בס"ד

תשובות

1. שאלה 1

a. כן. לכל n תהי S_n מחרוזת המורכבת מ n תוים שונים – למשל המחרוזת שהתו במקום הi הוא i (מעל א"ב $\Sigma = \mathbb{N}$). כל סיפא במילה S_n מתחילה באות שונה, ולכן תהיה בן נפרד של השורש, וגובה העץ יהיה 1 תמיד.

קודי שגיאה:

1) ענה "לא" אבל הניח א"ב סופי וההוכחה נכונה: 5-
2) ענה "לא" והוכחה לא נכונה: 10-
3) ענה "כן" וא"ב סופי: 10-
4) ענה "כן" א"ב אינסופי והוכחה לא נכונה: 6-

b. לא. על כל קשת מהשורש לעלה יש לפחות אות אחת, והמחרוזת S_n היא באורך n. אמנם אנו מוסיפים בסוף $ ולכאורה יכול להיות מסלול באורך n+1 אבל המשמעות של זה היא שאחרי n קשתות יש פיצול, צד אחד הולך ל$ והצד השני לתו אחר. כלומר תת המחרוזת המורכבת מהתוים ב n הקשתות הראשונות מופיע פעמיים ב S_n , פעם אחת בסוף ואחריה $ ופעם אחרת באמצע עם תו אחר אחריה. זה אומר שהאורך של S_n גדול מ n וזו סתירה.

קודי שגיאה:

1) ענה "לא" כי אורך המחרוזת n ולא שם לב שיש $ בסוף: 6-
2) ענה "לא" והוכחה אחרת מהסעיף הקודם ולא נכונה: 7-
3) ענה "כן" כי שם לב שאורך המחרוזת עם $ הוא n+1: 6-
4) ענה "כן" והוכחה אחרת מהסעיף הקודם ולא נכונה: 10-

c. הגובה של עץ הסיפות של S_n מעל א"ב בגודל 3 הוא לכל הפחות $\log_4 n + 1$ זאת כיוון שלכל קודקוד בעץ יש לכל היותר 4 ילדים(לפי האותיות a,b,c והתו $), ובנוסף יש בעץ בדיוק $n + 1$ עלים. לעץ עם לכל היותר k ילדים לקודקוד, שגובהו h יש לכל היותר k^h עלים, ולכן אצלנו כאשר $k = 4$ ומספר העלים הוא $n + 1$ חייב להתקיים $4^h \geq n + 1$ כלומר $h \geq \log_4 n + 1$. זהו חסם תחתון.
דוגמה לחסם עליון שנותן עומק (*O(log n)*: ניקח א"ב {0,1,2}. המחרוזת S_n תהיה ההצגה הבינרית של כל המספרים מ 0 עד n כאשר כל שני מספרים כאלה מופרדים ע"י התו 2. אורך המחרוזת הוא N= *n log n*. כל תתי המחרוזות באורך *log n* יופיעו על קשתות בעץ הסיפות אבל אף אחת לא תופיע פעמיים, לכן עומק העץ *(O(log n)*. כמו כן מתקיים *log* N=*θ(log n)*.

קודי שגיאה:

1) הוכיח חסם תחתון ולא חסם עליון: 5-
2) הוכיח חסם עליון ולא חסם תחתון: 5-
3) נתן חסם עליון יותר גבוה מ log n ואין שגיאה: 7-
4) בחר Sn כשתי מחרוזות: 7-
5) הוכיח עבור "לכל משפחה" במקום "קיימת משפחה": 8-
6) תשובה נכונה, אין הסבר: 6-
7) התייחס לSn כמשפחה, ולא כמחרוזת: 9-
8) (סעיף ב') ענה "לא" בטענה שאורך הסיפא הארוכה ביותר היא n ולא התייחס ל $: 8-

קודי שגיאה כלליים לבעיה 1:

א. עץ לא מכווץ: 25-

ב. מצייר עץ סיפות כשהתוים בקדקדים ולא בקשתות: 10-

2. משפט הדואליות החלש: בהינתן בעית תכנות לינארי המאופיינת ע"י משתני ההחלטה x_1,x_2,…,x_n הבעיה הפרימלית מוגדרת בתור:
$\max \sum_{i=1}^{n} c_i x_i$ תוך שלכל j בין 1 לm מתקיים $\sum_{i=1}^{n} a_{j,i} x_i \leq b_j$. $x_i \geq o, i = 1, \dots, n$.
והבעיה הדואלית:
$min \sum_{j=1}^{m} y_j b_j$ תוך שלכל i בין 1 לn מתקיים $\sum_{j=1}^{m} y_j a_{j,i} \geq c_i$. $y_j \geq o, j = 1, \dots, m$.

משפט הדואליות החלש אומר ש

$$\max \sum_{i=1}^{n} c_i x_i \leq \min \sum_{j=1}^{m} b_j y_j$$

פתרון פורמלי אפשר להעתיק מהשקפים באתר. הרעיון בלי אינדקסים: נסתכל על פתרון לבעיה הפרימלית $\{x_i\}_{i=1}^{n}$, ועל פתרון לבעיה הדואלית $\{y_j\}_{j=1}^{m}$.
הפתרון לבעיה הפרימלית מקיים את כל האי-שיוויונות של האילוצים. כיוון שכל $y_j \geq 0$ ניתן לכפול את שני אגפי הא"ש של האילוץ הj בy_j.
באופן דומה ניתן לכפול את האילוץ הi של הבעיה הדואלית בx_i.
כעת אם נסכום את כל צד שמאל של האילוצים של הבעיה הפרימלית, ואת כל צד שמאל של האילוצים של הבעיה הדואלית נקבל את אותו ביטוי בדיוק – $\sum_i \sum_j x_i y_j a_{ij}$.
לפי הסכימה של הבעיה הפרימלית, ביטוי זה קטן מהערך של הפתרון לבעיה הדואלית.
לפי הסכימה של הבעיה הדואלית, ביטוי זה גדול מהערך של הפתרון לבעיה הפרימלית.
לכן, בהכרח הערך של הפתרון לבעיה הפרימלית תמיד קטן או שווה לערך של הפתרון לבעיה הדואלית.
כיוון שאי-השיוויון מתקבל לכל זוג פתרונות לבעיות, בפרט מתקיים לפתרונות האופטימליים לכל בעיה, ולכן מתקבל המשפט.
קודי שגיאה: בשאלה זו ניסוח נכון הוא 10 נקודות והוכחה נכונה היא 20 נקודות.

1) אין מימדים בניסוח המשפט: 5-
2) בהוכחה, לא מוסבר העיקרון שהיות ואי השוויון מתקבל **<u>לכל</u>** זוג פתרונות, זה חייב להיות תקף באופטימום: 10-
3) לא השתמש בעובדה שהמשתנים חיוביים: 7-

3. תשובה
א':

עבור מערך A של n ביטים, נתאר אלגוריתם עם n מעבדים, לכל מעבד אינדקס i בין 1 לn.

נגדיר משתנה "פלט".
אלגוריתם(למעבד הi)
א. פלט <- 0
ב. אם A[i]=1
a. פלט <- 1

נכונות:

אם כל A[i]=0 אז בסיבוב 1 הערך של הפלט יקבע ל-0, והוא לא ישתנה יותר, ולכן פלט האלגוריתם תהיה 0, שזהו באמת הOR של המערך.

אם קיים i כך שA[i]=1 אז בסיבוב 2 הערך של הפלט יקבע להיות 1, משום שהמעבדים היחידים שרושמים הם אלה שהערך שלהם 1, לכן בpriority CRCW נרשם 1 בכל מקרה (פשוט נלקח ה"1" של המעבד בעל מספר ה ID הגדול ביותר), ומאז לא ישתנה המשתנה "פלט". ולכן האלגוריתם אכן מחשב את תוצאת ה OR .

זמן ריצה: O(1)

עבודה O(n)

אלגוריתם במודל הסדרתי צריך לקרוא את כל הקלט, ולכן ייקח $\Theta(n)$ זמן,

לכן האלגוריתם הוא Optimal-Speedup

ב':

האלגוריתם מסעיף א' גם כאן, זאת כיוון שבסבב השעון הראשון כל המעבדים כותבים אותו דבר (False), ובסבב השני כל המעבדים שכותבים, כותבים אותו דבר (True). לכן האלגוריתם מתאים למודל הcommon CRCW שדורש שכאשר מתבצעות מספר כתיבות במקביל כולן תהיינה עם אותו ערך.

ממילא, נכונות האלגוריתם נותרת, זמן הריצה הוא קבוע והאלגוריתם הוא Optimal speedup.

קודי שגיאה:

1) לא הגדיר נכון Priority CRCW: 4-
2) לא הגדיר נכון Common CRCW: 4-
3) לא ענה נכון על optimal speedup: 4-
4) בעיה קטנה בנכונות האלגוריתם: 4- (או 2- לכל אחד אם אלגוריתמים שונים)
5) בעיה בחישוב זמן האלגוריתם: 4- (או 2- לכל אחד אם אלגוריתמים שונים)
6) לא ניצל את כוח CRCW: 4-
7) איפה כותבים בזכרון המשותף? 3-
8) ניתן אלגוריתם מבוזר, לא מקבילי: 20-
9) אין תקשורת בין מעבדי PRAM: 15-
10) נתן אלגוריתם בזמן O(n): 12-
11) אין "ערך ריק" בזכרון: 5-
12) ב CRCW אם לא כל המעבדים כותבים את אותו ערך אז התכנית עפה: 10-
13) אין הסבר למה עבודה O(n) זה optimal speedup: 5-
14) נתן אלגוריתם למעבד יחיד: 15-
15) נתן אלגוריתם עבור שני ביטים בלבד: 15-
16) הגדרה לא נכונה של הבעיה (2 מערכים באורך n ו or אינדקס-אידקס): 5-
17) סעיף ב' ללא הסבר: 7-
18) הגדרת Common CRCW כהערך שנכתב הוא הרוב: 10-
19) לא נכתב איך אותחל הזכרון המשותף: 5-
20) איך יודעים מי המעבד עם ה ID הגבוה ביותר? 10-