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

23 למרץ 2012

פרופ"ח ארז פטרנק
שחר תמנת

אלגוריתמים לניהול זיכרון - 236780
חורף תשע"ב – מועד ב'

הנחיות:

- הבחינה עם חומר סגור.
- אסור להחזיק פלאפונים ברשות הנבחנים בזמן הבחינה.
- כתבו בצורה מסודרת ונקייה ובכתב ברור.
- בבחינה 3 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: (27 נקודות) שאלה זו מתייחסת לאלגוריתם ה- Mostly Concurrent.

א. (4 נק) תארו כל פאזה באלגוריתם: scanning, card cleaning, ו- stop-the-world.
ב. (2 נק) מתי ואיך card (כרטיס) הופך ל-dirty?
ג. (2 נק) מתי ואיך כרטיס מתנקה? (כלומר סימן ה-dirty שלו נכבה?) מה עושים כשמנקים כרטיס?
ד. (4 נק) לפעמים כאשר מנקים כרטיס אין צורך בשום סריקה של אובייקטים עבור הניקוי. נסחו את התנאי המדוייק המתאר מתי זה קורה.
ה. (4 נק) נניח שהתנאי מהסעיף הקודם מתקיים באמצע הביצוע, בדיוק בנקודה בה הולכים לסמן כרטיס כמלוכלך. האם חייבים לסמן את הכרטיס כמלוכלך? או שניתן לוותר על הסימון ולהשאיר את הכרטיס לא מסומן? אם ניתן לוותר, הצדיקו את הוויתור. אם לא, תנו דוגמה שבה נוצרת בעיית נכונות.
ו. (8 נק) תכננו שיפור אלגוריתמי, אשר יקטין את מספר הכרטיסים שהאלגוריתם סורק במהלך פעולתו. הסבירו תחת אילו נסיבות של הריצה, ייחסך ניקוי של כרטיס.
ז. (3 נק) האם באלגוריתם שתיארתם ייתכן כרטיס שלא נסרוק ועליו יהיו אובייקטים מסומנים?

שאלה 2: (16 נקודות) שאלה זו מתייחסת לאלגוריתם Copying Garbage Collection.

א. (3 נק') אם נצליח לשנות את האלגוריתם כך ש- to-space יתפוס פחות מחצי heap. כיצד יועיל השינוי?
ב. (3 נק') מדוע באלגוריתם המקורי גודל ה- to-space חייב להיות לפחות חצי מגודל ה-heap?
ג. (10 נק') נרצה לחסוך מקום ע"י חלוקת ה- heap לשלושה חלקים שווים. חלק אחד יישמר ריק (to-space). בעת האיסוף, נרצה להשתמש באלגוריתם ההעתקה הרגיל לעבור על שליש אקטיבי אחד של ה-heap (שייקרא from-space-1) ולהעביר את האובייקטים החיים ממנו אל השליש הפנוי (אל to-space). הבעיה היא שלא ניתן לדעת מי חי ב- from-space-1 כי חלק מהמצביעים אל אובייקטים אליו נמצאים באזור האקטיבי השני: from-space-2.
אלגוריתם הרכבת מצליח להתמודד עם הזזת אובייקטים משטח חלק של ה-heap.
   a. איך הוא פותר את הבעיה/ות שנוצרת?
   b. אם ננסה לממש את אלגוריתם הרכבת עם 2 קרונות המייצגים את שני חלקי ה-heap האקטיבי כפי שתואר בסעיף הקודם, מה תהיה העלות הנוספת של פתרון זה (בזמן ומקום) מעבר לאלגוריתם ה-copying הרגיל?
   c. האם נוכל לאסוף כל אובייקט לא נגיש? נמקו תשובותיכם!

שאלה 3: (33 נקודות) גם שאלה זו מתייחסת ל-Copying Garbage Collection .

א. (4 נק') להזכירכם, האלגוריתם של Cheney הוא Copying Algorithm שעובד ללא mark stack . במקום, הוא משתמש בשני מחוונים שמצביעים אל ה-To Space: Scan ו-Free. הסבירו בקצרה את האלגוריתם של Cheney תוך התייחסות לתפקידו של כל אחד משני ה-מחוונים האלו.
ב. (2 נק') מהו תנאי הסיום של האלגוריתם של Cheney ? כלומר, מתי יודעים שכל האובייקטים הועתקו ל: to-space?
ג. (3 נק') Baker's incremental copying הוא שדרוג לאלגוריתם של Cheney. הסבירו איזו בעיה באלגוריתם של Cheney האלגוריתם של Baker מנסה לפתור.
ד. (3 נק') באלגוריתם של Baker נעשה שימוש בשלושה pointerים: Scan, Free, Top. הסבירו מה תפקידו של ה-pointer הנוסף (Top) ומדוע הוא לא נחוץ באלגוריתם של Cheney.
ה. (3 נק') שאלה זו מתייחסת לאיכות הדחיסה של האלגוריתמים של Cheney ושל Baker. להזכירכם, איכות הדחיסה נמדדת בשני פרמטרים: מספר האזורים בהם יש אובייקטים לאחר הדחיסה, וסדר האובייקטים לאחר הדחיסה (Arbitrary, Sliding, Linearizing).

קבעו עבור כל אחד מן האלגוריתמים של Cheney ושל Baker מהו מספר האזורים שהוא יוצר, ומהו סדר האובייקטים לאחר הדחיסה. נמקו את תשובותיכם בקצרה.

ו. (8 נק') עליכם להציע שינוי לאלגוריתם של Cheney שישפר את איכות הדחיסה בכיוון של linearizing. המטרה היא שליד כל אובייקט A בסידור החדש יהיה אובייקט B אשר A מצביע עליו, אלא אם כן ל-A אין בנים או שכל בניו הועתקו כבר ל-to space בשלב בהגענו לטפל בו. על האלגוריתם החדש להמשיך לעבוד ללא mark stack, אולם הוא עלול לסרוק את האובייקטים החיים יותר מפעם אחת. אין להוסיף זיכרון לשם המימוש, פרט למספר קבוע של תאים. הסבירו את האלגוריתם החדש, ואת התאים הנוספים שעשיתם בהם שימוש.

ז. (3 נק) הסבירו מדוע האלגוריתם שהצעתם בסעיף הקודם נכון, הסבירו מדוע הוא אכן מעתיק בן ליד כל אובייקט, והסבירו מה העלות הנוספת שלו מעבר לאלגוריתם של צ'ייני (אם יש כזו).

ח. (3 נק') מהו תנאי הסיום (termination) עבור האלגוריתם החדש ?

ט. (4 נק') האם ניתן לבצע שינוי דומה על-מנת לקבל את אותה תכונה ב-to space המתקבל מהאלגוריתם של Baker? כלומר, שליד כל אובייקט יהיה אובייקט ילד שלו (אלא אם כן אין לו ילד ב-from space בזמן שמעתיקים אותו ל-space-to)? אם כן, הסבירו כיצד והסבירו מה העלות של השיטה והאם עלות זאת פוגעת בתכונת ה-incremental-יות של האלגוריתם. אם לא נתין להשיג זאת, הסבירו מדוע.

שאלה 4: (24 נקודות) מתייחסת ל- Parallel Garbage Collection

א. (3 נק) האם מבנה גרף האובייקטים ב-heap יכול להשפיע על האפשרות לבצע trace של ה-heap במקביל? רמז: חישבו על סריקה של heap המכיל רק linked-list אחד ארוך.

ב. (3 נק) יהי n מספר טבעי כלשהו. תנו דוגמא למבנה heap שעבורו יהיה קל למקבל סריקה עבור n חוטים (כלומר יהיה ניתן להפעיל באופן יעיל n חוטים מקבילים שיסרקו את ה-heap מהצורה שהצעתם) אבל מנגנון סריקה סטנדרטי לא יוכל לנצל יותר מ-n חוטים מקבילים לסרוק את אותו heap.

ג. (18 נק) נניח שיש מחשב מקבילי עם המון מעבדים, ורוצים לנצל אותו לסריקה מקבילית של ה-heap. אבל – מבנה ה-heap לא מאפשר זאת, כפי שהדגמתם בסעיפים הקודמים.

נתבונן בשיטה המנסה לנצל את משאבי החישוב הפנויים כך: כל מעבד שאינו מוצא עבודת סריקה פנויה, מתחיל "סריקה אופטימיסטית" שבה הוא דוגם אובייקט אקראי ב-heap שטרם נסרק, קובע איזשהו צבע שעדיין לא השתמשו בו (נניח כתום), מסמן את האובייקט הנבחר בכתום וסורק וצובע גם את כל בניו (שלא נסרקו עדיין) בכתום. כאשר הסריקה העיקרית (שצובעת אובייקטים נגישים בשחור) נתקלת באובייקט מצבע מסוים, הסריקה מסמנת לעצמה שכל האובייקטים המסומנים בצבע הזה חיים. אובייקטים הצבועים בצבעים שלא התגלו ע"י הסריקה המרכזית ייחשבו כמתים.

שימו לב שייתכן גם מצב בו סריקה של צבע כתום נתקלת באובייקט הצבוע כבר בירוק (כלומר, לא בשחור). עבור מקרים כאלו נשמור טבלה שתאמר איזה צבע נגיש מאיזה צבע.

a. (3 נק') כיצד יוחלט לאילו אובייקטים לעשות reclaim בשלב ה-sweep ?

b. (3 נק) בסריקה המתוארת, ייתכן שאובייקטים לא נגישים מהשורשים יקוטלגו כחיים. תארו מצב שבו אובייקט שאינו נגיש יסומן כנגיש.

c. (12 נק') בהערכות הביצועים שנעשו ל-color marking, הסתבר שעבור רוב התוכניות תועלתו קטנה. אבל הועלתה השערה שעבור תוכניות מסויימות הוא יהיה יעיל יותר.

1. תכניות בהן מתוך כל אובייקט יוצא רק pointer אחד.
2. תכניות המקיימות את התכונה הבאה: אובייקט A נגיש מאובייקט B אם ורק אם אובייקט B נגיש מאובייקט A, כלומר האובייקטים הנגישים יוצרים גרף קשיר הייטב.
3. תכניות העובדות תחת משאבי זיכרון מוגבלים. כלומר, השטח של האובייקטים החיים תופס כמעט את כל ה-heap והשטח של האובייקטים המתים הוא קטן יחסית.

כל תכונה מהשלוש הנ"ל חושפת חולשה של השיטה הנ"ל ומנסה למנוע אותה. תארו לכל תכונה כזו מהי החולשה באלגוריתם המקורי שהיא מנסה למנוע. (או פשוט הסבירו מדוע השיטה אמורה לפעול טוב לתוכניות כאלו.)