Lower bound for sorting
if the only operations allowed on the input numbers is comparisons
then such an algorithm must take at least $\Omega(n \lg n)$ time

All the algorithms we saw so far use only comparisons

If we do not have any information about the kinds of values that may be on the
input, the only operation on the inputs that is legitimate to allow our algorithm
to perform is comparisons. Thus, with no knowledge of the type of input, we
cannot sort better than in $\Omega(n \lg n)$ time in the worst-case

Next time we’ll see some algorithms that take advantage of special facts
known about the inputs and achieve running time that are less than $n \lg n$
Best-case partitioning: the two arrays are balanced
\[ T(n) = 2T(n/2) + \Theta(n) = \Theta(n \lg n) \] by master theorem

Average-case partitioning:
suppose partition always produced 9/10 - 1/10 split
then \[ T(n) = T(9n/10) + T(n/10) + \Theta(n) = \Theta(n \lg n) \] by iteration method
so, this is still a good split

But, on average, we won’t always have such guarantee
some splits will be good and some splits will be bad

How to increase our chances for good splits?
Randomized-Partition(A,p,r)
   i=Random(p,r)
   exchange A[p] with A[i]
   Partition(A,p,r)

Expected split by the Randomized-Partition partition is good
Expected running time of Randomized-Quicksort is \( \Theta(n \lg n) \)
Running time of Quicksort: depends on how the array is partitioned

\[ T(n) = T(q) + T(n-q) + \Theta(n) \]

Worst-case partitioning: the two arrays are as unbalanced as possible
one contains 1 element, the other contains \( n-1 \) elements

\[ T(n) = T(n-1) + T(1) + \Theta(n) \]

\[ T(n) = T(n-1) + \Theta(n) \]

Using iteration method:

\[ T(n-1) = T(n-2) + \Theta(n-1) \]

\[ T(n) = [T(n-2) + \Theta(n-1)] + \Theta(n) \]

\[ T(n) = [[T(n-3) + \Theta(n-2)] + \Theta(n-1)] + \Theta(n) \]

\[ = [[[T(n-4) + \Theta(n-3)] + \Theta(n-2)] + \Theta(n-1)] + \Theta(n) \]

\[ = T(n-5) + \Theta(n-4) + \Theta(n-3) + \Theta(n-2) + \Theta(n-1) + \Theta(n) \]

\[ = T(n-5) + \Theta(n-4) + \Theta(n-3) + \Theta(n-2) + \Theta(n-1) + \Theta(n) \]

\[ = T(n-k) + \Theta(n-(k-1)) + \Theta(n-(k-2)) + \ldots + \Theta(n-1) + \Theta(n) \]

\[ T(n) = T(1) + \sum_{k=n-2}^{0} \Theta(n-k) = \Theta(1) + \sum_{i=2}^{n} \Theta(k) = \Theta \left( \sum_{i=2}^{n} k \right) \]

\[ T(n) = \Theta \left( n^2 \right) \]
Running time of Partition: $\Theta(n)$
Lecture 5: Quicksort

Quicksort(A)

1) Divide: partition (divide) array A[p...r] into 2 nonempty subarrays A[p..q] and A[q+1..r] such that each element of A[p..q] is less than or equal to each element of A[q+1..r]
2) Conquer: recursively sort A[p..q] and A[q+1..r]
3) Combine: done

Partition(A,p,r)

\[
x = A[p] \\
i = p - 1 \\
j = r + 1 \\
\text{while true} \\
\quad \text{repeat } j = j - 1 \text{ until } A[j] \leq x \\
\quad \text{repeat } i = i + 1 \text{ until } A[i] \geq x \\
\quad \text{if } i < j \text{ then exchange } A[i] \text{ with } A[j] \\
\quad \text{else return } j
\]