**מבני נתונים עבודת בית 3 – מערכת לניהול חברת לוגיסטיקה**

**סעיף 1:**

**מחלקת מוצר:**

|  |
| --- |
|  |

**מחלקת מחסן:**

|  |
| --- |
|  |
|  |
|  |

**סעיף 2:**

כל מוצר מגיע עם מספר מוצר ייחודי עבורו (ברקוד), ויש אלפי הזמנות כל יום, אז בטח גם עשרות אלפי מוצרים במחסן.

להשתמש במערך לפי INDEX, מאוד לא חכם - תופס מלא מקום במיוחד שהמספרי מוצר לא עוקבים.

לשמור בעץ בינארי, ייקח מלא זמן בהגעה למוצר גם אם נשתמש בעץ AVL מינימום זמן - O(logN)

כמובן שיש עוד כמה וכמה אופציות אבל אנחנו החלטנו להשתמש ב-Hashmap שהגעה למוצר היא ב-O(1).

פונקציית hashmap מקצה מראש גודל של 16 מקומות אבל היא דינאמית, היא תגדיל את עצמה בהגעה ל75%.

כאשר המפתח הוא ה-productID והערך הוא המוצר עצמו (מצביע ל-product) – לא שמרנו את המספר מוצר בתוך product כי חבל לשמור אותו פעמיים.

|  |  |
| --- | --- |
| פונקציות על product | סיבוכיות |
| הוספת מוצר | בדיקה אם הproductID כבר קיים בפונקציית hash היא בזמן קבוע של - O(1)  הכנסה של מוצר לפונקציית Hash- O(1) |
| בדיקת מלאי | עבור מוצר בודד בדיקת מלאי היא O(1) |
| מחיקת מוצר | מחיקת מוצר בפונקציית hash היא בזמן קבוע של O(1) |
| הורדה/הוספה מכמות המלאי של המוצר | כנ"ל הורדה/הוספה מכמות המלאי, בפונקציית Hash היא בזמן קבוע של O(1) |

**סעיף 3:**

**מחלקת הזמנה:**

|  |
| --- |
|  |
|  |

**מחלקת ניהול הזמנות:**

מחלקה זו קצת מורכבת החלטנו לנהל אותה כך:

יש אלפי הזמנות ברמה יומית, ולכן יש counter, שמקצה מספר לכל הזמנה חדשה וגדל ב1 כל פעם, כדי לא לחזור על אותו orderID.

כדי להגיע להזמנה מסוימת אנו שומרים אותו בhashmap, ולא בArrayList, למרות שבשניהם היינו יכולים לשלוף הזמנה בO(1), אבל רצינו להשאיר גמישות למערכת בעתיד, שאם נרצה להסיר הזמנות, לא נצטרך לשמור על התאים ריקים כמו בarrayList, ולכן בחרנו ב Hashmap שיש לו את הגמישות הזו.

מספר ההזמנה הוא הKEY, וה-value הוא מצביע להזמנה, ההזמנה אינה מכילה את המספר הזמנה כי אין טעם לשמור אותו פעמיים.

בכל הזמנה שמרנו בhashmap את המוצרים להזמנה, הKEY הוא המספר מוצר, והVALUE הוא הכמות הנדרשת מאותו מוצר. נכון שבהקצאת map הוא מקצה 16 תאים, אבל אם הולכים על המקרה הקיצון ביותר, לכל הזמנה יכול להיות M מוצרים בתוכה ורצינו להשאיר את הגמישות בעתיד למערכת שניתן להסיר ולהוסיף מוצרים להזמנה.

**ניהול הזמנות לפי פריוריטי:** הגבלנו את הדחיפות בין 1-5

שעבור כל דחיפות יש את הLINKEDLIST שלה, הNODE בשרשרת מכיל orderID, מצביע להזמנה, והNEXT כלומר הבא בתור באותה דחיפות.

עבור כל דחיפות אנו מנהלים את זה כתור FIFO .

יש לנו שני מערכים של 5 מקומות של מצביעים, מערך אחד מצביע על הראשונים בתור והשני על האחרונים בתור.

אם הזמנה חדשה מגיעה מכניסים אותה בסוף התור ואם רוצים לטפל בהזמנה, לוקחים את ההזמנה הראשונה בתור.

אם אין הזמנות באותו פריוריטי הערך הוא NULL.

ואז ניתן לרוץ על ההLINKED LIST עבור כל פריוריטי לפי סדר החשיבות.

|  |
| --- |
|  |
|  |
|  |

|  |  |
| --- | --- |
| פונקציות על order | סיבוכיות |
| הוספת הזמנה | רצים על כל המוצרים והכמויות של ההזמנה ומכניסים לMAP- O(M),  היה אפשר ליצור את הMAP בMAIN ולשלוח להוספה כשזה מוכן. אבל לא רצינו שהMAIN יתעסק בזה.  הכנסת ההזמנה לMAP של ORDERS O(1)  יצירת אובייקט של תור והכנסה לתור הוא בO(1) |
| בדיקת מלאי עבור הזמנה | בדיקת מלאי של הזמנה זה תלוי בגודל כמות המוצרים השונים שיש בתוך ההזמנה- O(M) |
| שליחת N הזמנות | רצים על הLINKED LIST לפי התור כאשר מתחילים עם הפריוריטי הגבוה O(N)  אם מתעסקים גם בהורדת הכמויות אז עבור כל הזמנה בודקים אם  **כל**  המוצרים קיימים O(M) ואם הכל קיים, אז מורידים את כל הכמויות O(M).  לא מתחילים להוריד לפני שבודקים שהכל קיים.  בסוף מסירים מQUEUE O(1) |

**סעיף 4:**

|  |
| --- |
|  |
|  |
|  |
|  |
|  |
|  |

|  |  |
| --- | --- |
| Reports functions | סיבוכיות |
| הצגת כל ההזמנות הקיימות | רצים על הרשימות המקושרות לפי המצביעים של ראש התור שלהם לפי החשיבות 1-5 – מבחינת סיבוכיות O(N) כאשר N זה כמות ההזמנות הפתוחות  בגלל שאנחנו מציגים בדוח זה גם אם המוצרים זמינים אז רצים על כל הפריטים של כל הזמנה O(M) , כך שסך הכל O(N\*M) |
| הצגת כמות ההזמנות שלא ניתן לספק | כנ"ל כמו הדוח הראשון, חייבים לעבור על כל ההזמנות לבדוק אם הכמויות זמינות ובסוף מציגים רק את אלו שלא , אבל זה עדיין - O(N\*M). |
| כמות ההזמנות בדחיפות עליונה | רצים על הרשימות המקושרות כאשר מתחילים בדחיפות 1 אם יש שם מוצרים אז מציגים את כל הרשימה המקושרת הזו ופה מסיימים.  אם אין שם הזמנות אז עוברים לדחיפות 2 וכך הלאה. ולכן הסיבוכיות היא ככמות ההזמנות בדחיפות עליונה, O(N) |
| הצגת מלאי הנוכחי מהמוצר הנמוך לגבוה ביותר | עוברים על כל המוצרים ב - MAP O(N)  מסדרים בעץ AVL הערך בכל צומת היא כמות המוצרים, בכל צומת יש מערך ששם שומרים את הproductID של המוצרים בעלי כמות זהה. O(LOGN)  בסוף רצים inorder traversal ובכל צומת מדפיסים את כל המוצרים עם אותה כמות ששמורה שם. סה"כ סיבוכיות O(N\*LOGN) |
| הצגת K ההזמנות עם הכי הרבה מוצרים | מאוד דומה לדוח הקודם.  עוברים על כל הרשימות המקושרות (סך ההזמנות הפתוחות), O(N)  מסדרים בעץ AVL כל צומת היא כמות המוצרים באותה הזמנה, ויש מערך ששם שומרים את הorderID של ההזמנות בעלי כמות זהה. O(LOGN)  בסוף רצים inorder traversal ובכל צומת מדפיסים את כל ההזמנות עם אותה כמות. סה"כ סיבוכיות O(N\*LOGN) |

**סעיף 5:**

|  |
| --- |
|  |
|  |

**הדפסת הMain:**

\*ביצענו בדיוק את הדרישות שרשומות בסעיף 5, והוספנו כל מיני בדיקות בסוף הקוד של הMAIN

|  |
| --- |
|  |
|  |

**ההדפסה של הבדיקות שהחלטנו לצרף**

|  |
| --- |
|  |