Merge Sort

Recall our sorting method from before:

① Find location of smallest element in $A[0..n-1]$.

② Swap with first element ($A[0]$).

③ Repeat above, but with $A[1..n-1]$

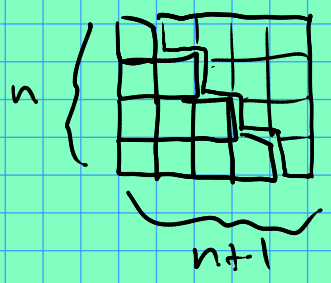
What is the cost in terms of n? (I.e., the approx. # of steps.)

It's $\approx n^2$. Why?

① costs $\approx n$ the first time, then $n-1$, $n-2$ ... 1.

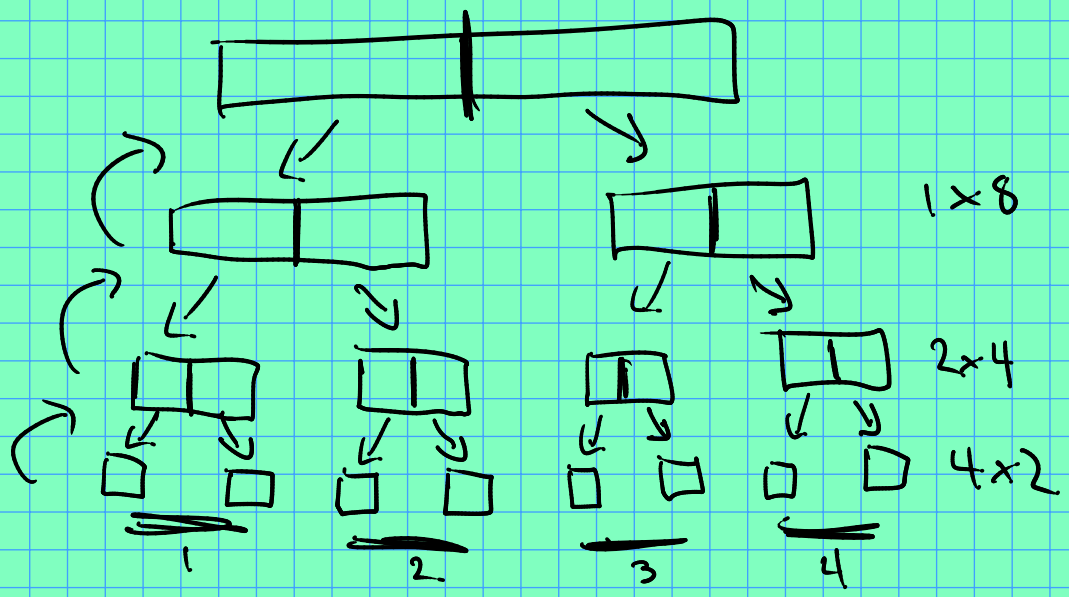
So total cost for ① is

$$1+2+3+\cdots n = \sum_{i=1}^{n} i = \frac{n(n+1)}{2} \approx n^2.$$

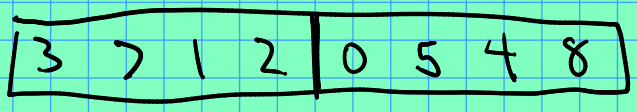


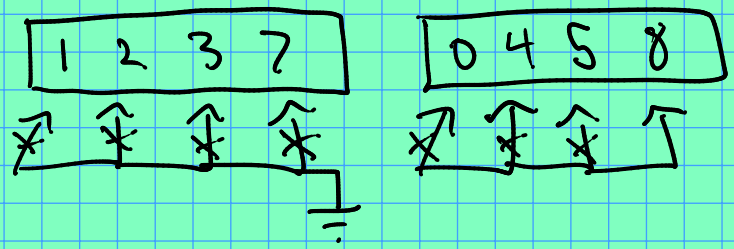
Can we sort with fewer steps?

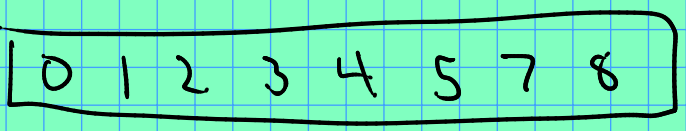
An idea: Break array in half, sort each sub-array, then stick them back together (merge them).



Example:







Steps for merge?

$$\approx n$$

So what's the total cost?

Merging an entire level into the one above always costs $\approx n$ steps.

Levels = # times we can divide n by 2 before getting 1. I.e.,

$$\frac{n}{2^l} = 1 \Rightarrow n = 2^l$$

So, $l = \log_2 n$.

$\therefore$ total cost $\approx n \cdot \log_2 n$

n: cost of 1 level

$\log_2 n$: # levels.