שלו וקסמן

ת.ז. 329396956

בונוס מרצה 2: מימוש פונקציות להקצאת זכרון

דו"ח זה מכיל:

- הגדרת הערימה
- עקרונות ההקצאות בערימה
- מבנה הנתונים השומר את מצב הערימה
- תיאור של תהליך ה~~העתק-הדבק מ-ChatGPT~~- מימוש בC
- דיון על סיבוכיות ואפיקים לשיפור

**<u>הערימה</u>**

*הערה: אני רוצה להבהיר של"ערימת" הזכרון בC אין קשר למבנה הנתונים "ערימה" שבצורת עץ.*
*אתה כמובן יודע את זה, אבל לי לקח הרבה יותר מדי זמן להבין את זה...😄*

**הערימה שלנו תהיה מערך גלובלי בגודל 128 בתים.**
(בחרתי גודל שרירותי, אבל חזקה של 2 כדי לדמות זכרון אמיתי - לפי הבנתי, הסיבה לכך שגודל הזכרון תמיד מגיע בחזקות של 2 הוא המבנה של הגישה לזכרון: יש לנו אפיק נתונים שמחבר פיזית את הזכרון למעבד, ואם יש n ביטים שמוקצים לכתובת שהמעבד רוצה לקבל מהזכרון, אז אפשר לקודד מקסימום $2^n$ כתובות שונות ומן הסתם נרצה להשתמש במלוא הפוטנציאל).

## הקצאות

**בהינתן בקשת הקצאה, כיצד נבחר איפה בערימה להקצות את הזכרון?**

הגדרה: רצף פנוי בערימה, שבשני צדדיו יש זכרון תפוס או גבול הערימה, ייקרא "תת-ערימה".

ניסחתי 3 עקרונות (יש סרטוטים בהמשך שמסבירים את זה):

1. מתוך תת-הערימות הנוכחיות, בקשת הקצאה תוקצה בתת-ערימה הקטנה ביותר שתוכל לאכלס אותה.
2. כאשר מקצים בקשה בתת-ערימה, נקצה בצמוד לאחד משני הקצוות.
3. נבחר את הקצה הקטן מבין השניים, כלומר זה שמספר הבתים **התפוסים** ממנו עד לתת-הערימה הבאה או עד סוף הערימה, הוא הקטן יותר.

אני אסביר כעת את ההגיון בבחירה הזאת, אומר מראש שההגיון לא מושלם, יש כאן תחושת בטן ונפנוף ידיים והזו כנראה לא הבחירה האידיאלית (אם בכלל יש כזו).

נניח שאנחנו מקבלים בקשה להקצות 8 בתים ואז עוד בקשה להקצות 4 בתים.

נעדיף להקצות כך:

| | | |
|---|---|---|
| 8 | 116 | 4 |

מאשר כך:

| | | |
|---|---|---|
| 8 | 4 | 116 |

משום שבאפשרות השנייה, אם נקבל בקשה לשחרר את ה-8 בתים, נקבל:

| | | |
|---|---|---|
| 8 | 4 | 108 |

ו"נתקענו" עם 8 בתים שמופרדים מה108 בתים. זה רע כי למשל, אם נקבל עכשיו 7 בקשות של 16 בתים, לא נוכל להקצות את כולן.

לעומת זאת, אם מלכתחילה נקצה כך:

| | | |
|---|---|---|
| 8 | 116 | 4 |

אחרי שחרור 8 הבתים נקבל

| | |
|---|---|
| 124 | 4 |

וזה מצב יותר מועדף.

לכן, עקרון במימוש שלי הוא "להיצמד לשני הקצוות".

נחזור למצב:

| | | |
|---|---|---|
| 8 | 116 | 4 |

נניח שאנחנו מקבלים עוד בקשה להקצאה של 4 בתים. באיזה קצה נשים אותם?

אפשרות ראשונה:

| | | | |
|---|---|---|---|
| 8 | 4 | 112 | 4 |

אפשרות שניה:

| | | | |
|---|---|---|---|
| 8 | 112 | 4 | 4 |

כדי להבין מה עדיף, נסתכל על מצב בעייתי שיכול להיווצר אחרי שחרור הקצאה:

באפשרות הראשונה:

| | | | |
|---|---|---|---|
| 8 | 4 | 112 | 4 |

או ב<u>אפשרות השנייה</u>

| | | | |
|---|---|---|---|
| 8 | 112 | 4 | 4 |

ונשאל את עצמנו איזה מצב אנחנו מעדיפים. לא חשבתי על תשובה חד משמעית לשאלה. אני חושב שאם העקרון שמנחה אותנו הוא אפשור הקצאות גדולות, אז כדאי שיהיו כמה שפחות "חורים" שמופרדים מהרצף הפנוי העיקרי, ואם ישנם חורים, אז שיהיו כמה שיותר קטנים.
לכן **אני מחליט שה<u>אפשרות השנייה</u> טובה יותר**.

כאמור, זה טוב במקרה שאנחנו תמיד רוצים לוודא שנוכל לאפשר הקצאות גדולות, אבל פחות טוב בגלל ש(בדוגמה שלנו), חייבים לחכות להקצאה של 4 בתים או פחות כדי לנצל את החור, ואם היינו בוחרים באפשרות הראשונה, כל הקצאה של 8 בתים או פחות הייתה מנצלת את החור ולא נכנסת לרצף המרכזי. אז אני מניח שהבחירה תלויה בהאם אנחנו מצפים להקצאות גדולות או להרבה הקצאות קטנות.

לסיכום, עקרון במימוש שלי הוא "להיצמד לקצה היותר קטן".

לסיום, נחשוב על ה"חורים" כעל ערימות קטנות בעצמם, כך שבכל אחד מהם נרצה לקיים את שני העקרונות הראשונים.

כך שבהינתן בקשה להקצאה, נמצא את ה"חור"/ערימה הקטנה ביותר שמסוגלת לאכלס אותה, ואז נשים אותה בהתאמה לשני העקרונות הראשונים.

למשל במצב הבא:

| | | | |
|---|---|---|---|
| 8 | 4 | 112 | 4 |

אם קיבלנו בקשה להקצאת 4 בתים, נבחר להכניס אותם ל"חור" הקטן של ה-8 בתים.

מבין שני הקצוות בהם אפשר לשים את 4 הבתים, נבחר לשים אותם בקצה השמאלי שהוא הקצה הקטן יותר. (בקצה השמאלי יש 0 בתים ובקצה הימני יש 4 בתים)

| | | | | |
|---|---|---|---|---|
| 4 | 4 | 4 | 112 | 4 |

כך שעקרון במימוש שלי הוא "קודם כל לאכלס ערימות קטנות".

**<u>מעקב אחר מצב הערימה</u>**

צריך לשמור איכשהו את המצב הנוכחי של הערימה (אילו בתים פנויים, אילו בתים שייכים לאילו הקצאות...)

אם נחשוב על מקרה קצה, בו יש לנו 128 הקצאות שונות, כל אחת של בית אחד, נבין שאנחנו חייבים לשמור מידע על כל בית ובית כדי שנוכל לשחרר הקצאה יחידה.

לכן, אני אגדיר מערך נוסף בגודל 128 בתים שישמור את מצב הערימה.

עבור כל בית:

אם הבית פנוי:

ערך 0.

אם הבית אינו פנוי:

אזי הבית שייך להקצאה מסויימת.

הערך יהיה מספר הבתים מהבית הנוכחי עד סוף ההקצאה.

(זה יעזור כאשר משחררים הקצאה, כדי להבין עד איזה מיקום לשחרר)

לדוגמה, עבור ערימה עם ההקצאות הבאות:

| | | | | |
|---|---|---|---|---|
| 4 | 4 | 4 | 112 | 4 |

מערך מצב הערימה ייראה כך:

| | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 4 | 3 | 2 | 1 | 0 | 0 | 0 | 0 | 4 | 3 | 2 | 1 | 112 אפסים | 4 | 3 | 2 | 1 |

כך, עבור כל בית ש"ננחת" עליו במערך מצב הערימה, נוכל לדעת אם הבית המתאים בערימה פנוי, או, במידה והוא שייך להקצאה, מה המרחק ממנו עד לסוף ההקצאה.

בפרט:

ניתן לעבור על כל מערך מצב הערימה, לספור אפסים ולמצוא כך תתי-ערימות.

ניתן לדעת גודלה של הקצאה על פי הערך הרשום בבית הראשון במערך מצב הערימה.

ניתן לשחרר הקצאה ע"י רשימת אפסים.

**מימוש בC**

לאחר ההסבר הנ"ל, היה נראה לי שפרטי המימוש בC די מובנים, ולכן ביקשתי מChatGPT לכתוב את המימוש בC, ע"י ההנחיה הבאה:

```
write a C program that will simulate allocating memort on a heap.
Our heap will be a global array with size HEAP_SIZE bytes defined to 128.
The principles for allocating memory on the heap:
Definition: a sequence of free bytes, that has an allocated byte or the heap's end in it's ends, will be termes a "subheap".
1. An allocation request will be served on the smallest subheap that can accomodate it.
2. The memory will be allocated on one of the two ends of the subheap.
3. To choose which end, we check the distance in bytes from an end to the next subheap/end of heap. We choose the end with the smaller such distance.

The data structure to keep the status of the heap will be another array of HEAPZISE bytes.
The value of every element in this array will be: 0 if it's free else, the number of remaining bytes until the end of it's memory block.

Hence, Allocating K bytes will include:
1. finding the index of the starting point of the minimum subheap that can accomodate K bytes, by going over the heap_status, counting zeros, and keeping track of the best one until now
2. finding the distances belonging to the two edges of the subheap
3. choosing the smaller edge and marking K bytes as allocated.

implement these functions: my_malloc, my_calloc, my_realloc, my_free
That have the same interface as the regular C functions.
```

אני לא בטוח אם זה חסך לי זמן, כי בקוד שקיבלתי היו טעויות קטנות וגם סתם דרכים עקומות לעשות דברים פשוטים. כנראה שעדיף להשתמש בבינה המלאכותית בצורה שיטתית לכתיבת פונקציות קצרות מאשר למימוש רעיון שלם.

עברתי על כל הקוד שלו, פעמים תיקנתי טעויות קטנות ופעמים כתבתי הכל מחדש, והוספתי מעט תיעוד.

בדקתי את הקוד ע"י הרצה ידנית של מקרים שחשבתי עליהם והדפסת מערך מצב הערימה בכל שלב, ולבסוף הייתי מרוצה מהפעולה שלו.

לאחר מכן ביקשתי ממנו להכין קובץ h. מתאים ובנוסף קבצים של בדיקת התכנית.
בשלב הזה כבר לא היה לי הרבה זמן, היו מעט בעיות בתוצאה המצופה בבדיקות שהוא כתב.
תיקנתי את הבדיקות אבל לא התעמקתי ובדקתי האם הוא בודק את כל מקרי הקצה. כאמור, בדקתי ידנית מקרים שחשבתי עליהם לפני כן כך שבשלב הזה כבר הכל התנהג כצפוי.

טעות שלו בכתיבת הבדיקות היא שאם בדיקה נכשלת, היא לא משחררת את הזכרון, מה שפוגע בבדיקות שלאחריה. לא תיקנתי את זה כי כבר לא היה לי זמן, כרגע כל הבדיקות מצליחות אז כל הזכרון משוחרר.

*הערה: שאלתי את ChatGPT מדוע הוא חשב שבחרתי בעקרונות הללו להקצאות בערימה, והוא ענה שמדובר בעיקר על Reducing fragmentation שזה אכן היה ההגיון שלי. אז לפחות לא דיברתי שטויות גמורות (עד כמה שזה מנחם שChatGPT מסכים איתי...)*

**<u>ניתוח סיבוכיות</u>**

מקום:
המימוש מחייב מערך נוסף הזהה בגודלו לערימה עצמה, על מנת לשמור את מצבה.
(למען האמת, צריך מערך מצב עם **כמות איברים** זהה לכמות הבתים בערימה, אך ככל שגודל הערימה גדל, צריך יותר סיביות כדי לייצג את המספר המקסימלי של גודל הקצאה, כך שמערך המצב יהיה **גדול יותר** מהערימה עצמה... במקרה בגודל 128 התאים ששניהם יהיו unsigned char אבל בכללי זה לאו דווקא כך. אני לא מתקן את זה כרגע אבל זו טעות שלי (וכמובן של ChatGPT)).

זמן:
הקצאה דורשת מעבר על כל הערימה עד למציאת תת-ערימה מתאימה (ואז עוד פעולות לינאריות קטנות יותר). כך שמדובר ב$O(n)$ כאשר n גודל הערימה.

פעולת שחרור רצה על כל ההקצאה ומאפסת אותה. כך שמדובר ב$O(k)$ כאשר k גודל ההקצאה.

כאמור, אני חושב שהמימוש הנ"ל הכרחי כי למשל במקרה של מלא הקצאות בנות בית אחד, אין מנוס מלשמור מידע על כל בית ובית.

במקרה שאנחנו מצפים לעבודה עם הקצאות יותר גדולות, אפשר לנסות לשמור מידע רק על תת-הערימות (רצפים ריקים) ועל גודל הגבולות שלהן. זה יכול להיעשות ברשימה מקושרת בה כל איבר ייצג תת-ערימה קיימת. כל איבר יכיל מצביע לאיבר שמייצג תת-ערימה צמודה, מידע על כתובת התת-ערימה, גודלה, גודל גבולה השמאלי וגודל גבולה הימני.
הקצאה בתוך תת-ערימה היא בצמוד לקצה ולכן פשוט תגדיל את הגבול שלה ושל הערימה הצמודה.
**כעת קל יותר לעשות הקצאה: במקום לעבור על כל גודל הערימה צריך לעבור רק על מידע מרוכז של תת-הערימות.**
שחרור עשוי להקטין גבולות של ערימות קיימות, או דווקא לפתוח תת-ערימה נוספת.
זה מסובך; במקרה כזה צריך לשנות את המצביעים של תתי הערימות בהתאם ו"לדחוף" איבר חדש ביניהן. לכן אנחנו משתמשים ברשימה מקושרת, אבל צריך לנהל אותה בתוך מערך שגודלו ידוע מראש. זה אפשרי (אני חושב) אבל מאוד מסובך ובמקרה הגרוע זה פחות יעיל ממערך מצב ולכן לא השתמשתי בזה.

לסיכום, ברור שיש הרבה שיקולים ואני רק גירדתי את קצה הקרחון. היה מאוד מעניין לעשות את המשימה, תודה רבה!