מבוא למדעי המחשב

יעילות וסיבוכיות

יעילות של תוכנית

- כאשר תוכנית מחשב רצה פעולות לוקחות זמן (וגם משאבים אחרים כמו זיכרון)
- האם אנחנו צריכים לכתוב אותה כך שזמן הריצה יהיה קצר יותר?

יעילות של תוכנית

- כאשר תוכנית מחשב רצה פעולות לוקחות זמן (וגם משאבים אחרים כמו זיכרון)
- האם אנחנו צריכים לכתוב אותה כך שזמן הריצה יהיה קצר יותר?
  - לא תמיד – לפעמים אנחנו יכולים לדעת מראש שזמן הריצה לקלטים אפשריים הוא מספיק מהר על החומרה של היום.

יעילות של תוכנית

- כאשר תוכנית מחשב רצה פעולות לוקחות זמן (וגם משאבים אחרים כמו זיכרון)
- האם אנחנו צריכים לכתוב אותה כך שזמן הריצה יהיה קצר יותר?
  - לא תמיד – לפעמים אנחנו יכולים לדעת מראש שזמן הריצה לקלטים אפשריים הוא מספיק מהר על החומרה של היום.

מראש: לפני תכנות או אחרי תכנות?

סיבוכיות זמן ריצה

- רוצים דרך להשוות אלגוריתמים לפני שמתכנתים אותם ועל מנת שנוכל לקבל "סדר גודל" של זמן ריצה לפני התכנות.
  - חוסך זמן מימוש/תכנות לאלגוריתמים איטיים מדי
  - חוסך זמן מימוש/תכנות אם אין צורך לתכנת משהוא מהיר ומסובך שאין צורך בכך

חיפוש ערך מקסימלי במערך

```
int max_val(const vector<int>& vec) {
  int max = vec[0];
  for (size_t i = 1; i < vec.size(); ++i) {
    if (vec[i]>max) max = vec[i];
  }
  return max;
}
```

- איך נבדוק זמן ריצה? נספור פעולות?

חיפוש ערך מקסימלי במערך

- איך נבדוק זמן ריצה? נספור פעולות?
  - בעייה 1: מספר הפעולות תלוי בקלט ולא רק בגודלו:

```
if (vec[i]>max) max = vec[i];
```

  - פיתרון: "worst case analysis": שם יפה ל "מה קורה שאין מזל"
  - במקרה שלנו – האיבר המקסימלי הוא האיבר האחרון.

חיפוש ערך מקסימלי במערך

- איך נבדוק זמן ריצה? נספור פעולות?
  - בעייה 2: מספר פעולות של ביטוי כלשהוא הוא תלוי שפת תכנות, צורה בה מקמפלים, מחשב

```
vec[i];
```

- פתרון: נספור רק "סדר גודל" של מספר פעולות.
- כלומר, נשאיר רק את הביטויים שגדלים הכי מהר שגודל הקלט גדל ונוריד את המקדם.

חיפוש ערך מקסימלי במערך

```
int max_val(const vector<int>& vec) {
  int max = vec[0];
  for (size_t i = 1; i < vec.size(); ++i) {
    if (vec[i]>max) max = vec[i];
  }
  return max;
}
```

- סדר הגודל של מספר הפעולות: vec.size() פעולות

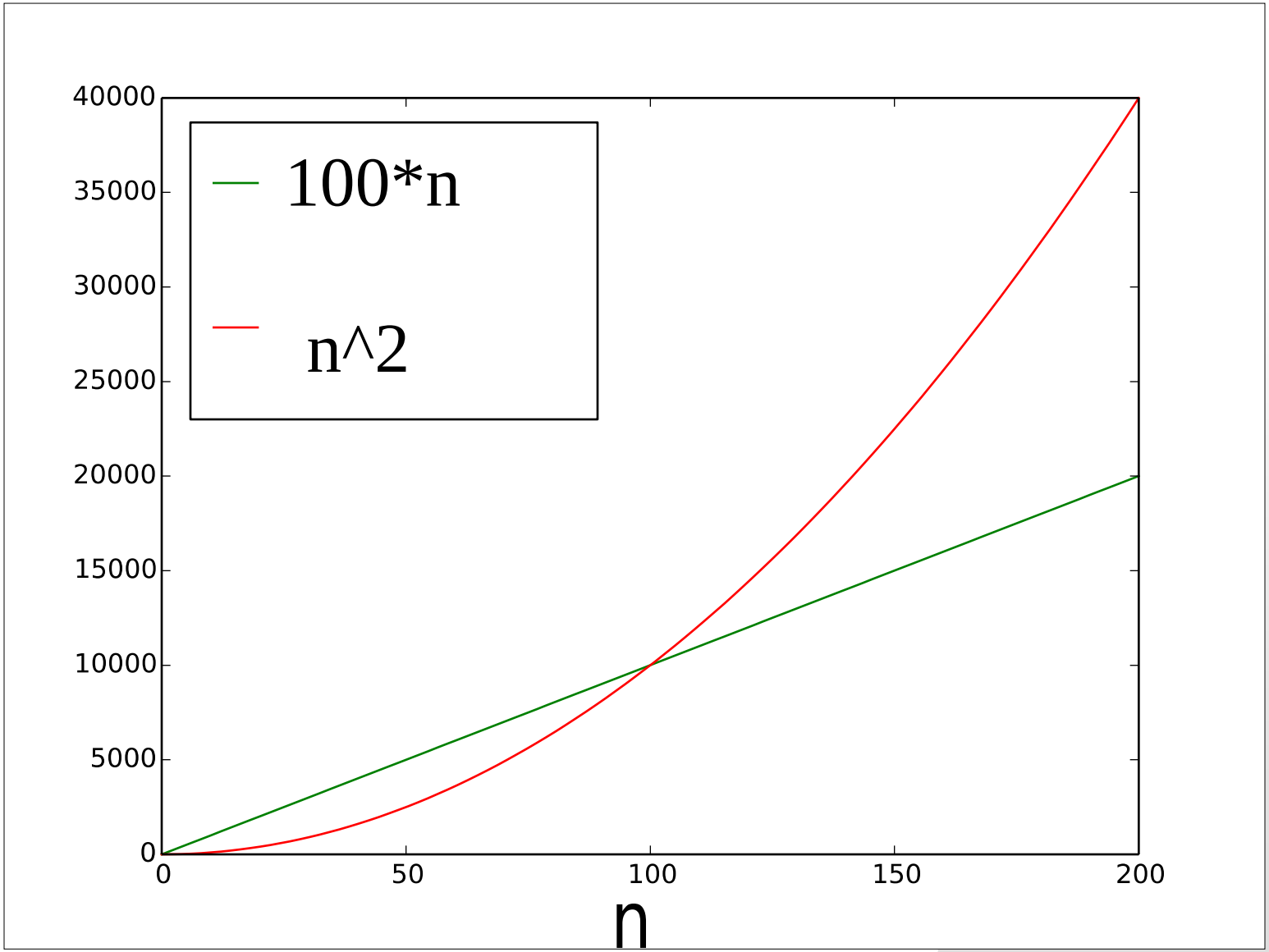
מיון

- קוד של שיעור 5
- האיטרציות של הלולאה ב sort לוקחות סדר גודל של זמן ריצה בהתאמה של:
- (size-1), (size-2), … , 1
- סכום הסידרה הזאת הוא:

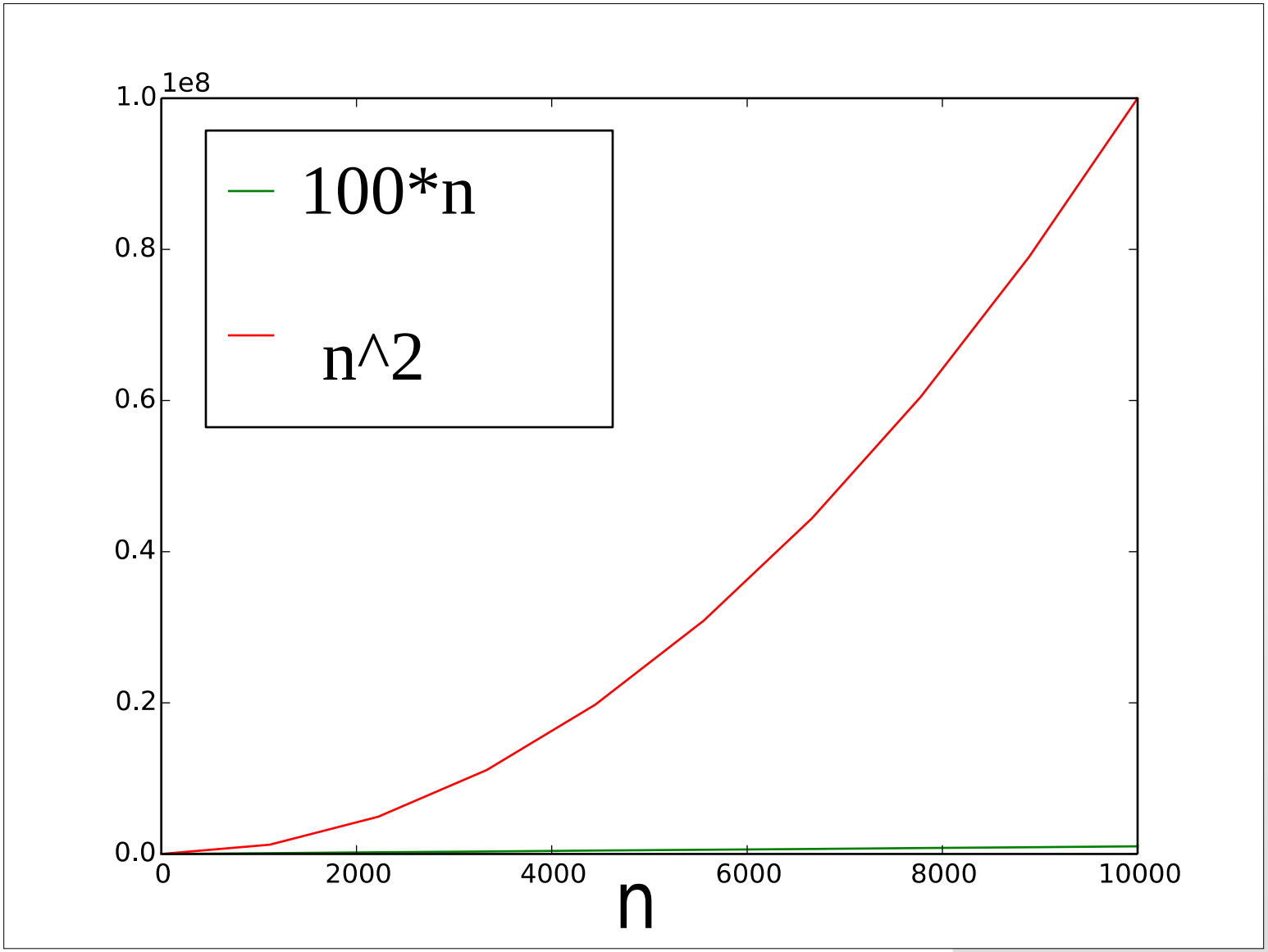
מיון

- קוד של שיעור 5
- האיטרציות של הלולאה ב sort לוקחות סדר גודל של זמן ריצה בהתאמה של:
- (size-1), (size-2), … , 1
- סכום הסידרה הזאת הוא: size*(size-1)/2
- סדר הגודל של זמן הריצה הוא: size^2
- האם אפשר למיין בזמן מהיר יותר?

למה סדרי גודל?



למה סדרי גודל?



מיון מהיר יותר – merge sort

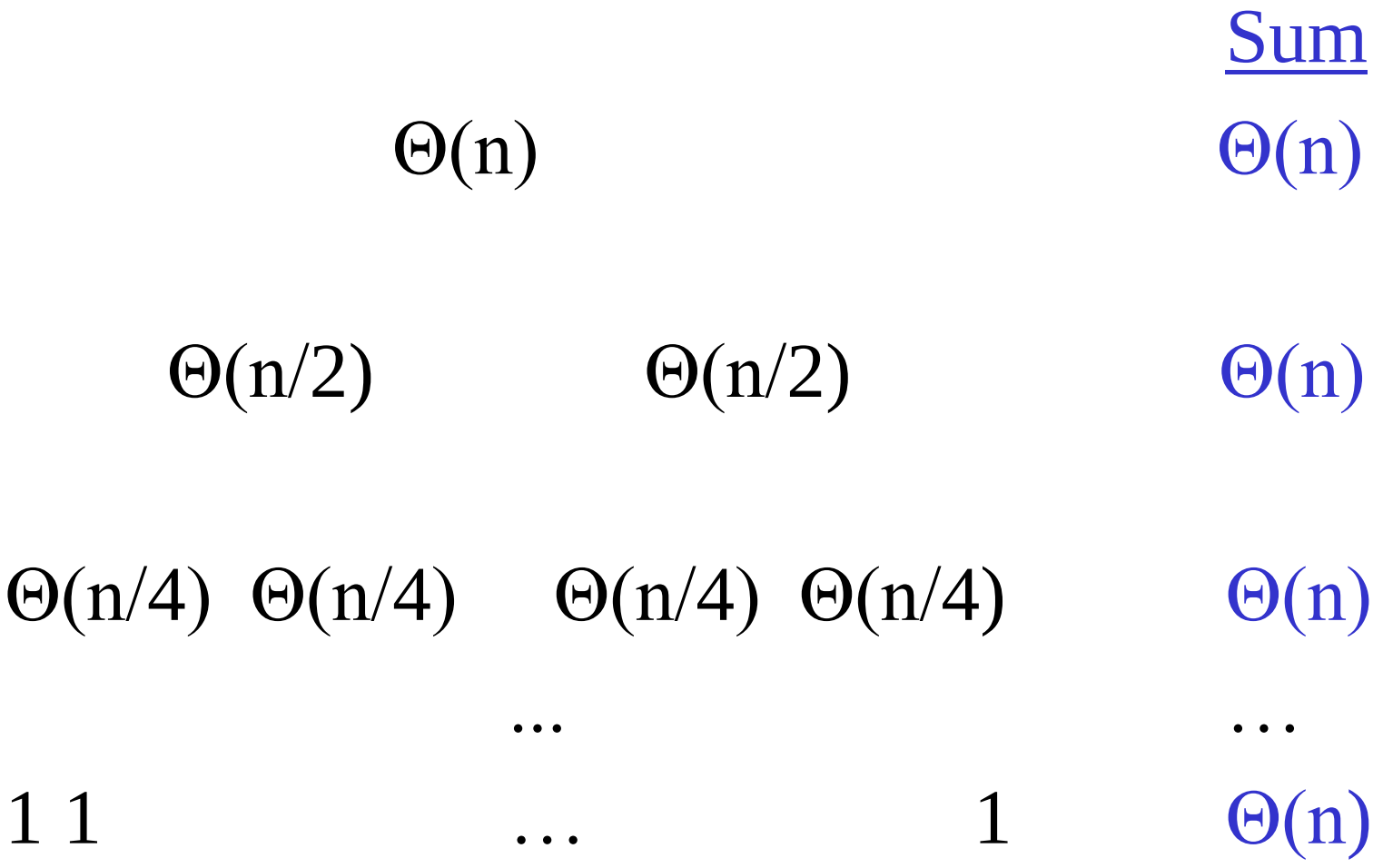
- הרעיון: נשתמש בפונקציה שתמזג (merge) 2 תתי מערכים ממויינים למערך ממויין אחד.
- איך נעשה את זה?
- מה זמן הריצה?

מיון מהיר יותר – merge sort

- עכשיו פשוט נחלק כל פעם את המערך לתתי מערכים קטנים עד שנגיע לתתי מערכים קטנים באורך 1 ואותם נמזג וכן הלאה...

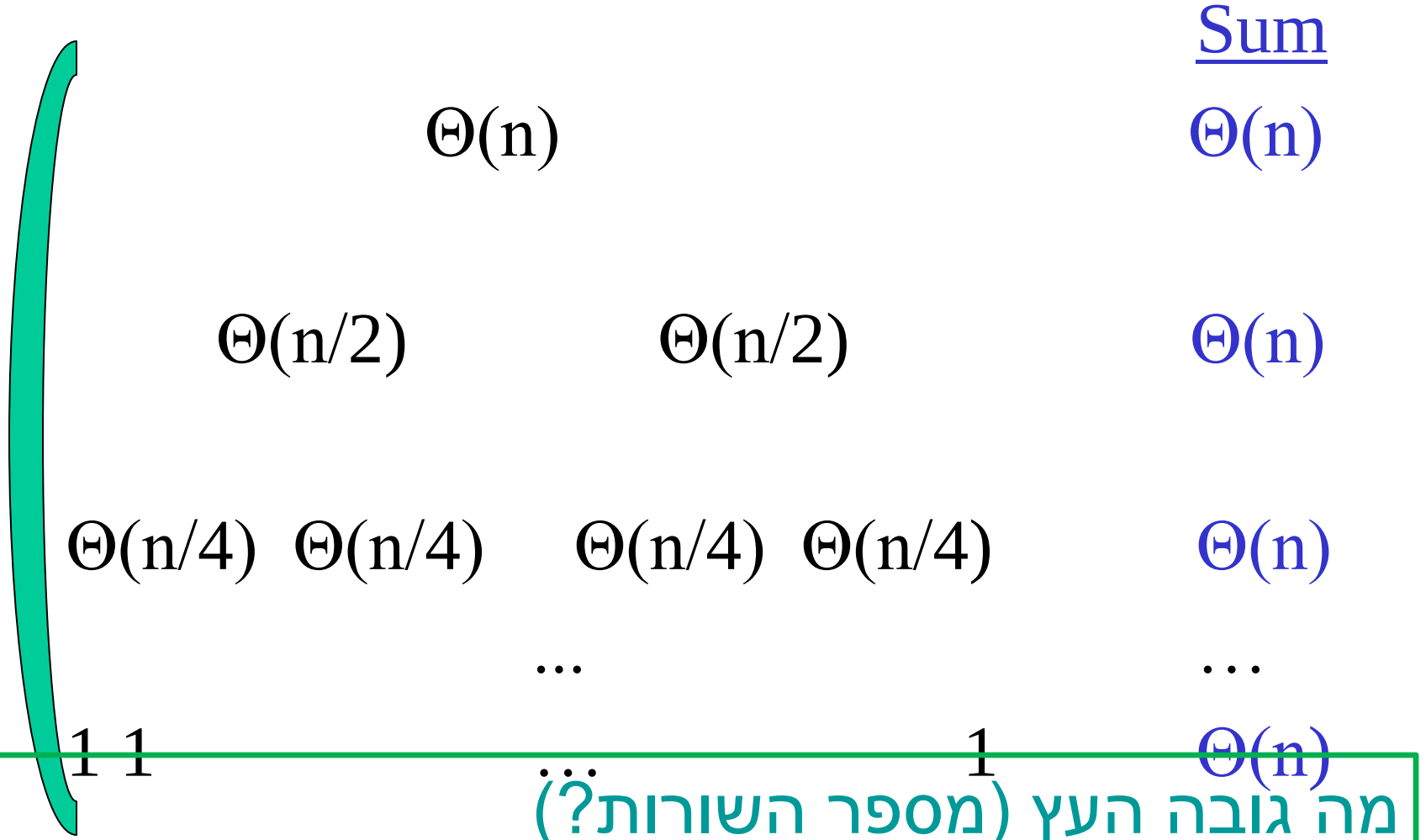
מיון מהיר יותר – merge sort

מיון מהיר יותר – merge sort



הערה (לא מדוייקת): Θ מסמל "סדר גודל"

מיון מהיר יותר – merge sort



מה גובה העץ (מספר השורות?)

כמה פעמים צריך להכפיל את 1 ב 2 ע"מ להגיע ל n?

מיון מהיר יותר – merge sort

מה גובה העץ (מספר השורות?)

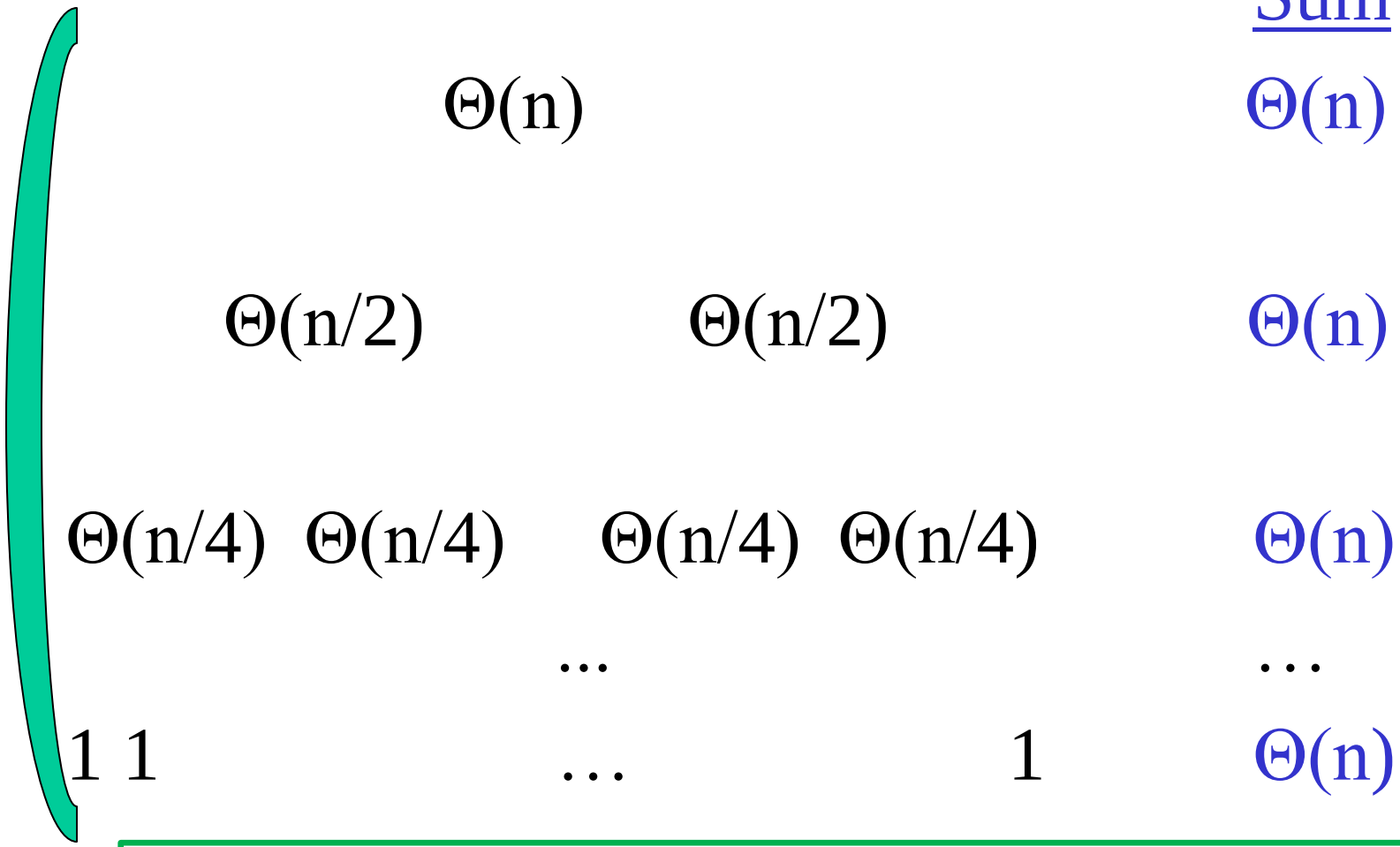
כמה פעמים צריך להכפיל את 1 ב 2 ע"מ להגיע ל n?

$2^x=n$

➔

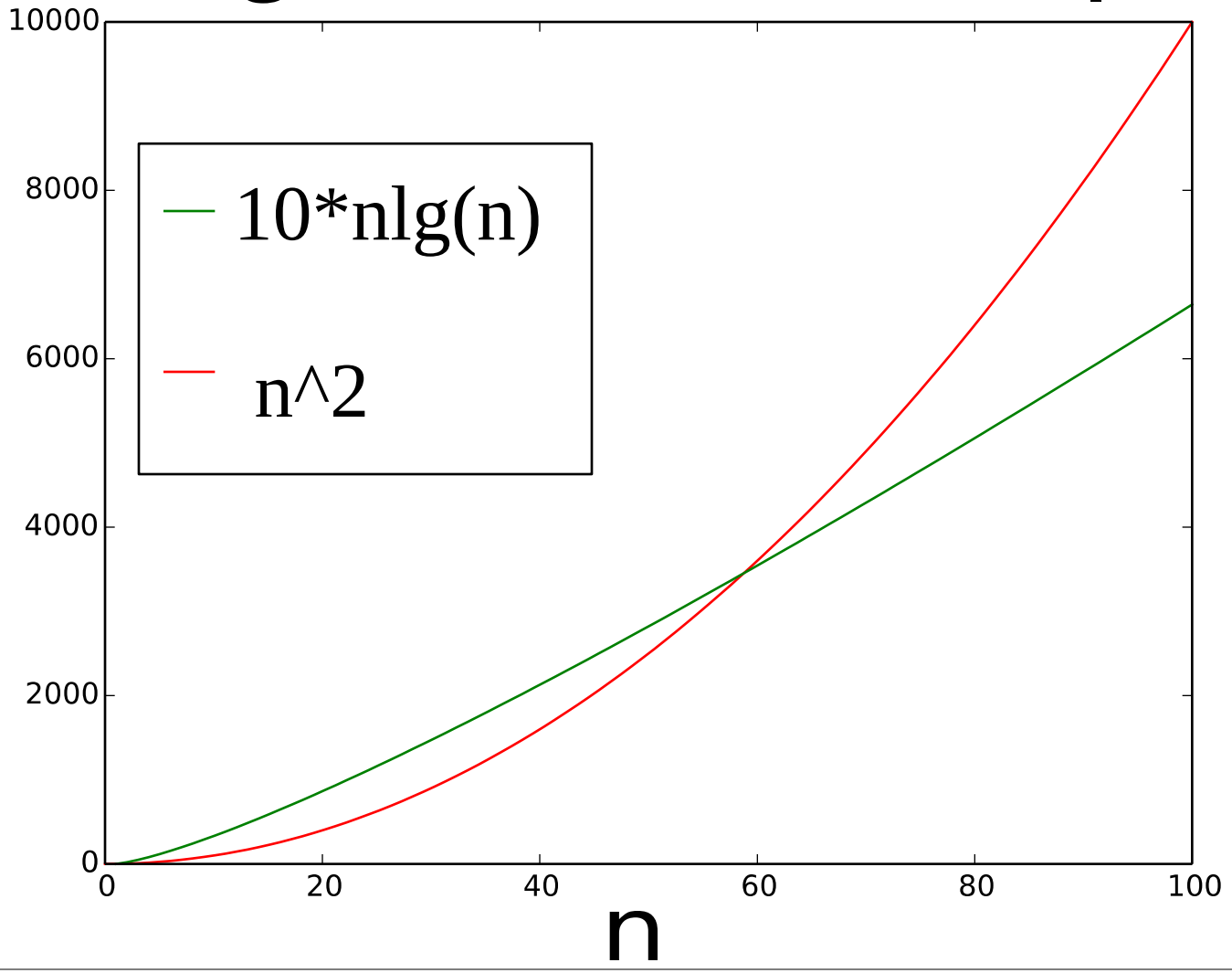
$x=\lg(n)$ ← lg is the logarithm with base 2

מיון מהיר יותר – merge sort

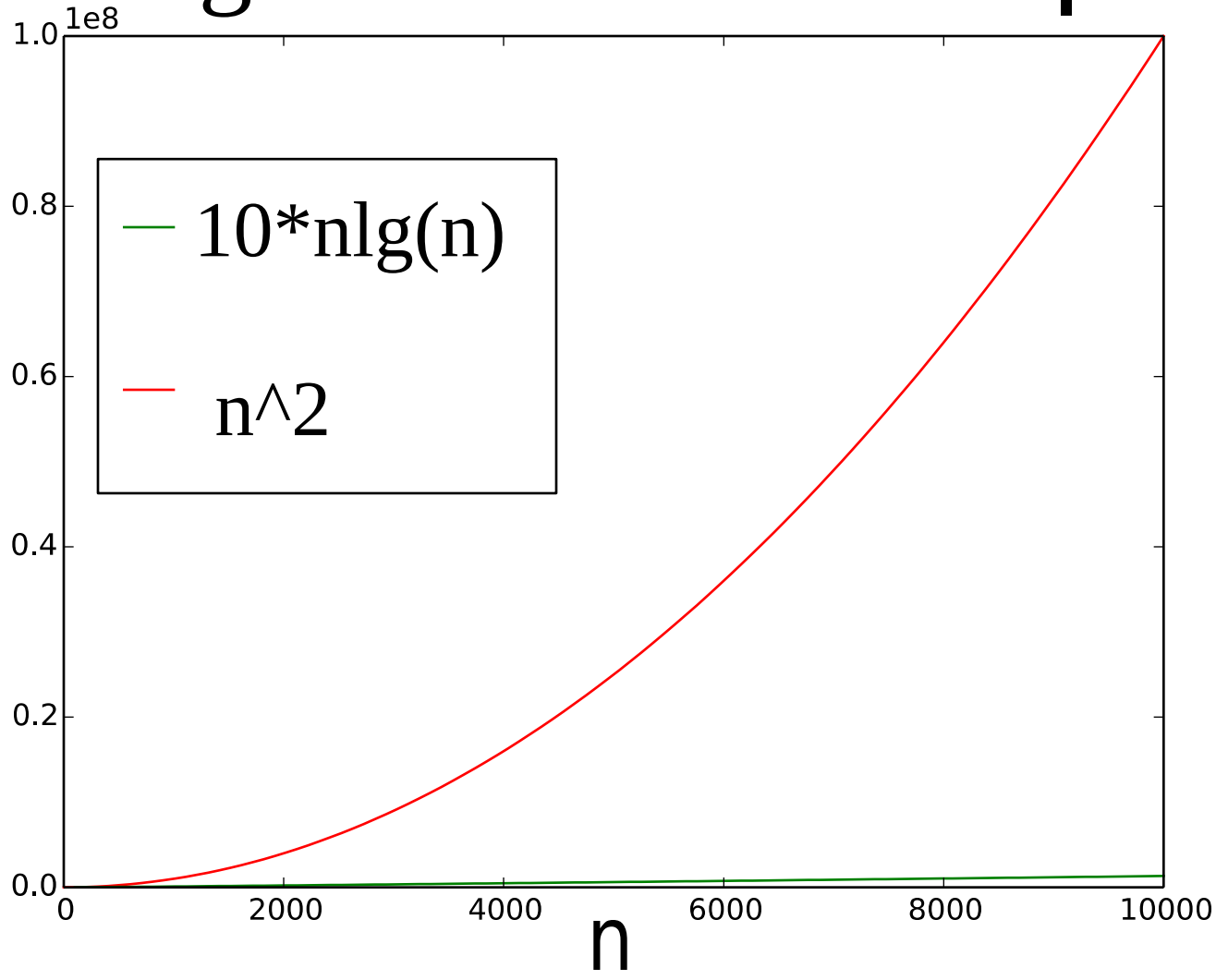


זמן הריצה הכולל של האלגוריתם: Θ(nlg(n))

merge sort – מיון מהיר יותר



merge sort – מיון מהיר יותר



מיון מהיר יותר – merge sort

- נניח שאנחנו צריכים למיין מיליון מספרים
- מיון בחירה יצטרך כפולה של $(10^6)^2$ פעולות שווה ל: **1,000,000,000,000** פעולות
- מיון מיזוג יצטרך כפולה של $10^6(\lg 10^6)$ פעולות שווה ל: **20,000,000** פעולות.

מיון מהיר יותר – merge sort:
קוד

טיפים למתקדמים

- לבדוק אם יש קוד שממש כבר את מה שצריך (נדיר) או ממש פונקציות שאני צריך להשתמש בהן (מאד נפוץ)
- לבדוק אם בשביל גודל הקלט הצפוי יש בעייה בכלל להריץ את האלגוריתם.
- בויז'ואל סטודיו: להעביר מצב קומפילציה ל Release mode
- אפשר גם לבדוק התאמת קימפול לחומרה הספיציפית וflags מיוחדים לקומפיילר.
- טכניקת ייעול מאד נפוצה: לזכור תוצאות חישוב ביניים.
- אם עדיין צריך להאיץ את התוכנית ולא מוצאים משהוא יעיל יותר בסדר גודל – צריך למדוד זמני ריצה ולחפש צוואר בקבוק (profiling)