PSO #1 Solutions Sketch (Week 2)

Week of 2021-08-30

1 Induction

1. (Weak Induction) Prove that $n! > 2^n$ for all $n \geq 4$

   Solution:

   Base case: Our base case is the smallest n for which we claim the statement true, $n = 4$. At $n = 4$, we have $24 > 16$, so our base case holds

   Inductive hypothesis: Suppose that $k! > 2^k$ for $k \geq 4$. Then,

$$\begin{aligned}
(k+1)! &= (k+1)k! && \text{unrolling factorial once} \\
&\geq (k+1)2^k && \text{applying IH} \\
&\geq 2^k \cdot 2 && k \geq 4 \rightarrow k+1 \geq 2 \\
&= 2^{k+1} && \text{simplifying}
\end{aligned}$$

   By the principle of mathematical induction, it follows that $n! > 2^n$ for all $n \geq 4$

2. (Weak Induction) Prove

$$\sum_{k=1}^{n} k = \frac{n(n+1)}{2}$$

   Solution:

   Base case: $n = 1$, $\sum_{k=1}^{1} k = 1$, $\frac{1(1+1)}{2} = 1$

   Inductive hypothesis: Suppose that $\sum_{k=1}^{n} k = \frac{n(n+1)}{2}$. Then,

$$\begin{aligned}
\sum_{k=1}^{n+1} k &= (n+1) + \frac{n(n+1)}{2} && \text{unrolling sum once} \\
&= (n+1) + \frac{n(n+1)}{2} && \text{applying IH} \\
&= \frac{(n+1)(n+2)}{2} && \text{simplifying}
\end{aligned}$$

3. (Strong Induction) Prove that every positive integer n can be written as a sum of distinct nonnegative integer powers of 2.

   Solution:

   Base case: $n = 1$. Then, $n = 2^0 = 1$, so the statement hold sfor $n = 1$

   Inductive hypothesis: Suppose for some $n \geq 1$ that for all $k \leq n$, we can write k as a sum of distinct powers of 2.

   Consider $n+1$. Let x be the largest integer such that $2^x \leq n+1$. Let $m = n+1-2^x$. Since $2^x \geq 1$, we have $m < n+1$. By the strong inductive hypothesis, there exist distinct integers $r_1, r_2, \ldots, r_s$ such that $m = 2^{r_1} + 2^{r_2} + \ldots + 2^{r_s}$. It follows that $n+1 = 2^x + 2^{r_1} + 2^{r_2} + \ldots + 2^{r_s}$.

   From here, we need to verify that $r_j \neq x$ for arbitrary $1 \leq j \leq s$. Suppose to the contrary that $r_j = x$ for some x. Then,

$$\begin{aligned} n+1 &= 2^x + 2^{r_j} + 2^{r_1} + \ldots + 2^{r_{j-1}} + 2^{r_{j+1}} + \ldots + 2^{r_s} \\ &= 2^x + 2^x + \ldots \\ &= 2^{x+1} + \ldots \end{aligned}$$

   But now x is not the largest integer such that $2^x \leq n+1$, a contradiction.

   Hence, $r_j \neq x$ for all j, and thus $n+1$ can be written as a sum of distinct nonnegative integer powers of 2.

   By the principle of strong mathematical induction, it follows that the statement holds for all $n \geq 1$

2 Asymptotic Runtimes

Give the big-O, Θ, Ω for the following pairs of functions:

1. $\sqrt{n} + (\log n)^5$ and $(\sqrt{2})^{\log n}$

   Solution:
   $(\sqrt{2})^{\log n}$ is $\Theta(\sqrt{n} + (\log n)^5)$

2. $8^{\log n}$ and $2n^3 + n^2(\log n)^4$

   Solution:
   $8^{\log n}$ is $\Theta(2n^3 + n^2(\log n)^4)$

3. Use L'hopital's rule to show $(\log n)^{1000}$ is $O(n^{.0001})$

   Solution:

$$\lim_{n\to\infty} \frac{(\log n)^{1000}}{n^{.0001}} = \lim_{n\to\infty} \frac{1000 * (\log n)^{999}}{.0001 * n * n^{-.9999}} = \cdots = \lim_{n\to\infty} \frac{1000!}{.0001^{1000} n^{.0001}} = 0$$

   Applying L'hopital's rule 1000 times in the intervening steps.

3 Evaluating Loops

1. Determine the number of times asymptotically the function F is called in the following code segments:

Algorithm 1 Code Segment 1

for $i = 2$ to n where $i = i^2$ **do**
    $F(i)$
end for

Runtime:
$O(\log\log(n))$

Write some values of i: $2^1, 2^2, 2^2 * 2^2 = 2^4, 2^4 * 2^4 = 2^8, \ldots$

We notice that after k iterations of the loop, the value of i is 2^{2^k}. Then, we want to solve

$$\begin{aligned} 2^{2^k} &= n \\ 2^k &= \log n \\ k &= \log\log n \end{aligned}$$

Algorithm 2 Code Segment 2

for $i = 1$ to n **do**
    for $j = i$ to n by $\sqrt{n}$ **do**
        F(i,j)
    end for
    F(i,0)
end for

Runtime:
$O(n^{3/2})$

We write a sum to demonstrate this: $\sum_{i=1}^{n} \frac{n-i}{\sqrt{n}} = \frac{1}{\sqrt{n}} * (n^2 - \frac{n(n+1)}{2}) = \frac{1}{\sqrt{n}} * (n^2 - n) = O(n^{3/2})$

End of Solutions Document