Name:____

8.1 Lower bounds for sorting

In a comparison sort, we use only comparisons between elements to gain order information about an input sequence $< a_1, a_2, \ldots, a_n >$. Without loss of generality, we perform tests like:

- $a_i \leq a_j$

So far, all sorts we've considered (insertion sort, selection sort (on midterm), merge sort, quicksort, heapsort) are worst-case $\Omega(n \lg n)$.

We can view comparison sorts abstractly in terms of decision trees.

Decision tree

- Abstraction of any comparison sort.
- Represents comparisons made by
  - a specific sorting algorithm
  - on inputs of a given size.
- Abstracts away everything else: control and data movement.
- We're counting *only* comparisons.

Recall the insertion sort algorithm:

Listing 1: insertion sort

```
INSERTION-SORT(A,n)
  for j = 2 to n
       key = A[j]
       //Insert A[j] into the sorted sequence A[1..j-1]
       i = j - 1
       while i > 0 and A[i] > key
           A[i+1] = A[i]
           i = i - 1
       A[i+1] = key
```

1. **At your boards:** Illustrate the operation of INSERTION-SORT on the array $A = \langle 6, 8, 5 \rangle$. Hint: Figure 2.2 page 18.

Consider the decision tree for insertion sort on 3 elements:

https://www.lucidchart.com/invitations/accept/4f21cbaf-bce9-49e1-b293-ea9f0cc8020e

2. **At your boards:** Write the decision tree for insertion sort on 3 elements. Identify/highlight the path corresponding to the decisions made when insertion-sorting $A = \langle 6, 8, 5 \rangle$.

3. **At your boards:** How many leaves are on the general insertion-sort decision tree when sorting $A[1..n]$? Explain briefly.

For any comparison sort,

- We get one tree for each n.
- View the tree as if the algorithm splits in two at each node, based on the information it has determined up to that point.
- The tree models all possible execution traces.

Lemma Any binary tree of height h has $\leq 2^h$ leaves. In other words:

- $l = \#$ of leaves
- $h =$ height
- then $l \leq 2^h$

(We'll prove this lemma later.) Why is this useful?

Theorem Any decision tree that sorts n elements has height $\Omega(n \lg n)$.
Proof

- the number of leaves $l \geq n!$ by Question 3
- $n! \leq l \leq 2^h$ by lemma, so $2^h \geq n!$
- taking logs: $h \geq \lg(n!)$
- Using Stirling's Approximation (WS04 Section 3.2): $n! > (n/e)^n$ so

$$\begin{aligned} h &\geq \lg(n/e)^n \\ &= n \lg(n/e) \\ &= n \lg n - n \lg e \\ &= \Omega(n \lg n) \quad \square \text{ theorem} \end{aligned}$$

Now prove the lemma:

Proof By induction on h.
Basis: $h = 0$: Here the tree is just one node, which is a leaf (and the root). So $2^h = 1$ and $l = 1 \leq 2^h$.
Inductive step: Assume true for height $= h - 1$: $l \leq 2^{h-1}$. Extend the tree of height $h - 1$ by making as many new leaves as possible. Each existing leaf becomes the parent to two new leaves. So

$$\begin{aligned}
\#\text{ of leaves for height } h &= 2 \cdot (\#\text{ of leaves for height } h - 1) \\
&\leq 2 \cdot 2^{h-1} (\text{ind. hypothesis}) \\
&= 2^h \quad \square \text{ lemma}
\end{aligned}$$

Corollary Heapsort and merge sort are asymptotically optimal comparison sorts.
Proof The $O(n \lg n)$ upper bounds on the running times for heapsort and merge sort match the $\Omega(n \lg n)$ worst-case lower bounds from the Theorem.