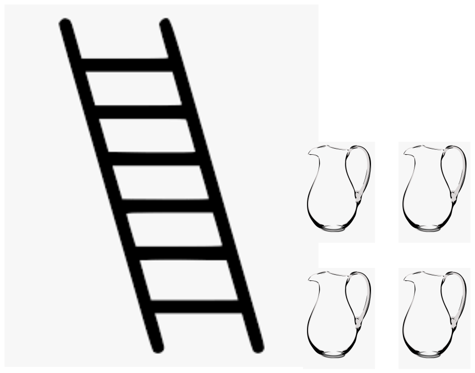
Notes, 5(d)

ECE 606

Another example

A final example in the lectures of divide-n-conquer, that does not involve $\lg n$.

Suppose we had a ladder of n rungs, and k identical jugs. And we want to find the "highest safe rung" of the ladder — the highest rung from which, if we drop a jug, the jug does not break. How would we go about it?



If $k = 1$, linear-search starting with the lowest rung seems to be the best option. This costs n drops of the jug, in the worst-case.

What if $k = 2$? That is, can we exploit the fact that we have a "budget" of two jugs to converge faster to the highest safe rung? Where "faster" means with fewer drops of a jug.

Here's an approach that is divide-n-conquer. Perceive the ladder as being comprised of $\sqrt{n}$ sequences each of $\sqrt{n}$ rungs. With one jug, we converge to the highest safe $\sqrt{n}$-sized sequence of rungs. That is, we drop the jug from rung # $\sqrt{n}$ from the bottom. Then from rung # $2\sqrt{n}$ from the bottom. And so on till it breaks.

Once we narrow down to a sequence of rungs of size $\sqrt{n}$, we perform linear search with the other jug. Our worst-case total number of drops $= \sqrt{n} + \sqrt{n} = 2\sqrt{n}$. And $\lim_{n\to\infty} \frac{2\sqrt{n}}{n} = 0$.

That is, asymptotically, our performance is strictly better with 2 jugs than 1.

Given k jugs we do the following.

We perceive the ladder as comprised of $n^{1/k}$ sequences each of $n^{\frac{k-1}{k}}$ rungs. Once we narrow down to $n^{\frac{k-1}{k}}$ rungs with a jug, we again perceive those rungs as comprised of $n^{1/k}$ sequences each of $n^{\frac{k-2}{k}}$ rungs. With our second jug, we narrow down to a sequence of $n^{\frac{k-2}{k}}$ rungs. And so on. With our second-last jug, we narrow down to a sequence of $n^{1/k}$ rungs, and with our last jug, to the highest safe rung.

For example, if $n = 243$ and $k = 5$, we narrow down with each jar as follows:
$243 \rightarrow 81 \rightarrow 27 \rightarrow 9 \rightarrow 3 \rightarrow 1$. Because $(243)^{1/5} = 3$.

Thus, our worst-case number of drops is $k \cdot n^{1/k}$. And with each additional jug we are provided, our performance gets strictly better. Because:

$$\begin{aligned}
\lim_{n\to\infty} \frac{k \cdot n^{1/k}}{(k-1)n^{1/(k-1)}} &= \frac{k}{k-1} \lim_{n\to\infty} \frac{n^{1/k}}{n^{1/(k-1)}} \\
&= \frac{k}{k-1} \lim_{n\to\infty} n^{\left(\frac{1}{k}-\frac{1}{k-1}\right)} \\
&= \frac{k}{k-1} \lim_{n\to\infty} n^{\left(\frac{-1}{k(k-1)}\right)} \\
&= 0.
\end{aligned}$$

Unlike prior examples, e.g., binary-search and merge-sort, rather than dividing the problem into $1/c$ pieces each of size cn for some $c < 1$, in this problem, we divide the problem into n^c pieces, each of size n^{1-c}.

Beyond that, this ladder-and-jugs problem is similar to binary-search and order-statistic, in that we can throw away all but one piece, and recurse. Because the piece we are left with is n^c, which is much smaller than cn, we converge much faster than $\Theta(n)$.