שאלה 1 – תכנות דינמי

3. נשים לב שבבעיה של התרמיל, אנחנו מנסים להגיע לפתרון אופטימלי- ערך מקסימלי תוך כדי התחשבות במגבלת המשקל. כלומר, ייתכן והפתרון המקסימלי יהיה בנוי מפריטים אשר לא 'ימלאו' את התיק מבחינת משקל אבל יהוו את התועלת הגבוהה ביותר. לעומת זאת, בבעיית ה subset sum אנו מחפשים דרך פתרון אשר תהווה בדיוק את הערך המבוקש. כלומר, תת סדרה שסכומה שווה לערך המבוקש (בהקבלה לתרמיל – מילוי התיק בדיוק בערך המשקל המקסימלי, כאשר התועלת שווה למשקל הפריט). בשני המקרים, דרך הפתרון מורכבת מתתי בעיות קטנות יותר, אשר בהפשטה, מסתכלות על האם "הכנסתי את הפריט לתיק" (סכמתי את המספר הנוכחי לסכום) או שלא, וחלוקה לתתי בעיות אשר מתאימות לכל אחד מהמקרים הנ"ל.
את בעיית התרמיל פתרנו בשיטת ה Bottom up שפותרת כל תת בעיה בנפרד, על מנת להבין את הפתרון האופטימלי הכללי. ב- subset sum אין חשיבות לתועלות, ולכן התועלת של כל מספר תהיה שווה לערכו. לכן, נגדיר את המשקל המקסימלי שה"תיק" יכול לסחוב בתור הערך המבוקש של ה subset sum ונדע שקיימת תת רשימה בסכום הזה, במידה והפתרון האופטימלי שווה ל"משקל המקסימלי של התיק". כלומר, אם הצלחנו למלא את התיק בדיוק במשקל המקסימלי שלו (הצלחנו לסכום מספרים עד בדיוק הערך המבוקש).

שאלה 2 – גרפים

3. האלגוריתם count islands שבנינו משתמש בכמה פונקציות בתוכו-

<u>Is Valid:</u> משתמש בפעולות בסיסיות ואין בו לולאות ולכן $O(1)$

<u>*BFS:*</u> אלגוריתם זה הוא בסיבוכיות $O(R * C)$ כאשר R מייצג את מספר השורות ו- C מייצג את מספר העמודות של המטריצה. זאת כי, בלולאת ה *while*, המקרה הכי גרוע הוא שכל התאים הם בעלי ערך 1, ובמקרה זה נעבור על כל אחד מהתאים במטריצה. לכן- $O(1) + O(R * C) = O(R * C)$

<u>*Count Islands:*</u> קורא לפונקציה *Is Valid*, עובר בלולאה כפולה על כל התאים במטריצה (על כל השורות ועל כל העמודות), ובתוך הלולאה קורא לפונקציה *BFS*.

סה"כ : $O(R * C) * O(R * C) = O([R * C]^2)$