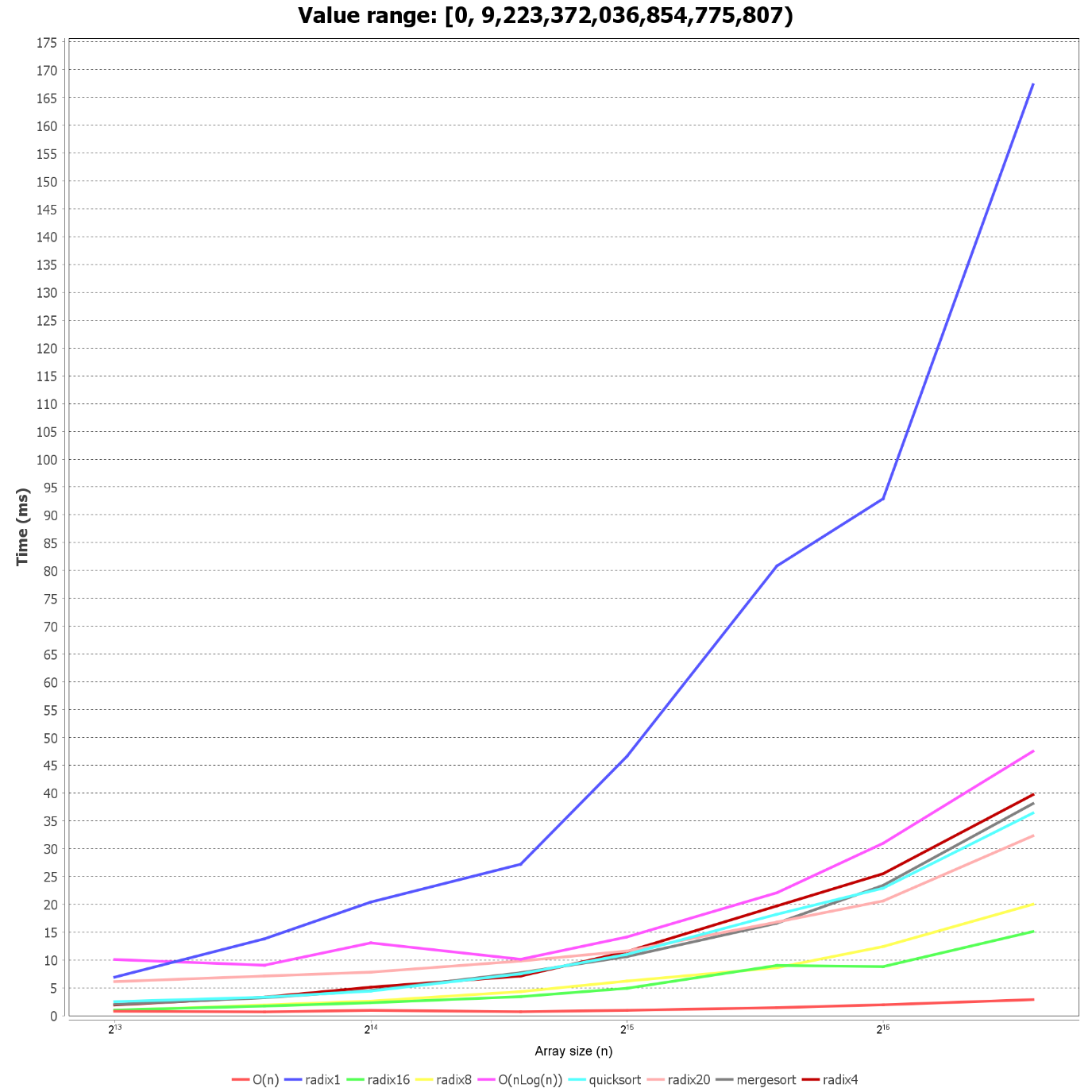
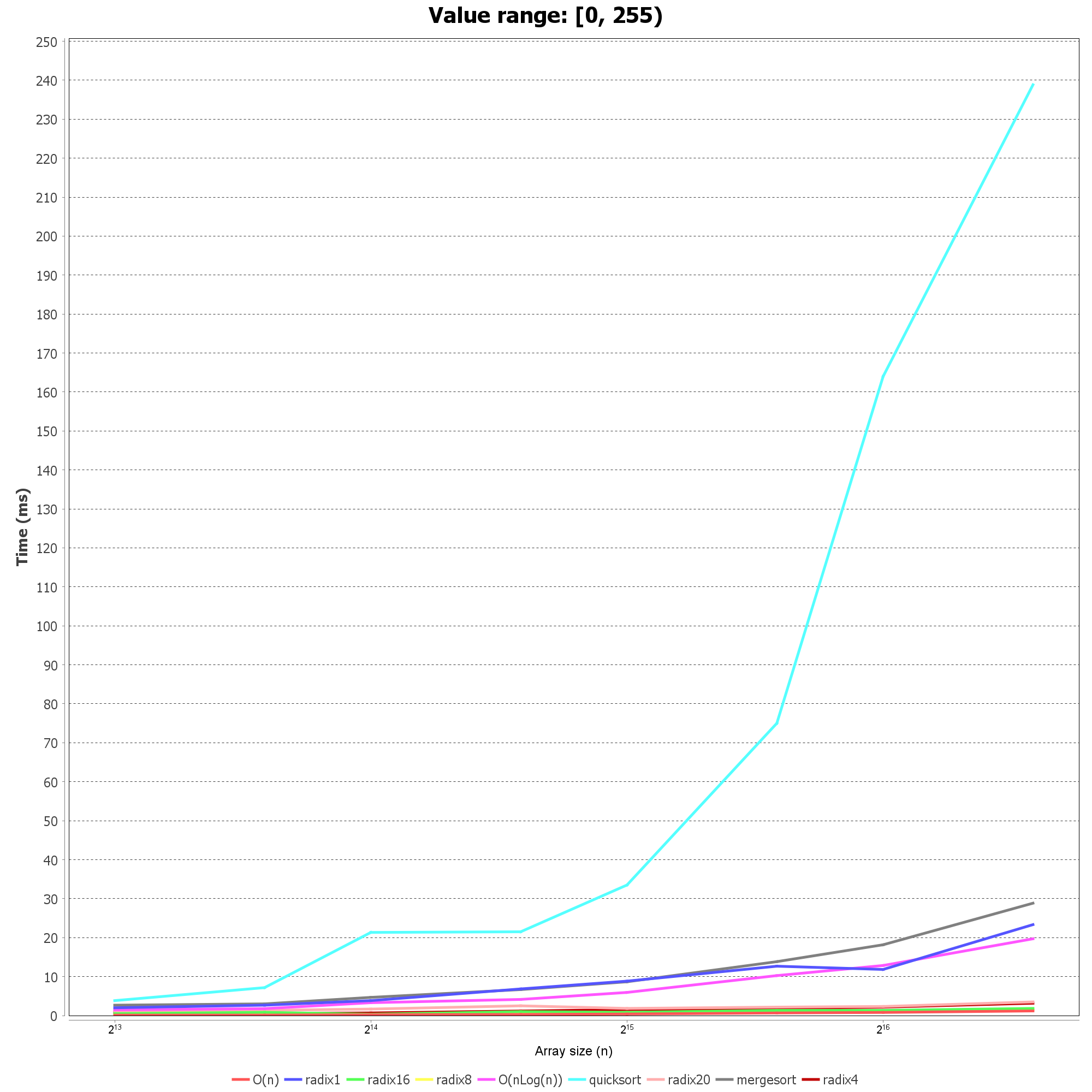
ראשית, נשים לב כי בכל המקרים האלה כל הגרפים שמבטאים את פונקציות המיון שלנו הן חסומות מלמטה ע"י אומגה של N, כיוון שבשביל למיין מערך אנו חייבים "לגעת" / להגיע" לכל איבר במערך לכן זה לפחות O(n).



נשים לב כי כאשר מדובר במערך כאשר טווח הערכים הוא גדול מאוד, כאשר נתבונן בגרף, נראה כי אכן המיון הכי "יקר" כלומר המיון שדורש הכי הרבה פעולות הוא RADIX 1 משום במיון radixSort אנו משתמשים בערכי האיברים בשביל למיין את את המערך, וליתר דיוק, אנו מחשבים את כמות הספרות של המספר המקסימלי במערך שהוא (מספר גדול מאוד) לפי בסיס בינארי, ובעצם נקבל מספר ספרות גדול מאוד כיוון במיון זה אנו משווים בין שדות של תחומים וכיוון שאנו משווין במיון כל ספרה במספר שמוצג בבינארית, כמות השדות תהיה גדולה, ולכן במיון בבסיס זה נבצע יותר פעולות מאשר במיון בבסיס שמותאם יותר למספרים, למשל מיון RADIX 16

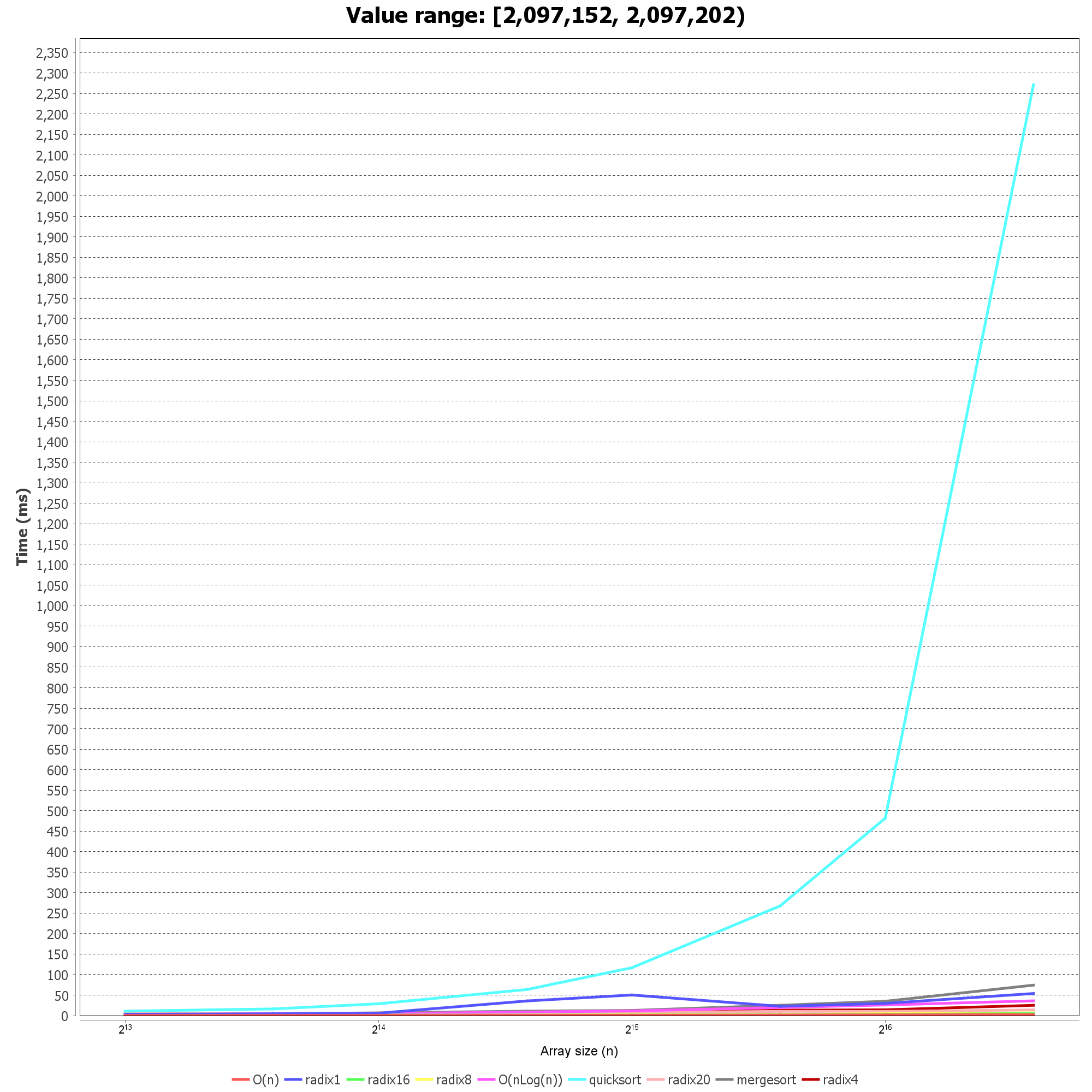
כאשר אנו מבצעים מיון בradixSort בבסיס 2 אז בכל פעם נמיין לחלוקה של 0 ו1, כלומר נשים את האיברים שהשדה שעליו אנו עובדים הוא 1 מעל האיברים שבשדה שלהם יש 0. אך כאשר המספרים גדולים וגם הבסיס שלו הוא גדול לדוגמא בradix 20 אז מיון בשדות המספרים יתחלקו ליותר "קטגוריות"/ פחות סיכוי שהאיברים יחזרו על עצמם ולכן המיון יתבצע בפחות פעולות.

נשים לי כי mergeSort וquickSort הם בסיבוכיות זמן ריצה של O(nlogn) .



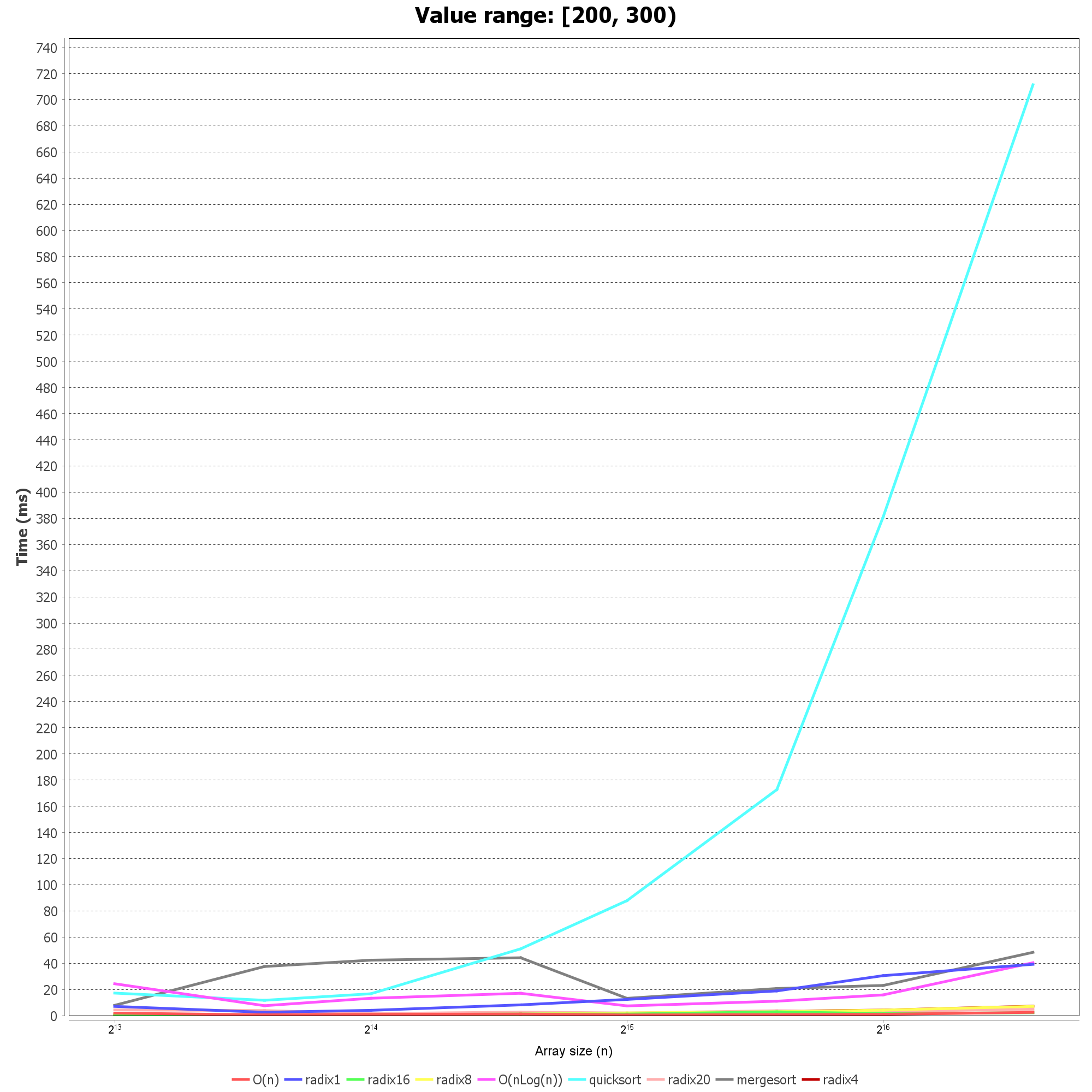
כאשר מדובר במיון ערכי המספרים במערך הם בטווח קטן והמספרים קטנים, אז נשים לי כי המיון ה"היקר" ביותר, כלומר בסיבוכיות זמן ריצה גדולה יותר הוא המיון quicksort כיוון שנשים לב שמקרה זה, כאשר n הוא מספר גדול מאוד, כלומר המערך גדול מאוד, מדובר במקרה הworst case של quicksort. זה מכיוון שבמקרה זה במערך יהיו הרבה מספרים שיחזרו על עצמם ולכן כאשר נבצע את המיון (כאשר מדובר בקלט גדול) זה כמו שנקבל מערך שכל האיברים בו הם אותו הדבר. כלומר במיון זה כאשר מדובר במערך גדול אז החלוקה של הפונצקיה Partition תחלק את המערך לאיברים שגדולים מpivot ולמספרים שקטנים מהpivot ולכן המיון יחלק את המערך לתתי מערכים, שבהם גם יהיו איברים זהים וכך הלאה, עד שנקבל תתי מערכים שבנויים מאיבר אחד, ובמקרה זה סיבוכיות זמן הריצה היא O(n^2).

נשים לב שכאן המיון באמצעות radix 1 כאשר מדובר במספרים קטנים הוא יותר מיון שמותאם יותר למערך, מאשר המקרה הקודם אך עדיין המיון בradixSort בבסיסים אחרים הוא טוב יותר כאשר המספרים הם בסדר גודל הזה, מכיוון שבמקרה זה המיון בבסיס שקרוב יותר לבסיס עשרוני למשל בסיס radix 4 שהוא בסיס 16(הקסדצימלי), כמות השדות שיהיו הן או 1 או 2, ולכן מספר השוואות תלוי במספר השדות ולכן מספר הפעולות יהיה קטן יותר.



נשים לב שבמיון זה המיון מתבצע באופן דומה למיון הראשון מבחינת radix 1.

ובנוסף גם כאן מדובר בטווח צר ולכן גם כאן הגרף דומה מבחינת quicksort.

באופו האופן – לפי ההסבר של הגרפים הקודמים.