Computer Science 180, Homework 1

Michael Wu
UID: 404751542

January 18th, 2018

Chapter 2, Problem 2

Total number of operations: $3600 * 10^{10}$ ops

a)

$$\begin{aligned} n^2 &= 3600 * 10^{10} \\ n &= 6000000 \end{aligned}$$

b)

$$\begin{aligned} n^3 &= 3600 * 10^{10} \\ n &\approx 33019 \end{aligned}$$

c)

$$\begin{aligned} 100n^2 &= 3600 * 10^{10} \\ n^2 &= 3600 * 10^8 \\ n &= 600000 \end{aligned}$$

d)

$$\begin{aligned} n \log n &= 3600 * 10^{10} \\ n &\approx 1290951819848 \end{aligned}$$

e)

$$2^n = 3600 * 10^{10}$$
$$n \approx 45$$

f)

$$2^{2^n} = 3600 * 10^{10}$$
$$n \approx 5$$

Chapter 2, Problem 3

1. $f_2(n) = \sqrt{2n}$
2. $f_3(n) = n + 10$
3. $f_6(n) = n^2 \log n$
4. $f_1(n) = n^{2.5}$
5. $f_4(n) = 10^n$
6. $f_5(n) = 100^n$

Chapter 2, Problem 7

Assume for simplification that each line is the maximum length c words. Then for x lines the total number of words n is

$$n = \sum_{i=1}^{x} ci$$
$$n = c\frac{x(x+1)}{2}$$
$$n = \frac{c}{2}(x^2 + x)$$

Solving this for x allows us to calculate the number of lines given a song with a total number of words n

$$\begin{aligned}
x^2 + x &= \frac{2n}{c} \\
x^2 + x + \frac{1}{4} &= \frac{2n}{c} + \frac{1}{4} \\
\left(x + \frac{1}{2}\right)^2 &= \frac{2n}{c} + \frac{1}{4} \\
x + \frac{1}{2} &= \sqrt{\frac{2n}{c} + \frac{1}{4}} \\
x &= \frac{1}{2}\left(\sqrt{\frac{8n}{c} + 1} - 1\right)
\end{aligned}$$

To encode the song of length n words, simply write down each unique line. This requires cx words. Thus our encoding as a function of n has a length

$$f(n) = \frac{c}{2}\left(\sqrt{\frac{8n}{c} + 1} - 1\right)$$

when $\frac{n}{c}$ is an integer. If some lines are less than c words, then we should be able to encode the song in a fewer amount of words than this because more words will be repeated. To make our $f(n)$ account for this, and for the fact that $\frac{n}{c}$ must be an integer, we change it to

$$f(n) = \frac{c}{2}\left(\sqrt{8\left\lceil\frac{n}{c}\right\rceil + 1} - 1\right)$$

This gives an upper bound for the number of words needed to encode a song of length n words. Thus by encoding the song by writing each unique line, our encoding grows in $O\left(\sqrt{n}\right)$ time.

Chapter 2, Problem 8

a) For n rungs our strategy consists of dropping the first jar beginning at the lowest rung. Then continuously try each rung $\lceil(\sqrt{n})\rceil$ above the previous until the first jar breaks. Then starting from the highest rung from which the first jar did not break, try moving up one rung at a time until you reach the highest safe rung, after which you will finish your testing. The first jar will break after at most $\lceil(\sqrt{n})\rceil$ tries, and the second jar will break after at most $\lceil(\sqrt{n})\rceil$ tries as well. Thus our solution requires at most $f(n) = 2\lceil(\sqrt{n})\rceil$ tries which is an $O(\sqrt{n})$ algorithm. This is better than a linear $O(n)$ solution.

b) For $k > 2$ jars available to break, drop the 1st jar starting from the bottom of the ladder while moving up in intervals of $\left\lceil n^{\frac{k-1}{k}} \right\rceil$ rungs. When it breaks, move to the next jar. Continue by dropping the ith jar from the previous jar's highest rung reached before breaking, then moving up in intervals of $\left\lceil n^{\frac{k-i}{k}} \right\rceil$ rungs. Each jar can be dropped a maximum of $\left\lceil n^{\frac{1}{k}} \right\rceil$ times and there are k jars, so this algorithm in the worse case requires $f_k(n) = k\left\lceil n^{\frac{1}{k}} \right\rceil$ steps. Thus this algorithm is $O\left(n^{\frac{1}{k}}\right)$. This algorithm's runtime satisfies the property that $\lim_{n\to\infty} f_k(n)/f_{k-1}(n) = 0$ for each k.