LECTURE 2

MERGESORT & FIRST ALGORITHM ANALYSIS
LECTURE 2
FIRST COMP. PROBLEM: SORTING
FIRST ALG. ANALYSIS

Sorting: Input: 10 12 37 15 9 55 20
Output: 2 5 9 10 20 37 55

Naive Algorithm: Selection Sort
for i = 1, ..., n -> 1 op, \( \text{Total of n} \)
let \( \text{minIndex} = i \) -> 1 op, \( \text{Total of n} \)
for j = i+1, ..., n -> 1 op
if \( X[j] < X[\text{minIndex}] \) -> 1 op
\( \text{Total of } n \cdot \frac{n-1}{2} \) \( \leq \) 8 op.
\( \text{minIndex} = j \) -> 1 op
\( X[i] \leftarrow \text{swap}(X[\text{minIndex}], X[i]) \) -> 1 op
\( \text{Total of } n \cdot \frac{n}{2} \leq 3 \sum_{i=1}^{n-1} i = 3 \frac{(n-1)n}{2} \)

Q: How many ops does Ssort take?

Total:
\( n + n + \frac{3(n-1)n + n}{2} \)
\( = \frac{3(n-1)n + 3n}{2} \ = \frac{3n^2 + 3n}{2} \)

\( i = 1 \Rightarrow \leq 3(n-1) \)
\( i = 2 \Rightarrow \leq 3(n-2) \... \)
Criticisms:

1: Loop increment is not 1 but 20 ps.
2: Swap is not 4 but 3 ops.
3: At machine level swap is 200 ops.

In CS 341 we will remain high level in our counting, and ignore these constants.

Top \equiv 1 \text{ high-level op.}

(Will make formal with Big-oh notation.)
Alg 2: Divide & Conquer

MergeSort (A of length n):
0. if A.size == 1, return A
1. L = mergeSort (A[1...\( \lfloor n/2 \rfloor \])
2. R = mergeSort (A[\( \lceil n/2 \rceil \)...n])
3. combine step; return merge(L,R)

Q: How many ops does mergeSort take?
Easier Q: How many ops does merge subroutine take on an input of size m?
Total: \( m+2+4m = 5m+2 \leq \frac{7m}{2} \) (true for all m)

merge (L,R of length \( \frac{m}{2} \))
out2 empty array of size m
\( i=1; j=1 \rightarrow 2\text{op} \)
for \( k=1 \ldots m \rightarrow 1\text{op} \)
if \( L[i] < R[j] \rightarrow 1\text{op} \)
out[k] = L[i] \rightarrow 1\text{op}
else
out[k] = R[j] \rightarrow 1\text{op}

hops per loop.
Claim: MSort takes $\leq 7n \log_2 n + 7n$ ops.

Proof:

Level 0: $M(n) \leq 7n \rightarrow \exists n$

Level 1: $M\left(\frac{n}{2}\right) \leq \frac{7n}{2}$

Level 2: $M\left(\frac{n}{4}\right) \leq \frac{7n}{4}$

Level 3: $M\left(\frac{n}{8}\right) \leq \frac{7n}{8}$

Level 4: $M\left(\frac{n}{16}\right) \leq \frac{7n}{16}$

Level 5: $M\left(\frac{n}{32}\right) \leq \frac{7n}{32}$

Level 6: $M\left(\frac{n}{64}\right) \leq \frac{7n}{64}$

Level 7: $M\left(\frac{n}{128}\right) \leq \frac{7n}{128}$

Level 8: $M(1) = 7$
Consider level $j$: # subproblems $2^j$ or MS calls
Size of inputs to subproblems at level $j$: $\frac{n}{2^j}$
Q: What’s the work done at level $j$?

$$\leq 2^j \cdot \left( \frac{7n}{2^j} \right) = \frac{7n}{2}$$

Since there are $\log_2 n + 1$ levels
The total # ops that MSORT takes is
$$\leq 7n \left( \log_2 n + 1 \right) = 7n \log_2 n + 7n$$

\[ \square \]
Assumptions (& Justifications) of CS 341 Analyses

1. Worst-case Runtime Analysis:
   (i.e., our runtime statements hold for every input).

2. We are not going to be very strict with our country, & focus on high-level ops
   - easier
   - impossible to agree on low-level ops
   and easier to do so for high-level ops.
Interested in very large inputs: mathematically:

\[
3n^2 + 3n \geq \log_2 n + 7n
\]

This statement is inaccurate.

\[
\frac{3n^2 + 3n}{\log_2 n + 7n} \geq 2
\]

but for large enough \( n \), it is always true, i.e.: 

\[
\frac{3n^2 + 3n}{\log_2 n + 7n} \geq 2
\]
RAM model of computation
Assume data is in memory and accessible with random-access memory.