**ממ"ן 11 אלגוריתמיקה**

**שאלה 1**

תוכנה להשמת מתמחים

הבעיה היא חלוקת מתמחים לרופאה בבתי חולים בצורה הטובה ביותר ביחס להעדפות של בתי החולים והמתמחים.

המודל המתמטי: יש m בתי חולים שכל אחד יכול לקבל מספר מסוים של מתמחים מבין n הסטודנטים לרפואה שסיימו את לימודיהם ורוצים להצטרף לאחד מבתי החולים. לכל בית חולים יש רשימת העדפות של סטודנטים ולכל סטודנט יש רשימת העדפות של בתי חולים. המטרה היא לשבץ את הסטודנטים כמתמחים בבתי החולים בצורה יציבה כלומר שאין מצב שיש בית חולים שרוצה סטודנט מסוים יותר משאר הסטודנטים שקיבל בזמן שאותו סטודנט רוצה את אותו בית חולים יותר מזה שנבחר לו.

הבעיה המתאימה במדעי המחשב היא בעיית הזיווג היציב.

לדעתי הפתרון הטכנולוגי משתמש בצורה מסוימת של אלגוריתם **גייל-שייפלי** לפיתרון הבעיה.

**שאלה 2**

נשתמש בשתי שגרות, PowerSum שמחשבת את סכום הריבועים של הספרות של מספר מסויים ו HaapyNum שבודקת אם אם מספר שלם וחיובי נתון הוא מספר שמח.

\*\* נצא מנקודת הנחה שהחלוקה ב10 מתבצעת ללא שארית (למשל ,146/10=14 3/10=0)

האלגוריתם משתמש ברשימה prev שמטרתה לשמור את כל המספרים שנבדקו בשביל שנוכל לזהות אם הגענו למספר שכבר נבדק ובעצם מתבצעת לולאה אין סופית והמספר הראשוני הוא לא מספר שמח.

מבנה הבקרה הראשי בתוכנית הוא לולאת while שרצה עד שמגיעים למספר 1, במקרה זה המספר הוא מספר שמח.

בתחילת האלגוריתם יוצרים את prev כרשימה ריקה, לאחר מכן מתחילה לולאת הwhile שבתוכה מתבצעת בדיקה בעזרת מבנה הבקרה if-else. אם המספר קיים ברשימה prevאז מוחזר False ואם לא אז מסיפים את המספר לprev ומשנים את ערכו לסכום הריבועים של הספרות שלו בעזרת השגרה PowerSum.

בשגרה PowerSum מאתחלים משתנה בשם sum ל0 ואז עוברים ספרה ספרה ומוסיפים לsum את הערך הריבועי שלה. המעבר על הספרות מתבצע באמצעות מבנה הבקרה while, כל עוד המספר גדול מ0 מוצאים את ספרת האחדות שלו באמצעות שארית ואז "נפתרים" ממנה על ידי חילוק ב10.

**שאלה 3**

האלגוריתם שלנו מקבל עץ בינארי שמייצג ביטוי אריתמטי ומדפיס את הביטוי שלו, בשביל לעשות את זה הוא צריך להדפיס את איברי העץ באמצעות סריקה בסדר תוכי (כלומר קודם הדפסת בן שמאלי ואז צומת נוכחי ואז בן ימני) אם הצומת הוא אופרטור בינארי ואמצעות סריקה בסדר תחילי (קודם צומת ואז בן שמאלי ואז בן ימני) אם הצומת הוא אופרטור אונרי.

הצומת הוא אופרטור בינארי אם יש לו שני בנים, אופרטור אונרי אם יש לו בן אחד ומספר טבעי אם אין לו בנים (אנחנו יוצאים מנקודת הנחה שהקלט תקין). אם מדובר באופרטור בינארי צריך להוסיף סוגריים, הדרך לעשות זאת היא להוסיף סוגר שמאלי לפני ההדפסה של הבן השמאלי וסוגר ימני אחרי ההדפסה של הבן הימני.

מימשתי את השגרה בצורה רקורסיבית וחילקתי אותה לשלוש אפשרויות לפי הערך של הצומת הנוכחית (אופרטור בינארי, אופרטר אונרי או מספר) ולפי זה אפשר לדעת באיזה סדר להדפיס אום להוסיף סוגריים או לא. הקריאה הרקורסיבית מתבצעת על הילדים כי כל ילד מהווה גם הוא ביטוי אריתמטי ותנאי העצירה הוא שמגיעים לעלה כלומר שלצומת הנוכחית אין ילדים.

**שאלה 4**

א. הרעיון באלגוריתם הוא שאין סיבה להכניס לתת הקבוצה עלים כי אפשר להכניס במקומם את האבות שלהם מה שיכול לכסות יותר צמתים ומה שזה אומר שחייבים להכניס את כל האבות של עלים

נסמן בE את קבוצת הכיסוי על ידי צמתים.

**מציאת כיסוי על ידי צמתים בT:**

(1)

(2) כל עוד יש יותר מצומת אחת בעץ:

(2.1) בחר צומת T שכל ילדיה עלים

(2.2) הוסף את T לE -

(2.3) מחק את ילדיו העלים של T ואת T עצמו (ואת הקשתות המחברות ביניהם)

ב. תחילה נוכיח את טענתנו בסעיף א' שאפשר למצוא כיסוי ע"י צמתים מינמלי בלי עלים.

בהינתן כיסוי ע"י צמתים מינמלי X בהכרח אין בX גם עלה וגם את אביו (כי אחרת הוא לא מינמלי) לכן אפשר להחליף את כל העלים בהוריהם ולקבל כיסוי מינימלי C בלי עלים.

ננסה למצוא את C, בגלל שאין בו עלים חייבים להיות בו כל ההורים של העלים לכן בלולאה (2) אנחנו מכניסים הורה כזה ואז מוחקים מהעץ את כל הצמתים שכבר לא רלוונטיות כלומר צמתים שכל הקשתות היוצאות מהן כבר כוסו (בפועל מדובר בהורה וילדיו העלים).

הלולאה תסתיים רק שאין יותר צמתים בעץ או שהצומת היחיד שנשאר זה השורש המקורי (במקרה זה ילדיו כבר נכנסו לE ואין לו הורה לכן כל הקשתות היוצאות ממנו כבר מכוסות).

**שאלה 5**

א. נגדיר תת בעיה בצורה הבאה: בהינתן תוצאה סופית וכמות הדרכים השונות להגיע לכל תוצאה כך ש וגם (אבל לא וגם ), בכמה דרכים שונות ניתן להגיע לתוצאה ?

תת בעיה זו מאפשרת לנו לעבוד בצורה דומה לאינדוקציה כלומר לשים ערך לתוצאה המינימלית 0:0 ומשם להתקדם לכל תוצאה אפשרית.

מבנה הנתונים שמתאים לשמירת הפתרונות של תת בעיה זו הוא מטריצה שהאינדקסים שלה מסמנים את התוצאה והערך את מספר הדרכים להגיע אליה.

ב.

(1) ניצור מטריצה בשם Matrix בגודל M+1 על N+1 שמאותחלת באפסים

(2)

(3) לכל a מ0 עד M:

(3.1) לכל b מ0 עד N:

(3.1.1) אם הוסף ל את

(3.1.2) אם הוסף ל את

(3.1.3) אם הוסף ל את

(3.1.4) אם הוסף ל את

(3.1.5) אם הוסף ל את

(3.1.6) אם הוסף ל את

(4) החזר את

באלגוריתם זה תחילה אנחנו מייצרים מטריצה שתחזיק את כמות הדרכים השונות להגיע לכל תוצאה, את המטריצה אנחנו מאתחלים באפסים ושמים את הערך 1 באיבר שמייצג את התוצאה 0:0 (יש רק דרך אחת להגיע לתוצאה זו והיא שאין סלים בכלל).

לאחר מכן אנחנו מתחילים להכניס למטריצה את הערכים לשאר התוצאות, באמצעות שתי לולאות מקוננות שבתוכם אנו פותרים את תת הבעיה שהגדרנו בסעף א' כלומר מחשבים את כמות הדרכים האפשריות לפי המהלך האחרון (2 קבוצות 3 אפשרויות לנקודות בכל התקפה סך הכל 6) והערכים של המטריצה עבור תוצאות נמוכות יותר (יש תנאי שמוודא שאין חריגה לאינדקסים שליליים).

לדוגמא עבור התוצאה 6:5 כמות הדרכים היא סכום מספר הדרכים האפשריות עבור התוצאות 6:2, 6:3,6:4,3:5,4:5,5:5.

לאחר סיום הלולאות המטריצה "מלאה" ונשאר רק להחזיר את הערך של המטריצה עבור התוצאה הרצויה.