



מבני נתונים 08a

תרגול 15

27/3/2008

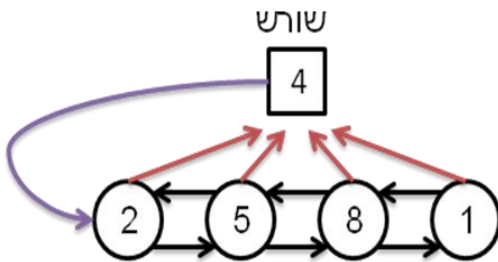
Union find + הופמן

ליאור שפירא

שאלה ב-union find

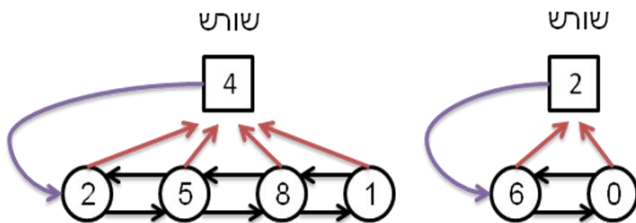
□ להלן מבנה נתונים ל-union find:

- כל קבוצה תוחזק ברשימה משורשרת דו-כיוונית, כל אלמנט מצביע לצומת מיוחד (שורש) המייצג את הקבוצה

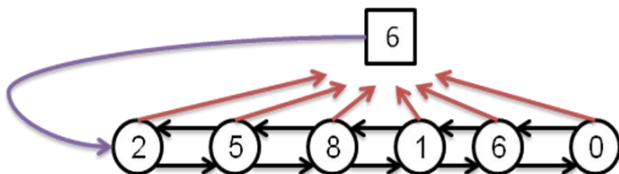


- Find מקבלת מצביע ומחזירה מצביע לשורש

- Union מוסיפה קב' קטנה לגדולה ומעדכנת את השורש



דוגמה ל-union:



שאלה ב-union find

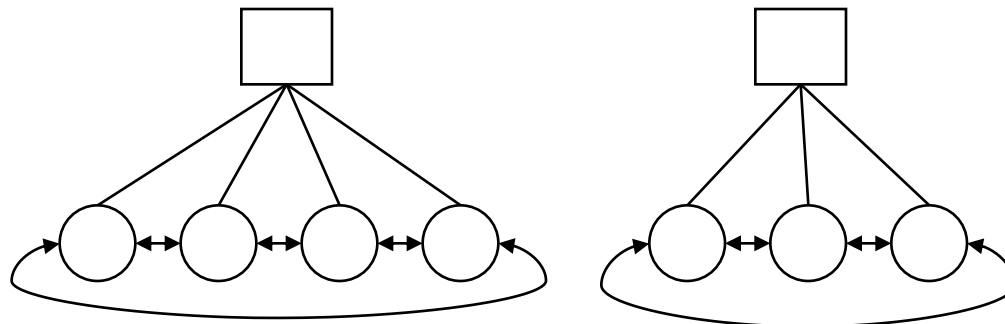
□ כמה זמן תיקח סדרת m פעולות כאשר נתחיל מאוסף של n קבוצות בעלות אבר אחד (כל אחת). הנח $n > m$

□ תשובה

■ נתבונן בסדרה כלשהיא בת m פעולות

■ כל פעולת find לוקחת $O(1)$ ולכן סה"כ $O(m)$

■ כל פעולת union פרופורציונאלית למס' האברים שמשנים את שורש הקב' המכילה אותם



שאלה ב-union find

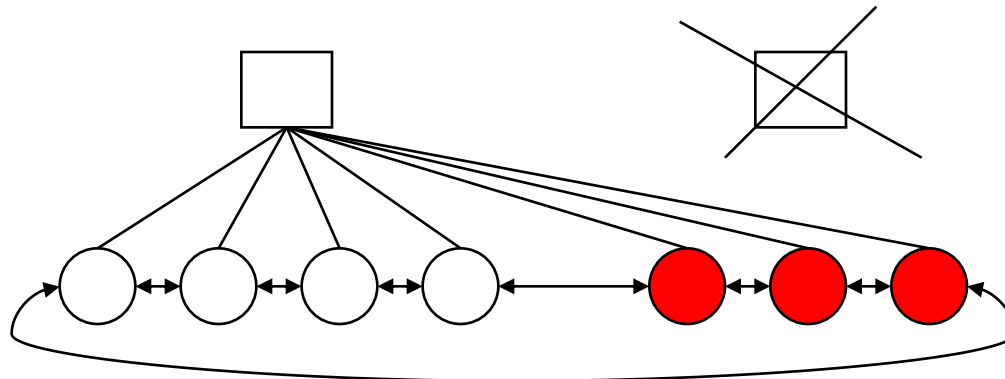
□ כמה זמן תיקח סדרת m פעולות כאשר נתחיל מאוסף של n קבוצות בעלות אבר אחד (כל אחת). הנח $n > m$

□ תשובה

■ נתבונן בסדרה כלשהיא בת m פעולות

■ כל פעולת find לוקחת $O(1)$ ולכן סה"כ $O(m)$

■ כל פעולת union פרופורציונאלית למס' האברים שמשנים את שורש הקב' המכילה אותם



שאלה ב-union find

□ כמה זמן תיקח סדרת m פעולות כאשר נתחיל מאוסף של n קבוצות בעלות אבר אחד (כל אחת). הנח $n > m$

□ תשובה

■ נתבונן בסדרה כלשהיא בת m פעולות

■ כל פעולת find לוקחת $O(1)$ ולכן סה"כ $O(m)$

■ כל פעולת union פרפורציונאלית למס' האברים שמשנים את שורש הקב' המכילה אותם

■ לכן אם נחסום לכל אבר את מס' הפעמים שמשתנה השורש אליו הוא מצביע ונסכם על כל האיברים, נקבל חסם על העלות הכוללת של פעולות ה-union.

המשך תשובה

- נתבונן באבר x . בכל פעם שמשתנה השורש אליו הוא מצביע, גודל הקבוצה בה x נמצא מוכפל בלפחות 2. מאחר וגודל קב' קטן שווה לח מס' הפעמים שמצביע השורש של x יכול להשתנות חסום ע"י $\log n$
- נסכם עבור כל האברים ונקבל חסם כללי $O(n \log n)$ על העלות הכללית של פעולות ה-union.
- סה"כ אם נסכם נקבל עלות כללית $O(n \log n + m)$

חלק שני

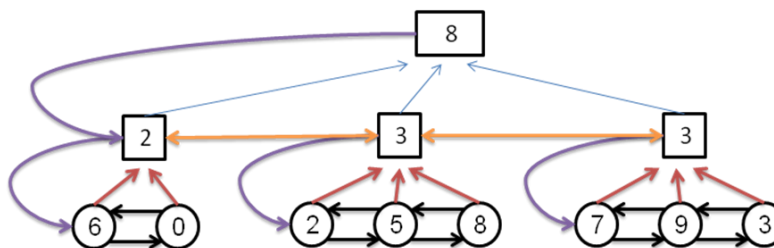
□ נסמן $k = \lfloor \log n \rfloor$

□ קבוצות שגודלן k לכל היותר נייצג כמו קודם

□ קבוצה S בגודל גדול מ- k תיוצג ע"י אוסף של t קבוצות בגודל k בדיוק (נקרא להן **מלאות**), וקבוצה אחת לכל היותר בגודל קטן מ- k (**חסרה**).

- תתי קבוצות אלו מיוצגות כמו שתואר בחלק א' של השאלה.
- שורשי תתי הקבוצות מסודרים ברשימה משורשרת דו-כיוונית ומצביעים לצומת חדש המייצג את הקבוצה כולה.
- שורש זה מכיל מצביע לרשימת שורשי תתי הקבוצות ואת גודל הקבוצה.
- הקבוצה החסרה (אם קיימת) תהיה תמיד ראשונה ברשימה.

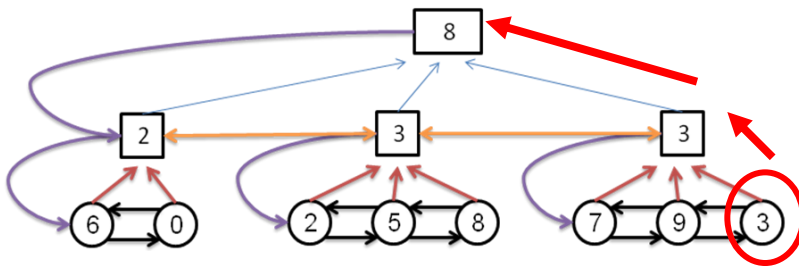
דוגמה לקבוצה גדולה עם שתי קב' מלאות ואחת חסרה, $k = \log n = 3$



איך מוגדרות הפעולות?

- Find ימצא את שורש תת הקב' המאוכסן בצומת המייצג את האבר ומשם את שורש הקבוצה הגדולה

דוגמה לקבוצה גדולה עם שתי קב' מלאות ואחת חסרה, $k=\log n=3$



- Union יתבצע באופן הבא

- שתי קב' קטנות שגודלן יחד פחות מ- k נאחד כמו בסעיף הקודם
- אם גודלן יחד גדול מ- k נאחד לקבל קב' אחת מלאה ואחת חסרה. ניצור קב' חדשה שאלה תתי הקב' שלה
- אם אחת קטנה ואחת גדולה, נאחד את הקטנה עם הקב' החסרה של הגדולה, ונוסיף את תתי הקב' המתקבלות לקב' הגדולה
- אם שתי הקב' גדולות נעביר את המלאות מהקב' עם פחות אלמנטים לקב' עם יותר אלמנטים ונאחד את החסרות

חלק שני

□ מה זמן הריצה של סדרה בת m פעולות כאשר נתחיל מאוסף של n קבוצות, כל אחת מכילה רק אבר יחיד?

□ תשובה

■ נתבונן בסדרה שרירותית בה m פעולות

■ עלות find כמו קודם – $O(m)$

■ כדי לחסום את עלות ה-union, נחסום את מס' הפעמים שאלמנט משנה את "מצביע תת קבוצה" שלו, ואת מס' הפעמים שתת קבוצה מלאה משנה את "מצביע הקבוצה" שלה

■ נתבונן באלמנט x , כל פעם שהוא משנה מצביע תת קבוצה, גודל התת קבוצה שהוא בתוכה גדל בלפחות פע 2. מאחר שגודל תת קבוצה אינו עולה על $\log n$, מס' הפעמים שא x יכול לשנות מצביע תת קבוצה חסום ע"י $\log \log n$.

■ אם נסכום על כל האברים, מספר הפעמים שאלמנט כלשהוא משנה את מצביע התת קבוצה שלו הוא $n \log \log n$.

המשך תשובה

- נותר לחסום את מספר הפעמים שתת קב' מלאה משנה את מצביע הקבוצה שלה.
- בכל פעם שתת קבוצה γ משנה את מצביע הקבוצה שלה, גודל הקבוצה בה γ נמצאת גדל בלפחות פי 2.
- מאחר וגודל קב' הוא לכל היותר n , מספר הפעמים שתת קבוצה γ משנה מצביע הקבוצה שלה חסום ע"י $\log n$.
- המספר הכולל של תת קבוצות הוא לכל היותר $n/\log n$ ולכן סה"כ שינויי מצביעי קבוצות בתת קבוצות הוא

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

- נסכם סה"כ לכל פעולות union: $O(n + n \log \log n) = O(n \log \log n)$
- נשים לב שבכל פעולת union, רק תת קבוצה חסרה אחת יכולה לשנות מצביע קבוצה ולכן סה"כ עלות זו $O(n)$.

- ולכן סה"כ עלות סדרת הפעולות: $O(n \log \log n + m)$

Prefix codes & Huffman

Prefix Codes

■ נניח א"ב של 6 תווים שכל אחד מופיע בתדירות מסוימת

■ קידוד אורך קבוע

■ אם נייצג כל תו במס' ביטים שווה נזדקק ל... 3 ביטים

■ עבור קובץ של 100000 תווים נזדקק ל... 300000 ביטים

■ קידוד אורך משתנה (variable length code)

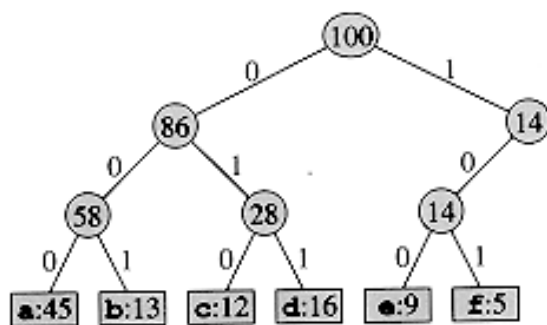
■ תווים שחוזרים הרבה נייצג במעט ביטים (וההפך)

	a	b	c	d	e	f
Freq.	45	13	12	16	9	5
Fixed encoding	000	001	010	011	100	101
Variable encoding	0	101	100	111	1101	1100

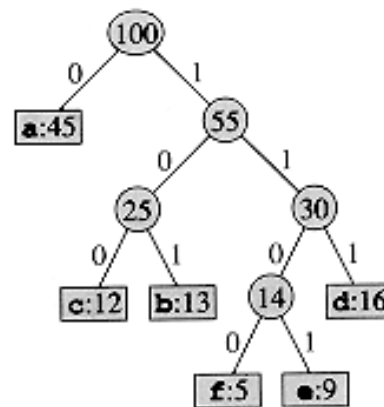
■ עבור אותו קובץ נזדקק ל... 224000 ביטים (25% חסכון)

Prefix Codes

- נגדיר קודים כך שאף קוד לא משמש כ-prefix לקוד אחר
- הדבר יקל על encoding/decoding
- ניתן לייצג קוד ע"י עץ בינארי
- קוד אופטימאלי **תמיד** מיוצג ע"י עץ בינארי מלא



(a)

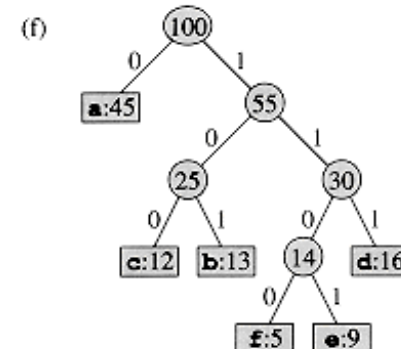
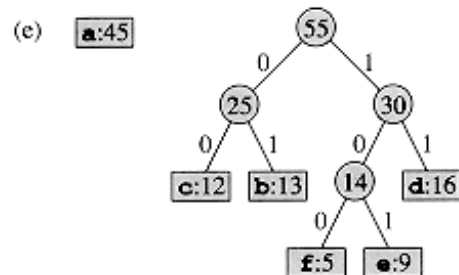
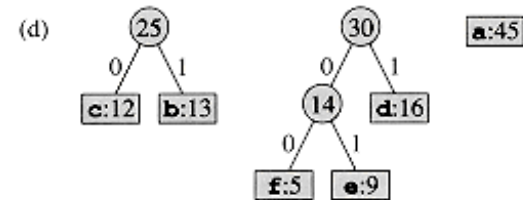
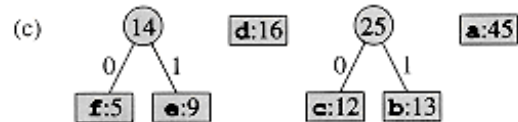
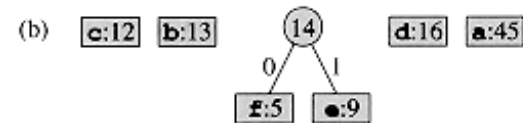


(b)

קוד הופמן

- הומצא ב-1951 ע"י הסטודנט David Huffman בתור מטלה לקורס תיאוריית המידע
- זהו אלגוריתם חמדני הבונה עץ של prefix code למעלה

(a) f:5 e:9 c:12 b:13 d:16 a:45



קוד הופמן

□ משפט 1

■ אם S מחרוזת ו- C_H קוד הופמן שלה, ו- C הוא Prefix Code
אחר. אזי $|C_H(S)| \leq |C(S)|$



Huffman הוא קידוד Prefix Code אופטימאלי

קוד הופמן ואנטרופיה

□ משפט 2

■ עבור כל מחרוזת S מתקיים

$$H_0(S) \cdot n \leq |Huff(S)| \leq (H_0(S) + 1) \cdot n$$

"גודל ממוצע של קידוד" הוא בין $H_0(S)$ ל- $H_0(S) + 1$

□ משפט 3

■ עבור כל אלגוריתם כיווץ A תהיה מחרוזת S

$$|A(S)| \geq n \cdot H_0(S) \quad \text{כך ש}$$

אין אלג' כיווץ שמסוגל לכווץ כל מחרוזת לפחות מ- $nH_0(S)$

קוד הופמן ואנטרופיה

□ הוכחה (משפט 2)

■ בהינתן S נגדיר C Prefix Code כך ש- $|C(S)| \leq (H_0(S) + 1) \cdot n$

■ נגדיר את הקוד כך:

□ את התו a_i הקוד C יקודד בעזרת מחרוזת ביטים באורך $\left\lceil \log_2 \left(\frac{n}{n_i} \right) \right\rceil$

■ נניח קיים קוד כזה, אזי:

$$|C(S)| = n_1(< \text{Coding Size} > a_1) + \dots + n_k(< \text{Coding Size} > a_k)$$

$$= \sum_{i=1}^k n_i \left\lceil \log_2 \left(\frac{n}{n_i} \right) \right\rceil \leq \sum_{i=1}^k n_i (\log_2 \left(\frac{n}{n_i} \right) + 1)$$

$$= \sum_{i=1}^k n_i (\log_2 \left(\frac{n}{n_i} \right)) + \sum_{i=1}^k n_i$$

$$= n \cdot H_0(S) + n = (H_0(S) + 1) \cdot n$$


$$|Huff(S)| \leq (H_0(S) + 1) \cdot n$$

קוד הופמן ואנטרופיה

□ טענה 1 (בדרך להוכחת משפט 3)

■ לא קיים אלגוריתם כיווץ A כך ש $\forall S, |S| = n : |A(S)| \leq n - 1$

□ הוכחה

■ אלגוריתם כיווץ הינו פ' חד-חד ערכית

■ נניח קיימת פ' כזו, אזי היא פ' חח"ע מקבוצה בגודל 2^n לקבוצה בגודל 2^{n-1}

■ אין פונקציה כזו ולכן לא קיים אלגוריתם כזה

קוד הופמן ואנטרופיה

□ הוכחה (משפט 3) - ההתחלה

■ נקבע $n_1, \dots, n_k$ ואת n לערכים מסוימים

■ כמה מחרוזות כאלה יש?

$$X := \binom{n}{n_1, n_2, \dots, n_k} = \frac{n!}{n_1! n_2! \dots n_k!}$$

□ טענה 2

■ אין אלגוריתם שמכווץ כל מחרוזת מקבוצה בת X מחרוזות לפחות מ- $\log(X)$ ביטים

□ הוכחה (טענה 2)

■ הדבר יגדיר פונקציה חח"ע מקבוצה בגודל X לקבוצה בגודל $2^{\log X} > X$

■ בסתירה לטענה 1, ולכן הטענה נכונה

קוד הופמן ואנטרופיה

□ אזי $\log(X)$ חסם תחתון על הכיווץ

□ נחשב את $\log(X)$: נשתמש בהערכה ש- $\log(m!) \approx m \log m$

$$\log X = \log \frac{n!}{n_1! \dots n_k!} = \log n! - \log n_1! - \log n_2! - \dots \log n_k!$$

$$\approx n \log n - n_1 \log n_1 - n_2 \log n_2 - \dots n_k \log n_k$$

$$= (n_1 + n_2 + \dots n_k) \log n - n_1 \log n_1 - n_2 \log n_2 - \dots n_k \log n_k$$

$$= n_1 \log n + n_2 \log n + \dots n_k \log n - n_1 \log n_1 - n_2 \log n_2 - \dots n_k \log n_k$$

$$= n_1 \log \frac{n}{n_1} + \dots n_k \log \frac{n}{n_k}$$

$$= nH_0(S)$$



הוכחנו את משפט 3. מש"ל

תם הטקס – נגמר הקורס - הסוף

בהצלחה
במבחן!!

