קובץ PDF עם כמה מילים על הגרפים השונים (יגור ויצחק):

בעמוד הזה מצורפים כמה מילים על הגרפים, בעמוד הבא מצורפים כל הגרפים ממוספרים.

גרף מספר 1:

כפי שניתן לראות, הגרף הזה מציג טסט שהאלגוריתם שלנו עשה על מספרים קטנים בטווח קטן. למעשה, קיבלנו מערך רנדומלי/ אקראי בטווח של [200,300] וכפי שניתן לראות בגרף המיון המהיר ביותר הוא המיון שעובד עם אלגוריתם המיון של radix-sort עם 16 ביטים. ולמעשה, האלגוריתם מיון האיטי ביותר הוא quick-sort . הסיבה לכך להערכתנו היא כיוון שמורכבות הזמן במקרה הגרוע ביותר הוא $O(n^2)$. כי מיון מהיר משתמש באלגוריתמי מיון פשוטים כגון insertion-sort שהמורכבות זמן שלו הממוצעת היא $T_w(n^2)$ למערכים עם מספרים בטווח נמוך, וגם radix-sort שהמורכבות זמן שלו הממוצעת הוא $O(n * k)$. אפשר גם לראות ש- merge-Sort הוא $O(n \log n)$ כאשר המורכבות היא $T_{w_1} T_{A_1} T_B(n \log n)$.

גרף מספר 2:

בגרף הזה נוכל לראות שהאלגוריתם שלנו בקוד עובד על טווח צר יותר של מספרים גדולים. המיון המהיר ביותר כמו שאפשר לראות מהגרפים הוא radix-sort . והכי איטי הוא quick-sort כמו בגרף הראשון. שמציג זמן גרוע ביותר של מורכבות זמן של אלגוריתם $O(n^2)$. אבל הפעם זה היה יותר איטי מאשר בגרף הראשון (210 אלפיות השנייה לעומת 78 אלפיות השנייה) לביצוע האלגוריתם. כל מיון אחר ביצע מיון כצפוי. merge-Sort עם מורכבות זמן $O(n \log n)$. שסיים את המיון פחות או יותר בדומה לזמן כמו בגרף 1 עם 22 אלפיות השנייה.

גרף מספר 3:

בגרף הזה האלגוריתם עבד על טווח צר יותר של מספרים קטנים. בגרף רואים שיש לנו כמעט אותם תוצאות כמו ב-2 הגרפים הקודמים. האלגוריתם-מיון של radix-sort שוב מסתמן כאלגוריתם המהיר מכולם. והאלגוריתם מיון של quick-sort שוב מסתמן כאלגוריתם האיטי ביותר כמו שכבר שמנו לב ב-2 הטסטרים הקודמים. השוני הוא שבגרף הזה להבדיל מהקודמים הוא היה טיפה מהיר יותר בזמן של 64 אלפיות השנייה כדי למיין מערך. merge-Sort ביצע מיון באותו זמן כמו בטסטים הקודמים.

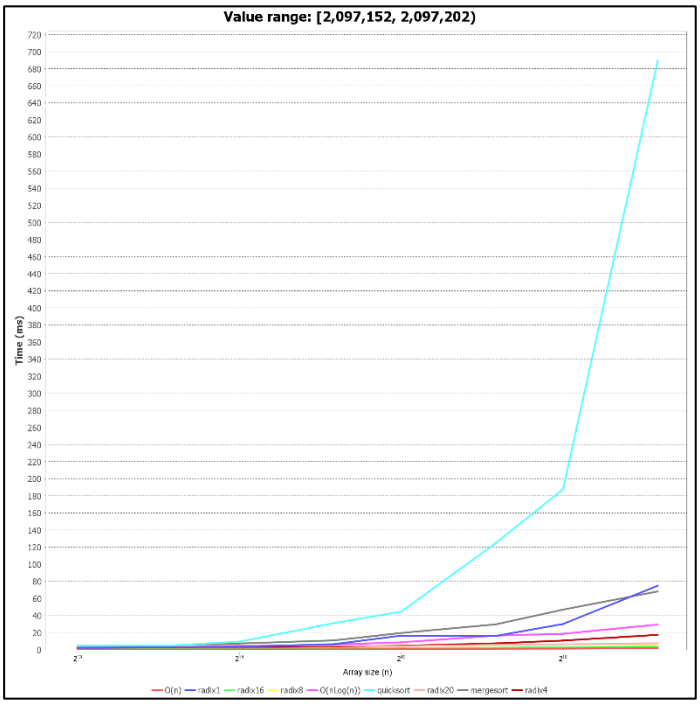
גרף מספר 4:

כאן הגרף מציג את התוצאות של האלגוריתם שלנו כאשר הוא עבד על מערך אקראי של מספרים גדולים. הפעם קיבלנו תוצאות שונות בקשר לאלגוריתם מיון radix-sort , בניסויים הקודמים מיינו מערך מהר יותר מכל סוג אחר אבל הפעם זה לא קרה. אפשר לראות שהאלגוריתמים quick-sort ו merge-Sort מיינו את המערכים הללו בזמן ממוצע של $O(n \log n)$. עוד משהו מעניין שאפשר לראות כאן זה ש quick-sort עשה את זה למעשה מהר יותר מאשר merge-Sort . (ב-4 אלפיות השנייה). הבדל נוסף הוא שמורכבות הזמן של radix-sort היה $O(n \log n)$ אז אפשר להסיק מזה בעצם ש $k = O(n \log n)$. דבר אחד חריג שכדאי לשים לב אליו הוא radix-sort(1) נכנס למורכבות הזמן של $O(n^2)$, אז חישבנו רק ספרה 1 בכל פעם בתהליך המיון. במקרה הזה זה הפך להיות דומה ל-counting-sout עם מורכבות זמן של $O(n^2)$.

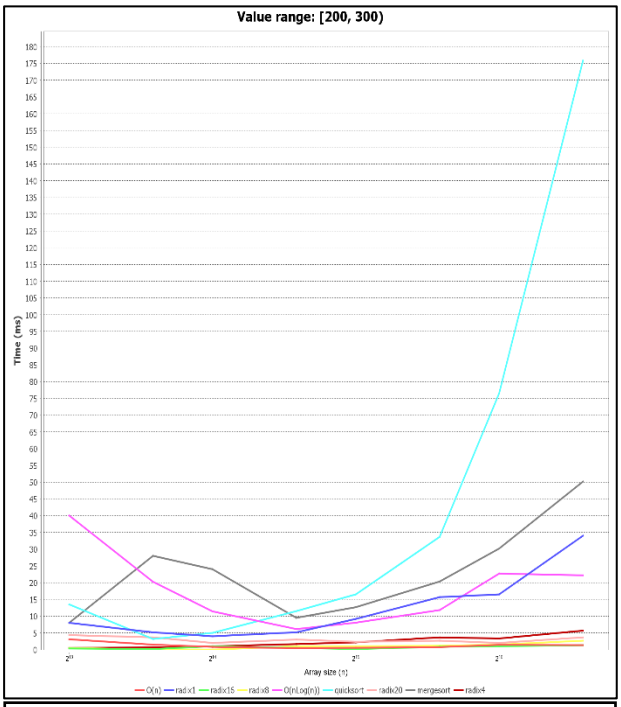
לסיכום:

הטסטים האלה הראו לנו שאלגוריתם המיון הטוב ביותר למיון מערך המספרים יהיה radix-sort למספרים עם 16 ספרות. ראינו ש-quick-sort הוא לא ממש שימושי במערכים קטנים, הוא כן יכול להיות מועיל עם מערכים גדולים. ראינו ש- merge-Sort יעיל מאוד בכל מיני סוגים של מערכים בטווחים שונים של מספרים.

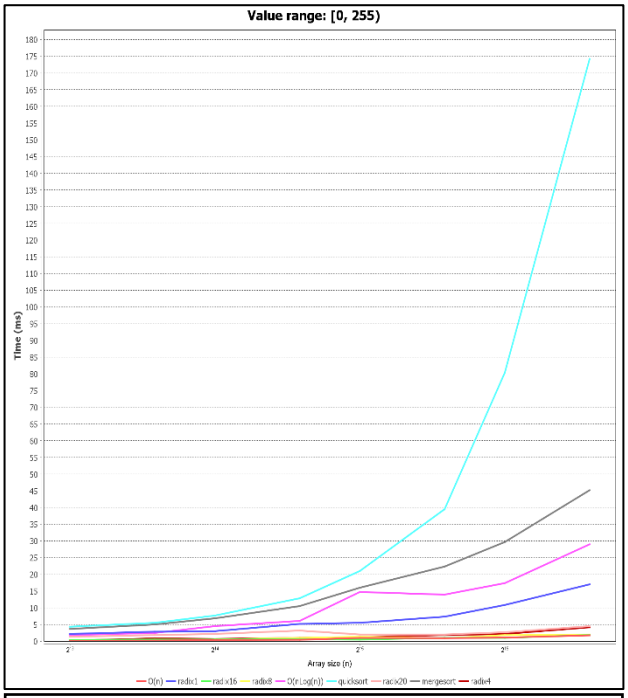
אפשר לראות שבשלושת הגרפים הראשונים quick-sort הוא $O(n^2)$ לעומת זאת, בגרף האחרון עם הערכים המאוד גדול הוא נשאר $O(n \log n)$, כי ב3 גרפים הראשונים יש טווח קטן יחסית. וכשמגיעים למערכים שהם יותר בטווח גדול יותר (יחסית), בעצם נקבל כל פעם הרבה "תתי-מערכים" ממויינים כי הם עם אותו ערך. ולכן, יש חשיבות קטנה יחסית לכך שבוחרים partition אקראי כי בסופו של דבר הוא יצטרך לעבור על כל איבר ואיבר (כל תא במערך) באותו חלק. ערך ה- partition תמיד יהיה שווה לאיבר המקסימלי של אותו מקטע.



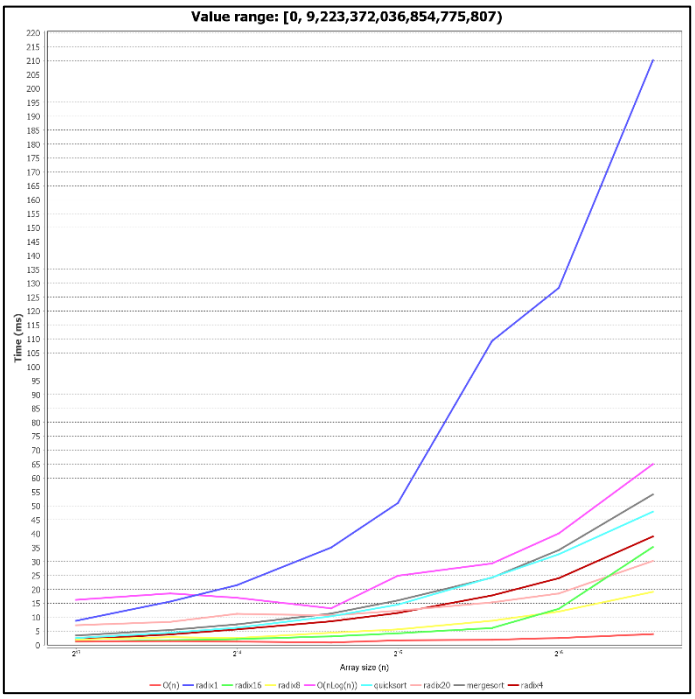
גרף מספר 2



גרף מספר 1



גרף מספר 3



גרף מספר 4