הטכניון - מכון טכנולוגי לישראל **הפקולטה למדעי המחשב**

1 ליולי 2015

פרופ"ח ארז פטרנק
נחשון כהן

אלגוריתמים לניהול זיכרון - 236780
אביב תשע"ה – מועד א'

הנחיות:

- הבחינה עם חומר סגור.
- אסור להחזיק פלאפונים ברשות הנבחנים בזמן הבחינה.
- כתבו בצורה מסודרת ונקייה ובכתב ברור.
- בבחינה 4 שאלות, משך הבחינה שלוש שעות.
- נמקו את כל תשובותיכם.
- כאשר מבקשים להציע פיתרון אלגוריתמי לאיזושהי בעיה, יש לנסות לתת פיתרון יעיל ככל היותר במקום ובזמן.

בהצלחה !

שאלה 1: (21 נקודות) שאלה זו מתייחסת לתזמון של אלגוריתם mark-and-sweep.

א. (2 נק) מה גורם להתחלת איסוף באלגוריתם mark-and-sweep stop the world? הסבירו מדוע זו בחירה טובה (של זמן לתחילת איסוף).

ב. (8 נק) בדרך כלל רוב זמן הריצה של איסוף מושקע בסריקה (mark) וכמעט כל השאר ב-sweep. אילו מהפרמטרים הבאים משפיעים על זמן הסריקה? עבור כל פרמטר של ה-heap כתבו האם יש לו השפעה רבה על זמן הריצה הכולל של הסריקה. נמקו.
   a. מספר האובייקטים החיים
   b. מספר האובייקטים המתים
   c. מספר הבנים (שאינם NULL) הממוצע של אובייקטים חיים
   d. גודל ה heap

ג. (6 נק) חוקרים טענו שתיתכננה תוכניות מחשב שעבורן אפשר להרויח אם מתחילים את האיסוף בזמנים התלויים בהתנהגות התוכנית ולאו דווקא לפי הגורם שתיארתם בסעיף הראשון. תארו תוכנית שעבורה עדיף להתחיל את האיסוף לפני הזמן שנקבע בסעיף א. יש לתאר מצב שבו אם נתחיל את האיסוף מוקדם אז האיסוף יהיה מהיר, ואם נתחיל את האיסוף מאוחר (כמו שתואר בסעיף א) אז האיסוף יהיה איטי.
יש להסביר מה מצב ה heap בזמן ה"מוקדם", מה מצב ה heap בזמן ה"מאוחר", ומה התוכנית עושה שיוצר את המצבים האלה של ה-heap.

ד. (5 נק) נתבונן בריצה מסויימת של תוכנית ונסמן ב t_{start} זמן מסויים שבו ה heap התמלא. נסמן גם ב- t_{end} את הזמן שבו הוקצה האובייקט הראשון אחרי t_{start}. ברור שחייב להתבצע איסוף לפני t_{end} (אחרת לא יהיה אפשרי להקצות זכרון, כי הנחנו שה heap מלא). שימו לב שבזמן שבין t_{start} ל t_{end} אין אף הקצאת זכרון, אבל חוטים יכולים עדיין לרוץ ולבצע פעולות אחרות. האם יש פעולות שהתוכנית עושה שיכולות להשפיע על יעילות הסריקה? האם הפעולות האלו יגרמו לסריקה מהירה או איטית יותר? במילים אחרות, האם במצב כזה עדיף להתחיל את האיסוף בסוף האינטרוואל שבין t_{start} ל-t_{end}? או בתחילתו? או שלא משנה? הסבירו את תשובתכם.

שאלה 2 (25 נקודות) שאלה זו מתייחסת לאלגוריתם סריקה מקבילי

באלגוריתם סריקה mark&sweep ישנן 2 שיטות מרכזיות: לסמן שהאובייקט חי בתוך ה header של האובייקט (נסמן ב header mark-bit), או להשתמש בטבלה מיוחדת שבה לכל אובייקט מתאים ביט אחד (נסמן שיטה זו ב side mark-bit).

א) (3 נק) לכל אחת מהשיטות תארו יתרון שלה על השיטה האחרת.

ב) (2 נק) שימו לב שבסריקה מקבילית ייתכן ששני חוטים מגלים את אותו אובייקט ומסמנים אותו במקביל. אבל כאשר משתמשים ב header mark-bit אין צורך לסנכרן את פעולת הסימון הסבירו מדוע. לצורך השאלה הניחו שפרט ל mark-bit ה header נשאר קבוע לאורך זמן הסריקה.

ג) (4 נק) הסבירו למה חוסר סינכרון בסימון (שמוצג בסעיף הקודם) יכול לגרום לעיתים ל"בזבוז" ולסריקה יותר איטית של ה-heap. הדגימו מקרה (כלומר צורת heap ומהלך סריקה) נדיר שבו החוסר בסינכרון יגרום לאיטיות ממשית בסריקה ואילו סינכרון היה מונע האטה זאת.

ד) (2 נק) הסבירו מדוע חובה להשתמש בפעולה מסונכרנת כאשר משתמשים ב mark-bit table.

ה) (8 נק) נניח שמשתמשים בטבלת סימון נפרדת mark-bit table. שימו לב שכאשר לאובייקט אין בנים כלל (למשל, אובייקט מסוג Integer), אין צורך לקרוא את תוכן האובייקט מה heap (כי אין צורך לגלות מי בניו) אלא מספיק לסמן אותו. הקריאה הראשונה מהאובייקט (למשל קריאת ה header) היא פעולה שגורמת פעמים רבות ל- cache miss ולכן היא מאד איטית. היה טוב אם היה ניתן לחסוך קריאה זו. עם זאת, אלגוריתם סריקה רגיל מגלה את סוג האובייקט רק על-ידי קריאת ה header שלו ורק אז יודע שאין לו בנים.

תכננו אלגוריתם ניהול זכרון שיהיה יותר מהיר מבחינת זמן הסריקה משום שיוכל לגלות שלאובייקט אין בנים מבלי לקרוא אותו (ולטעון אותו לזיכרון). ניתן להניח שרוב האובייקטים אינם מכילים בנים כלל כך שהאופטימיזציה שתציעו מייעלת באופן ברור את הריצה.

ו) (6 נק) נניח שאלגוריתם האיסוף מחזיר דפים ריקים לחלוטין למערכת ההפעלה.

a. (2 נק) כיצד נבדוק האם דף הוא ריק לחלוטין כאשר משתמשים ב side mark-bit table? כיצד נבדוק האם דף הוא ריק לחלוטין כאשר משתמשים ב header mark-bit? מה יותר יעיל?

b. (4 נק) כאשר משתמשים ב header mark-bit, הציעו שינוי לאלגוריתם הסריקה שיאפשר לבדוק האם דף ריק לחלוטין בצורה מהירה. השתדלו להציע שיטה שעלות הזיכרון שלה אינה גבוהה מדי.

שאלה 3 (26 נקודות) שאלה זו מתייחסת לאלגוריתם ה compressor

א) (4 נק) ה compressor משתמש ב bit-vector (הפלט מאלגוריתם סריקה) וב- offset vector, כדי לעדכן מצביעים ל to-space. הסבירו כיצד מחשבים כתובת חדשה של אובייקט בהינתן כתובת ישנה שלו.

ב) (4 נק) ה parallel compressor מעביר את כל האובייקטים ב from-space ל to-space. אבל, לא כמו semi-space copying, לא שומרים חצי מהזכרון פנוי בשביל to-space. הסבירו כיצד משתמשים בשחרור והקצאה של זיכרון וירטואלי כדי לחסוך במשאבים של מקום בזמן הריצה.

ג) (4 נק) הסבירו מה לא יעבוד אם ננסה להשתמש באותו רעיון של סעיף ב כדי ליעל את semi-space copying.

ד) (2 נק) באלגוריתם ה compressor, מדוע לא ניתן (לפחות באופן נאיבי) לאחד את שלב הסריקה (tracing) עם שלב הדחיסה (compact)?

ה) (8 נק) רוצים לשפר את המקביליות של אלגוריתם דחיסה כלשהו ע"י הרצה בו זמנית של הסריקה והדחיסה. לצורך כך נניח שה-heap מנוהל בשני חלקים: A, ו-B, וידוע שאין אף פויינטר מחלק A אל אובייקט בחלק B.

אנו רוצים לתכנן עבור מקרה כזה אלגוריתם שידחוס את חלק B (לאחר שכבר ידוע מי חי בו) בזמן שחוטים אחרים סורקים את חלק A (כדי לדעת מי חי בו). תארו אלגוריתם איסוף עבור שני החלקים של ה-heap שמאפשר מקביליות חלקית של סריקה ודחיסה. הסבירו את שלבי האלגוריתם ותארו כיצד הוא יעבוד.

ו) (4 נק) מה משתנה אם קיימים מחוונים מאובייקטים בחלק A אל אובייקטים בחלק B? האם האלגוריתם שהצעתם עדיין עובד? אם כן – נמקו, אם לא תנו דוגמא לבעיה.

שאלה 4: (28 נקודות) שאלה זו מתייחסת לאלגוריתם reference counting ולשיטת ה-update coalescing.

א. (4 נק) באלגוריתם reference counting הבסיסי מעדכנים את שדה ספירת המחוונים בכל כתיבה. הסבירו מה העידכון ומתי אוספים אובייקט.

ב. (4 נק) באלגוריתם reference count עם update coalescing לא מעדכנים את שדה ספירת המחוונים כל כתיבה, אלא רק בזמן איסוף. לצורך כך בזמן כתיבה שומרים עותק של האובייקט כפי שהיה לפני הכתיבה (אם לא קיים כבר עותק).
הסבירו כיצד מעדכנים את שדה ספירת המחוונים לאחר מכן בזמן האיסוף.

ג. (6 נק) עבור אלגוריתם reference count עם update coalescing שדה ספירת המחוונים של אובייקט לא מכיל תמיד את כמות האובייקטים שמצביעים עליו.

a. תארו תנאי (מספיק, לאו דווקא הכרחי) עבור אובייקט A כך שאם התנאי מתקיים אז מונה המחוונים של אובייקט A מדוייק (כלומר אומר כמה מחוונים מצביעים על A מה-heap) למרות שהתוכנית שינתה את ה-heap מאז העידכון האחרון של מוני המחוונים. התנאי צריך להיות מתואר באמצעות המחוונים שרשומים כרגע ברשימת (הכתובות של) המחוונים שהשתנו מאז המחזור הקודם, הערך שזכור עבורם ברשומה שלהם והערך העכשווי שלהם. נמקו את תשובתכם.

b. תארו תנאי (הכרחי ומספיק) שאם הוא מתקיים אז כל האובייקטים ב-heap מכילים ספירת מחוונים נכונה אפילו אם התוכנית שינתה את ה-heap מאז העידכון האחרון של מוני המחוונים. התנאי צריך להיות מתואר באמצעות אותם משתנים וערכים שהוזכרו בסעיף הקודם. נמקו את תשובתכם.

ד. (6 נק) נניח שממבצעים איסוף לפי דורות. תארו שני יתרונות של reference counting יחסית ל-copying לאיסוף הדור המבוגר. הסבירו את היתרון של copying על reference counting לאיסוף הדור הצעיר מנקודת מבט של עלות האיסוף.

ה. (8 נק) נתבונן במערכת שבה משתמשים ב- reference counting עם update coalescing לאיסוף הדור המבוגר.

a. (3 נק) הסבירו מהם inter-generational pointers ולמה הם חשובים לאיסוף הדור הצעיר (באופן כללי עבור איסוף בשיטת דורות).

b. (5 נק) נניח שבכל איסוף של הדור הצעיר מעבירים את כל האובייקטים ששורדים אל הדור המבוגר. הסבירו כיצד מוצאים את המחוונים הבין-דוריים מהמנגנון שממילא קיים באיסוף של הדור המבוגר בלי להוסיף שום מבנה נוסף. נמקו!