

Lower Bounds for Comparison Sorting

Sorting returns a permutation of its input.

There are $n!$ permutations of n elements.

$a_i \leq a_j$ determines if a_i is before/after a_j .

$a_i \leq a_j$ chooses between two sets of permutations.

Map to a binary *decision* tree.

Map each comparison to a node in a tree.

Map sets of permutations to subtrees.

Map singleton sets to leaves.

A binary tree with height h has $\leq 2^h$ leaves.

A binary tree with $n!$ leaves has $h \geq \lg(n!)$

$\lg(n!) \in \Theta(n \lg n)$, so $h \in \Omega(n \lg n)$.

Worst-case of comparison sorting is $\Omega(n \lg n)$.

$\leq 2^d$ leaves at depth $\leq d$.

if $d < \lg(n!) - 1$, then $2^d < n!/2$

$\geq n!/2$ leaves have $d \geq \lg(n!) - 1$

Average-case of comparison sorting is $\Omega(n \lg n)$.

COUNTING-SORT is $\Theta(n + k)$ and stable.
 It assumes that each $A[j] \in \{1, 2, \dots, k\}$.

COUNTING-SORT(A, B, k)

$C \leftarrow$ an array of k zeros

for $j \leftarrow 1$ **to** $length[A]$

do $C[A[j]] \leftarrow C[A[j]] + 1$

▷ $C[i]$ is the number of elements equal to i

for $i \leftarrow 2$ **to** k

do $C[i] \leftarrow C[i] + C[i - 1]$

▷ $C[i]$ is the number of elements $\leq i$

for $j \leftarrow length[A]$ **downto** 1

do $B[C[A[j]]] \leftarrow A[j]$

$C[A[j]] \leftarrow C[A[j]] - 1$

$A = (3, 2, 1, 2)$

$C = (1, 2, 1)$ after first loop

$C = (1, 3, 4)$ after second loop

$j = 4, B = (\square, \square, 2, \square), C = (1, 2, 4)$

$j = 3, B = (1, \square, 2, \square), C = (0, 2, 4)$

$j = 2, B = (1, 2, 2, \square), C = (0, 1, 4)$

$j = 1, B = (1, 2, 2, 3), C = (0, 1, 3)$

RADIX-SORT is $\Theta(d(n + k))$.

It assumes that each value has d digits.

Each digit has one of k values.

RADIX-SORT(A, d)

for $i \leftarrow 1$ **to** d

do use COUNTING-SORT to sort A
 on digit i

238	230	230	045
796	934	934	230
756	045	537	238
045 \Rightarrow	796 \Rightarrow	238 \Rightarrow	537
537	756	045	756
230	537	756	796
934	238	796	934
	\uparrow	\uparrow	\uparrow

Assuming k is $O(n)$, RADIX-SORT will outperform $\Theta(n \lg n)$ if d is $o(\lg n)$.

BUCKET-SORT is $\Theta(n)$ on average.

Assumes uniform distribution over the interval $[0, 1)$

BUCKET-SORT(A)

$n \leftarrow \text{length}[A]$

$B \leftarrow$ an array of n empty lists

for $i \leftarrow 1$ **to** n

do insert $A[i]$ into list $B[\lfloor n A[i] \rfloor]$

for $i \leftarrow 0$ **to** $n - 1$

do sort list $B[i]$ with INSERTION-SORT
concatenate the lists $B[0]$ to $B[n - 1]$

Average Case:

There are n elements and n buckets.

Let n_i be number of elements in bucket i .

INSERTION-SORT on n_i elements is $O(n_i^2)$.

Need to bound $E[n_i^2]$ (expected value of n_i^2).

n_i is binomial with prob. $p = 1/n$ and n trials.

$$E[n_i] = np = 1$$

$$\text{Var}[n_i] = np(1 - p) = 1 - 1/n$$

$$\text{Var}[n_i] = E[n_i^2] - E[n_i]^2 \text{ implies } E[n_i^2] < 2$$

Expected time of second loop is $\Theta(n)$.