=b%|a|,a #that line do r=b%a , b=a,a=r
4     return b

```

**סיבוכיות:** נניח  $n < a < b < 0$  מהי הסיבוכיות?

**טענה:**  $gcd$  עושה לכל היותר  $2\log(n)$  איטרציות

$$r < \frac{1}{2}b$$

נחלק למקרים:

$$1. \text{ אם } r \leq \frac{1}{2}b \text{ אז מתקיימים}$$

$$2. \text{ אם } r = b - a < b - \frac{1}{2}b = \frac{1}{2}b \text{ (סוף הוכחה הטענה)}$$

מזההנו, במהלך האלגוריתם  $a_i \cdot b_i$  קטנה לפחות פי 2 בכל איטרציה, ולכן מס' האיטרציות לכל היותר

$$\log_2(n^2) = 2\log_2 n$$

$$a_i \cdot b_i < \frac{1}{2}a_{i-1} \cdot b_{i-1} \cdot a_0 b_0 < n^2$$

בכל איטרציה  $O(\log^2 n)$  ולכון  $O(\log^3 n)$ . ניתן יותר זהיר לראות  $O(\log^2 n)$  אפשר קרוב ל-

**שאלה:** האם מספר האיטרציות הוא  $\Theta(\log n)$ ?

מספר פיבונאצ'י  $F_{n+2} = F_{n+1} + F_n$  ומתקיים

$$F_{n+2} \% F_{n+1} = F_n \longrightarrow F_n \sim \phi^n \quad \phi = \frac{1 + \sqrt{5}}{2}$$

### 12.1.4 *האלגוריתם אוקלידס המורחב*

**טענה:** לכל  $\mathbb{Z}$  ו-  $a, b \in \mathbb{Z}$  קיימים  $x, y \in \mathbb{Z}$  כך ש-

$$gcd(a, b) = ax + by$$

**הוכחה: ראשית נראה עבור**  $a = 0$

$$\gcd(a, b) = b = 0 \cdot a + 1 \cdot b$$

ואם  $b = 0$  כנ"ל.

או נשאר רק להניח (בה"כ)  $0 < a \leq b$ , אז nociah באינדוקציה על  
ニזcker שנאחנו יכולים להביע b בצורה הבאה : צעד האינדוקציה:

$$\gcd(a, r) = rx' + ay' \text{ for } x', y' \in \mathbb{Z}$$

$$\Rightarrow \gcd(a, b) = \gcd(r, a) = rx' + ay' = (b - aq)x' + ay' = (y' - qx')a + x' \cdot b \underset{x:=y'-qx' \atop y=x'}{=} xa + yb$$

### האלגוריתם: גרסה רקורסיבית

```

1 def extended_gcd(a,b):
2     if a==0:
3         return (b,0,1) #return (b,x,y)
4     (q,r)=divmod(b,a)#insert the result to q and remainder part to r
5     (d,x,y)=extended_gcd(r,a)
6     return (d,y-q*x,x)

```

תרגיל: כתבו גרסה איטרטיבית, עם  $O(1)$  מספרים בזיכרון.

### דוגמה: שימוש באלגוריתם

$$\begin{aligned} \gcd(111, 243) &\underset{q=2 \text{ return}(3,16,-3)}{=} \gcd(21, 111) \underset{q=5 \text{ return}(3,-3,1)}{=} \gcd(6, 21) \underset{q=3 \text{ return}(3,1,0)}{=} \gcd(3, 6) \underset{q=2 \text{ return}(3,0,1)}{=} \gcd(0, 3) = \\ &\cdot (d, x, y) = (3, -35, 16) \quad (d, x, y) \end{aligned}$$

$$\gcd(a, b) = 3 = -35 \cdot 111 + 16 \cdot 243 = ax + by$$

## 12.2 חשבון מודולרי

יהי  $n > 0$

נדיר: יחס  $a \% n = b \% n$  אם  $a \equiv b$   
 $a \equiv_k b$

לדוגמה:  $7 \equiv 22 \pmod{5}$ ,  $a \equiv b \pmod{n}$

הוּא יָסֵד שְׁקִילוֹת: אוסף מחלקות השקילות הוא  $\mathbb{Z}_n$ , לדוגמה:  $\mathbb{Z}_6 = \{\bar{0}, \bar{1}, \bar{2}, \bar{3}, \bar{4}, \bar{5}\}$

$$\bar{x} = x + 6\mathbb{Z} = \{x + 6y : y \in \mathbb{Z}\}$$

$$example : \bar{4} = \{\dots, -2, 4, 10, 16, \dots\}$$

חיבור:  $\bar{x} + \bar{y} = \overline{x+y}$  להשלים!!

### 12.2.1 החבורה הכפליות $(\mathbb{Z}_n^*, \cdot)$

הגדרה: נאמר ש- $a$  הוא הפיך אם קיים  $b \in \mathbb{Z}_n$  כך ש- $a \cdot b \equiv 1$

$$\mathbb{Z}_n^* = \{a : a \in \mathbb{Z}_n \text{ reversible}\}$$

דוגמאות: 1. תמיד הפיך, 2. תמיד הפיך כי  $1 \cdot (-1) \equiv 1$  ו-0 אף פעם לא הפיך.

$$3 \in \mathbb{Z}_{10} \text{ הפיך כי } 3 \cdot 7 \equiv 1_{10}$$

$$4 \cdot 5 \equiv_{10} 0 \neq 1 \quad 4, 5 \in \mathbb{Z}_{10} \text{ לא הפיכים כי}$$

$$\mathbb{Z}_{10}^* \equiv \{1, 3, 7, 9\}$$

טענה:  $(\mathbb{Z}_n^*, \cdot)$  היא חבורה כי היא מקיימת את האקסיום של חבורה (שהן):

$$(a \cdot b)c \equiv a(bc) \quad .1$$

$$ab = ba \quad .2$$

$$a \cdot 1 \equiv a \quad .3$$

$$a \cdot b \equiv 1 \quad .4$$

$$(a \cdot b)^{-1} \equiv a^{-1} \cdot b^{-1} \quad \text{וגם מתקיים } a \cdot b \in \mathbb{Z}_n^* \iff a, b \in \mathbb{Z}_n^*$$

נפunning:  $\varphi(10) = |\mathbb{Z}_{10}^*| = |\{1, 3, 7, 9\}| = 4$  לדוגמא:  $\varphi(n) := |\mathbb{Z}_n^*|$

טענה:  $a$  הפיך ב- $\mathbb{Z}_n$   $\iff \gcd(a, n) = 1$

הוכחה:  $a \cdot x \equiv 1 \pmod{n}$  או מתקיים  $y \cdot a = 1$  אז  $x$  הופכי של  $a$  כי  $\gcd(a, n) = 1 \iff d = 1$   $d|1$   $d|n$   $d|a$ .  $ax = 1 + ny$ ,  $1 = ax - ny \iff ax \equiv 1$   $x \in \mathbb{Z}_n$   $\iff a$  הפיך  $\iff \gcd(a, n) = 1$

הערה:  $d = ax + by$  הקטן ביותר שאפשר לכתוב  $d \geq 0$  הוא  $d = \gcd(a, b)$

טענה: בהינתן  $b, a$ , איך מחשבים  $b$  כך ש-?

```

1 def inverse(a,n):
2     d,x,y=gcd(a,n)
3     if d not equal 1: return None
4     return x

```

שאלה: (سان צו', המאה ה-3 לספרה)

יש מספר לא ידוע של דברים מסוימים. כטופרים אותם בשלשות, נשארים שניים.

כטופרים אותם בחמשיות נשארים שלושה.

כטופרים אותם בשבעיות, נשארים שתים.

כמה דברים ישנו?

תשובה: נבין מה שאלה שמתקיים

$$x \equiv 3 \pmod{3}, x \equiv 3 \pmod{5}, x \equiv 2 \pmod{7}$$

או מתקיים מהלgorיתם  $x = 23 + 105k$ , או יותר דיק  $x = 23 + 23 + 105k$  פתרון לכל  $k$ .

### 12.2.2 משפט השאריות הסיני

הדגמה של האלגוריתם (נשים לב  $a \cdot b^{-1} \pmod{n}$  זו ההופכי):

$$x = 3 \cdot 5 \cdot ((3 \cdot 5)^{-1} \pmod{7}) \cdot 2 + 3 \cdot 7 \cdot ((3 \cdot 7)^{-1} \pmod{5}) \cdot 3 + 5 \cdot 7 ((5 \cdot 7)^{-1} \pmod{3}) \cdot 2$$

$$= 3 \cdot 5 \cdot 1 \cdot 2 + 3 \cdot 7 \cdot 1 \cdot 3 + 5 \cdot 7 \cdot 2 \cdot 2 = 30 + 63 + 140 = 233 \equiv 23 \pmod{105}$$

המשפט: יהי  $N = n_1 \cdot n_2 \cdots n_k$  כאשר  $n_i$  זרים זה לזה.

יש פתרון ב- $\mathbb{Z}$ , והוא היחיד מודולו  $N$  והנוסחה:  
או למערכת המשוואות

$$\begin{cases} x \equiv a_1 \pmod{n_1} \\ x \equiv a_2 \pmod{n_2} \\ \vdots \\ x \equiv a_k \pmod{n_k} \end{cases}$$

$$X = \sum_{i=1}^k \frac{N}{n_i} \cdot \left( \left( \frac{N}{n_i} \right)^{-1} \pmod{n_i} \right) \cdot a_i + \sum_{\forall k \in \mathbb{Z}} N$$

נסתכל על  $X \pmod{n_i}$

$$X \equiv_{n_i} \frac{N}{n_i} \cdot \left( \left( \frac{N}{n_i} \right)^{-1} \pmod{n_i} \right) \cdot a_i \equiv_{n_i} a_i$$

**הוכחה:** להשלים.

**משפט:** העתקה

$$F : \mathbb{Z}_{n_1 \cdot n_2 \cdots n_k} \rightarrow \mathbb{Z}_{n_1} \times \mathbb{Z}_{n_2} \times \cdots \times \mathbb{Z}_{n_k}$$

**המוגדרת** ( $F(x) = (x \bmod n_1, x \bmod n_2, \dots, x \bmod n_k)$ ) **היא איזומורפיזם של חוגים.** (חח"ע ועל)

**הוכחה:** הייתה סוג של בעל פה, בעיקר מתכונות של חוגים וחברות

**דוגמה:** מתקיים:

$$\mathbb{Z}_6 \cong \mathbb{Z}_2 \times \mathbb{Z}_3 = \{(a, b) : a \in \mathbb{Z}_2, b \in \mathbb{Z}_3\}$$

$$0 \rightarrow (0, 0)$$

$$1 \rightarrow (1, 1)$$

$$2 \rightarrow (0, 2)$$

$$3 \rightarrow (1, 0)$$

$$4 \rightarrow (0, 1)$$

$$5 \rightarrow (1, 2)$$

ומתקיים

$$F(1) + F(4) = (1, 1) + (0, 1) \underset{\mathbb{Z}_2 \times \mathbb{Z}_3}{=} (1, 2) = F(1 + 4)$$

$$F(5) + F(2) = (1, 2) + (0, 2) = (0, 1) = F(2 \cdot 4)$$

$$\begin{aligned}
 & \text{גפ } \gcd(x, n_i) = 1 \quad \forall i \quad \text{גפ } \gcd(x, N) = 1 \quad \boxed{N = n_1 \cdots n_k} \\
 & \text{גפ } x \in \mathbb{Z}_{n_i}^* \quad \forall i \quad x \in \mathbb{Z}_N^*
 \end{aligned}$$

$$\begin{aligned}
 F : \mathbb{Z}_N^* & \longrightarrow \mathbb{Z}_{n_1}^* \times \mathbb{Z}_{n_2}^* \times \cdots \times \mathbb{Z}_{n_k}^* \\
 \varphi(n) = |\mathbb{Z}_N^*| &= \left| \mathbb{Z}_{n_1}^* \times \cdots \times \mathbb{Z}_{n_k}^* \right| = \\
 &= \prod_i |\mathbb{Z}_{n_i}^*| = \prod_i \varphi(n_i)
 \end{aligned}$$

$$\varphi(n) = \varphi(p_1^{e_1} p_2^{e_2} \dots p_k^{e_k}) =$$

$$= n \cdot \prod_{i=1}^k \left(1 - \frac{1}{p_i}\right)$$

$$= \frac{n}{\log \log n}$$

$$\varphi(120) = \varphi(2^3 \cdot 3 \cdot 5) = 120 \cdot \left(1 - \frac{1}{2}\right) \left(1 - \frac{1}{3}\right) \left(1 - \frac{1}{5}\right)$$

$$= 120 \cdot \frac{8}{30} = 32$$

(G)  $\varphi(n)$

$$\varphi(p) = p - 1 = p^{(1-\frac{1}{p})} \quad \text{wegen } n=p \quad \text{zu 25}$$

$$\varphi(p) = p^e - p^{e-1} \quad \text{wegen } n=p^e \quad \text{zu 25}$$

$$= p^e \left(1 - \frac{1}{p}\right)$$

$$\varphi(3^2) = |\mathbb{Z}_7^*| = |\{1, 2, 4, 5, 7, 8\}|$$

$$\varphi(n) = p^{(1-\frac{1}{p})} q^{(1-\frac{1}{q})} \quad \text{wegen } n=pq \quad \text{zu 25}$$

$$= n \left(1 - \frac{1}{p}\right) \left(1 - \frac{1}{q}\right) \neq n - 1$$

### 13.1 המשך חשבון במודולרי

הערה: אפשר להחזיר על החומר ע"י השאלות שאלו לאתר - בעיקר ברמת הבנה.

הערה: ראיינו ש-

חיבור וחיסור אפשר לעשות -  $\log n$

כפל, חילוק ומודולו<sup>2</sup> ( $\log n$ )

חזקת מודולו  $n$  ( $\log n$ )<sup>3</sup>

הוצאת שורש של  $n$  ( $\log n$ )<sup>3</sup> - תרגיל!

### 13.2 בדיקת ראשוניות

בעיה: נתון  $n \in \mathbb{Z}$ . האם  $n$  ראשוני?

הצעה: אלגוריתם נאיבי ראשון:

```

1 def is_prime(n):
2     for a in range(2,n):
3         if n%a==0:
4             return false
5     return true
6

```

בסיבוכיות  $O(n \cdot \log^2 n)$

ניסיון שיפור - אם הוא פrisk איז מתקיים או אחד הם בודאות מקיים  $\lfloor \sqrt{n} \rfloor$  ונקבל  $O(\sqrt{n} \cdot \log^2 n) = n_1 \cdot n_2$  וזה לא באמת שיפר אותו ממספרים.

משפט: משפט המספרים הראשוניים

$$\Pi(n) = |\{p : p \leq n, p \text{ prime number}\}|$$

$$\Pi(n) \sim \frac{n}{\log n}$$

למשל אם הגרנו 1000 מספרים עם 100 ספרות, "כנראה" יהיה שם ראשוני.

הערה: השתמש בתכונות של  $\mathbb{Z}_p$  כדי לזהות לא ראשוניים

### תכונה 1: משפט פרמה

אם  $p$  ראשוני, אז  $a \in \mathbb{Z}$  ו-  $a^p \equiv a \pmod{p}$  ניסוח שקול למשפט -  $a \in \mathbb{Z}_p^*$  שווה תאcls אומר  $a^{p-1} \equiv 1 \pmod{p}$ . (משפט לגראנז)

דוגמה: להלן-

$$\begin{aligned} (\pm 1)^6 &\equiv 1 \equiv 1 \\ (\pm 2)^6 &\equiv 64 \equiv 1 \\ (\pm 3)^6 &\equiv 729 \equiv 1 \end{aligned}$$

משפט: אוילר - הערכה, לא ממש בחומר

$$a \in \mathbb{Z}_n^* \text{ לכל } a^{\phi(n)} \equiv 1$$

הוכחה: משפט פרמה

כל  $a \neq 0$  ב-  $\mathbb{Z}_p$  הוא הפיך.  
הפונקציה  $f(x) \equiv ax$ ,  $f : \mathbb{Z}_p \rightarrow \mathbb{Z}_p$  חד-ע. ועל.

$$(p-1)! \equiv 1 \cdot 2 \cdots (p-1) \stackrel{\equiv_p}{\substack{\text{same numbers} \\ \text{different order}}} a \cdot 2a \cdot 3a \cdots (p-1)a \equiv_p a^{p-1} \cdot 1 \cdot 2 \cdots (p-1)$$

(דוגמה :  $p = 7$   $a = 2 \equiv 2 \cdot 2 \cdot 3 \cdot 4 \cdot 5 \cdot 6 \equiv_7 2 \cdot 4 \cdot 6 \cdot 1 \cdot 3 \cdot 5$  כאשר 1 נכפיל בהופכיים של 1, 2, 3, ...)

תרגיל: חשבו את  $(p-1)! \pmod{p}$

תכונה 2: אם  $p > 2$  ראשוני אז למשוואת  $x^2 \equiv_p 1$  יש בדיק שני פתרונות  $x$  ב-  $\mathbb{Z}_p$

הוכחה: נראה שם:

$$x^2 \equiv 1 \pmod{p}$$

$$\Rightarrow (x-1)(x+1) \equiv 0 \pmod{p}$$

$$\Rightarrow (x-1)(x+1) = pk \quad k \in \mathbb{Z}$$

מפריקות יחידה נובע  $x \equiv -1 \pmod{p}$ ,  $x \equiv 1 \pmod{p}$ ,  $(x-1)|p$  או  $(x+1)|p$

הערה: אם  $n_1, n_2$  זרים. אז יש יותר פתרונות ל-

$$a \equiv \pm 1 \pmod{n_1}, b \equiv \pm 1 \pmod{n_2}$$

(יש 4 אופציות) משפט השאריות הסיני  $x \equiv b \pmod{n_2}, x \equiv a \pmod{n_1}$ ,  $x^2 \equiv 1 \pmod{n_1 \cdot n_2}$ , כי הוא מודולו  $n_1$  ומודולו  $n_2$ .

**הצעה:** מבחן פרמה נתון  $n$  או לכל  $a \in \mathbb{Z}_n$  לבדוק אם  $a^n \equiv a \pmod{n}$  או לא נגיד ש- $n$  פריק.  
בסוף, אם לא אמרנו פריק, נניח ראשוני

**בעיה:** לא יעיל ולא נכון (אם הוא פריק או אכן פריק, אבל אם יגיד שהוא ראשוני הוא לא בהכרח כי משפט פרמה הוא לא משפט אמת).

**טענה:**  $x^{561} = x \pmod{561}$   $x \in \mathbb{Z}_{561}$ . לכל  $561 = 3 \cdot 11 \cdot 17$ .

**הוכחה:** נשים לב ש-  $2|560$  ו-  $10|560$  ו-  $16|560$  (שהם מינוס אחד הגורמים)  
ניתן לעשות

$$x^{561} - x \equiv_{17} x^{16 \cdot 35} \cdot x - x \equiv_{17} 0$$

באותו אופן  $x^{561} - x \equiv_{11} 0$  ו-  $x^{561} - x \equiv_3 0$   
 $x \in \mathbb{Z}_{561}$   $x^{561} - x \equiv_{561} 0$  לכל

**הגדרה:**  $n$  הוא מספר קרמייקל (*carmichael*)  
אם  $a^n \equiv a \pmod{n}$  אבל  $a \in \mathbb{Z}_n$ .

**דוגמאות:** .....561, 1105, 1729

**הערה:** יש  $\infty$  מספרים קרמייקלים, אבל יש  $(n)$  בין 1 ל- $n$  (רובם לא כאליה) אבל  $\Omega(n^{\frac{1}{3}})$ .

**הערה:** יש משפט שאומר ש- $n$  הוא מספר קרמייקל  $\iff$  הוא מכפלה של ראשוניים כך ש-  $|n-1|$ หาร  $n-1$ .

**הצעה:** משופרת

אם  $1 \equiv a^{n-1}$ ,  $a^{\frac{n-1}{2}} \equiv 1$ ,  $a^{\frac{n-1}{2k}} \equiv 1$ , נסתכל על  $(a^{\frac{n-1}{2}})^2 \equiv 1$ ,  $a^{\frac{n-1}{2}} \equiv \pm 1$   
אם הוא לא  $\pm 1$  אז  $n$  פריק.

\* זה עובד עבור  $n=561$ . יש הרבה  $a$  עם  $a^{280} \not\equiv \pm 1$   
\* לא עובד עבור  $a^{865} \equiv 1$  ש-  $a=864$ ,  $12|864$ ,  $18|864$  כי  $7 \cdot 13 \cdot 19 = 1729$  ו-  $\frac{1729-1}{2} = 864$  כך ש-  $a^{865} \equiv 1$ .

**הצעה:** משופרת 2

נסתכל על המספרים  $a^{\frac{n-1}{2k}}, a^{\frac{n-1}{4}}, \dots, a^{\frac{n-1}{2k}}$  כאשר  $\frac{n-1}{2k}$  אי זוגי.  
אם יש  $1 \equiv a^{\frac{n-1}{2^i}}$  אז  $n$  פריק. כי  $a^{\frac{n-1}{2^i}} \not\equiv \pm 1$ ,  $a^{\frac{n-1}{2^i}} \equiv 1$ .

**דוגמה:** יש הרבה  $a$  כך ש-  $a^{108} \equiv 1$  אבל  $a^{54} \not\equiv 1$

**סיכום:** נתון  $n$  אי-זוגי, לכל  $a$   
1. חשב את  $a^{\frac{n-1}{2k}}, a^{\frac{n-1}{4}}, \dots, a^{\frac{n-1}{2k}}$  (כasher  $\frac{n-1}{2k}$  אי זוגי)  
2. אם  $1 \not\equiv a^{n-1}$  או נזכיר  $n$  פריק

3. אם  $a^{\frac{n-1}{2^i}} \not\equiv \pm 1 \pmod{n}$  אז נזכיר  $n$  פריך  
 $\star = 3$  השלבים הראשונים הם  $O(\log^3 n)$   
 אם לא הכרזנו פריך, נזכיר ראשוני!

שאלה: אולי יש הרבה  $a$  שמעמידים על פריקות?  
 אכן, אולי מספיק לנסות  $a$ ים כדי למצוא אחד כזה?  
 (מילר 1976) יש כזה  $n < 70\log^2 n$  בהנחת השערת רימן המורחבת.  
 (רבין 1980) רוב  $a$ -ים הם  $\star$  יגידו  $\star$  פריך, לפחות  $\frac{n-1}{2}$   
 עבור  $n$  פריך, נגריל 100  $a$ -ים בלחתי תלויים ונבדוק את  $\star$  (ביסיכום)  
 אם הוכרזו  $\star$  פריך  $\Leftarrow$  פריך  
 אם לא, ננחש ראשוני.

$$P[\text{announce prime} | n \text{ Composite number}] \leq \frac{1}{2}$$

$$P[\text{announce prime} | n \text{ Composite number}] \leq \frac{1}{2^{100}}$$

הערה: בשנת 2004 התגלה אלגוריתם דטרמיניסטי בזמן  $O(\log^{12} n)$

טענה: אם  $n$  פריך אז יש לפחות  $\star$  יגידו  $\star$  פריך.  
מקרה א: (לא קרמייקל)  
 אם קיימים  $a \in \mathbb{Z}_n^\star$  כך  $a^{n-1} \not\equiv 1$ . נגידיר:

$$G = \{x \in \mathbb{Z}_n^\star : x^{n-1} \equiv 1 \pmod{n}\} \subseteq \mathbb{Z}_n \setminus \{0\}$$

נשים לב שלכל  $x \in G$ ,  $x^{n-1} \equiv 1 \pmod{n}$ . למה? נחשב  $x^{n-1} \cdot 1 \not\equiv 1$ .  $ax \notin G$ .  
 הפונקציה  $f(x) = ax$  ל $\mathbb{Z}_n^\star$  היפה מ- $\mathbb{Z}_n^\star$  ל-

$$|G| = |\{f(x) : x \in G\}| \leq (n-1) - |G|$$

$$|G| \leq \frac{n-1}{2}$$

לכן יש לפחות  $\frac{n-1}{2}$   $a$ -ים שלפיהם נזכיר פריך בבדיקה 1.  
מקרה ב: (קרמייקל)

אם קיימים  $a \in \mathbb{Z}_n^\star$ , כלומר לכל  $a \in \mathbb{Z}_n^\star$  כך  $a^{n-1} \equiv 1$   
 במקרה זה  $n = n_1 n_2$  זרים [nociah בסוף]  
 $\left(-1\right)^{\frac{n-1}{2^k}} \equiv -1$ .  
 יהי  $i \in \{1, \dots, k\}$  המינימלי כך שקיים  $b$  עם  
 נגידיר:

$$H = \left\{ x \in \mathbb{Z}_n^\star : x^{\frac{n-1}{2^i}} \equiv \pm 1 \pmod{n} \right\} \subseteq \mathbb{Z}_n \setminus \{0\}$$

לכל  $a \in \mathbb{Z}_n^{-1}$  האלגוריתם יርיז פריק כי  
נמצא

$$a \in \mathbb{Z}_n \xleftarrow{\text{chinese}} \begin{cases} a \equiv b \text{mod}_{n_1} \\ a \equiv 1 \text{mod}_{n_2} \end{cases}$$

$$a^{\frac{n-1}{2^i}} \equiv 1 \text{ mod}_{n_2}$$

$$a^{\frac{n-1}{2^i}} \equiv -1 \text{ mod}_{n_1}$$

טענה: אם היה  $a^{\frac{n-1}{2^i}} \not\equiv \pm 1$  סתירה למודול  $n_2$ , אם היה  $-1$  סתירה למודול  $n_1$ .  
כמו קודם, היא פונקציה חד-עגל על איברים  $\mathbb{Z}_n^*$  ( $a$  הפיך).  
אם  $x \in H$  אז  $ax \notin H$ , כי  $(ax)^{\frac{n-1}{2^i}} \equiv_n a^{\frac{n-1}{2^i}} x^{\frac{n-1}{2^i}} \not\equiv_n \pm 1$   
כמו קודם

$$|H| \leq (n-1) - |H|$$

$$|H| \leq \frac{n-1}{2}$$

ולכן יש לפחות  $\frac{n-1}{2}$  עברים האלגוריתם יגיד פריק.

טענה: נשאר להוכיח שם  $a^{n-1} = p^d$  חזקת ראשוני או יש  $a$  הפיך כך ש- $1 \not\equiv a^n$  (ואז אנחנו במקרה א')

נראה ש- $a^n \not\equiv 1$

הוכחה: נסתכל על  $(p+1)$

$$(p+1)^{p^d} \equiv \sum_1^{p^d} \binom{p^d}{k} p^k \equiv 1 + \binom{p^d}{1} p + \binom{p^d}{2} p^2 + \dots$$

$$\equiv 1 \text{ mod}_{p^d=n}$$

$$\equiv 1 \text{ mod}_{p^2}$$

$$\Rightarrow \not\equiv p+1 \text{ mod}_{p^2} \setminus \text{mod}_{p^d}$$