Intro. to Algorithm Assignment 1 20181257 최찬혁

1. Show that if $2^{n+1} = \Theta(2^n)$ or not by using the definition of O and Ω notation (2 points).

$f(n) = O(g(n)) \rightarrow \{ f(n) : \exists c > 0, \exists n_0 > 0 \text{ s.t. } 0 \le f(n) \le c \cdot g(n) \text{ for } \forall n \ge n_0 \}$

$f(n) = \Omega(g(n)) \rightarrow \{ f(n) : \exists c > 0, \exists n_0 > 0 \text{ s.t. } 0 \le c \cdot g(n) \le f(n) \text{ for } \forall n \ge n_0$

$f(n) = \Theta(g(n)) \rightarrow \{ f(n) : \exists c_1 > 0, \exists c_2 > 0, \exists n_0 > 0 \text{ s.t. } 0 \le c_1 \cdot g(n) \le f(n) \le c_2 \cdot g(n) \text{ for } \forall n \ge n_0 \}$

First, we need to show that $2^{n+1} = O(2^n)$.

It is obvious that $\exists c \ge 0$ and $\exists n_0 \ge 0$ s.t. $0 \le 2^{n+1} \le c \cdot 2^n$ for $\forall n \ge n_0$. We can choose $c = 2$, $n_0 = 0$

So, it is satisfied to O notation.

Second, we need to show that $2^{n+1} = \Omega(2^n)$

It is also obvious that $\exists c \ge 0$ and $\exists n_0 \ge 0$ s.t. $0 \le c \cdot 2^n \le 2^{n+1}$ for $\forall n \ge n_0$. We can choose $c = 1$, $n_0 = 0$.

So, it is satisfied to Ω notation.

Therefore, $2^{n+1} = \Theta(2^n)$ because both O and Ω apply

"Q.E.D"

2. Rank the following functions by order of growth; that is, find an arrangement $g_1, g_2, \ldots, g_{26}$ of the functions satisfying $g_1 = \Omega(g_2), g_2 = \Omega(g_3), \ldots, g_{25} = \Omega(g_{26})$. Partition your list into equivalence classes such that functions $f(n)$ and $g(n)$ are in the same class if and only if (iff) $f(n) = \Theta(g(n))$ (2 points).

$n \lg n$	$2^{2^{n+1}}$	$(\sqrt{2})^{\lg n}$	n^2	$n!$	$(\lg n)!$	$(\lg n)^{\lg n}$	e^n	$4^{\lg n}$
$(\frac{3}{2})^n$	n^3	$\lg^2 n$	$\lg(n!)$	2^{2^n}	$n^{1/\lg n}$	$(n+1)!$	$\sqrt{\lg n}$	$2^{\sqrt{2\lg n}}$
$\ln \ln n$	2^n	$n \cdot 2^n$	$n^{\lg \lg n}$	$\ln n$	1	$2^{\lg n}$	n	

Dominance relation in lecture slide : $n! \gg 2^n \gg n^3 \gg n^2 \gg n \cdot \log n \gg n \gg \log n \gg 1$.

Let me check equivalence classes across functions first :

① $n^{\lg \lg n} = (\lg n)^{\lg n}$

② $n^2 = (2^{\lg n})^2 = 4^{\lg n}$

③ $n = 2^{\lg n}$

④ $n^{1/\lg n} = 2^{\lg(n^{1/\lg n})} = 2^{(1/\lg n)\lg n} = 2 = 1$

(In Algorithm Analysis, constant can be expressed by 1)

⑤ $\lg(n!) \approx n \cdot \lg n - \cancel{n}$ we can omit lower degree term

(In Text book 78 page, Stirling Approximation $n! \le n^n$) → new dominance relation !

$n! = \sqrt{2\pi n}\left(\frac{n}{e}\right)^n\left(1 + \theta\left(\frac{1}{n}\right)\right)$

Now, we can rank the above function by order of growth :

$2^{2^{n+1}} > 2^{2^n} > (n+1)! > n! > n \cdot 2^n > e^n > 2^n$

$> (\frac{3}{2})^n > \{(\lg n)^{\lg n}, n^{\lg\lg n}\} > (\lg n)! > n^3$

$> \{n^2, 4^{\lg n}\} > \{n \lg n, \lg(n!)\} > \{n, 2^{\lg n}\}$

$> (\sqrt{2})^{\lg n} > 2^{\sqrt{2\lg n}} > \lg^2 n > \ln n > \sqrt{\lg n}$

$> \ln(\ln n) > \{1, n^{1/\lg n}\}$

p.s. $\frac{d}{dn}(\ln n) = \frac{1}{n} > \frac{d}{dn}(\lg n) = \frac{1}{n \cdot \ln 2}$

$\therefore \ln n > \lg n.$ → new dominance relation

3. Let's define a sequence S_1, S_2, S_3, ... by the rule that $S_1 = 1$, $S_2 = 1$, $S_3 = 2$ and every further term is the sum of the proceeding two. Thus, the sequence begins 1, 1, 2, 3, 5, 8, 13, ... If $k = (1+\sqrt{5})/2$, prove if the following is true or not for all positive integers n by using mathematical induction (2 points).

$$S_n \le k^{n-1} \leftarrow \text{Induction Hypothesis. (I.H.)}$$

S_n is a Fibonacci number, defined as :

$$S_i = \begin{cases} 0, & n = 0 \\ 1, & n = 1 \\ S_{n-1} + S_{n-2}, & n \ge 2 \end{cases}$$

Base Case : if $n = 1$, then $S_1 = 1 \le k^0 = 1 \rightarrow$ True.

Induction step : if $S_n \le k^{n-1}$ is true,

$S_{n+1} \le k^n$ should be true.

$$S_{n+1} = S_n + S_{n-1} \le k^{n-1} + k^{n-2} \leftarrow \text{I.H.}$$

$$\begin{aligned} k^{n-1} + k^{n-2} &= \left(\frac{1+\sqrt{5}}{2}\right)^{n-1} + \left(\frac{1+\sqrt{5}}{2}\right)^{n-2} \\ &= \left(\frac{1+\sqrt{5}}{2}\right)^{n-2}\left(\frac{1+\sqrt{5}}{2} + 1\right) \\ &= \left(\frac{1+\sqrt{5}}{2}\right)^{n-2}\left(\frac{3+\sqrt{5}}{2}\right) \\ &= \left(\frac{1+\sqrt{5}}{2}\right)^{n}\left(\frac{1+\sqrt{5}}{2}\right)^{-2}\left(\frac{3+\sqrt{5}}{2}\right) \\ &= \left(\frac{1+\sqrt{5}}{2}\right)^{n}\cancel{\left(\frac{\overset{3}{\cancel{6}}+\cancel{2}\sqrt{5}}{\cancel{4}\,2}\right)^{-1}}\cancel{\left(\frac{3+\sqrt{5}}{2}\right)} \\ &= k^n. \end{aligned}$$

$$\therefore S^{n+1} \le k^n \qquad \text{"Q.E.D"}$$

4. Prove that the number of different triples that can be chosen from n items is precisely $n(n-1)(n-2)/6$ by using mathematical induction (2 points).

(# of different triples chosen from n items)

$$= \frac{n(n-1)(n-2)}{2} = T_n \leftarrow \text{Induction Hypothesis.}$$

Base case : if $n=3$, $T_3 = 3 \cdot (3-1)(3-2)/6 = 1$,

which means only one possible that can be chosen

$\rightarrow$ true.

Inductive step : if T_n is true, we need to show that

T_{n+1} is true.

$$\Rightarrow T_{n+1} = \frac{(n+1) \cdot n \cdot (n-1)}{6}$$

We can think that # of different triples chosen from n+1 items is also given as sum of # of different tripple chosen from n items and # of way to chose two elements from a set of n elements.

$$\Rightarrow \frac{n(n-1)(n-2)}{6} + {}_nC_2$$

$$= \frac{n(n-1)(n-2)}{6} + \frac{n(n-1)}{2}$$

$$= \frac{n(n-1)(n-2+3)}{6}$$

$$= \frac{(n+1) \cdot n \cdot (n-1)}{6}$$

This is same as T_{n+1}. "Q.E.D."

5. Think of a process moving from the integer x to y via multiple steps based on the following rules.

   A. The length of each step is nonnegative, and the length of the first and last step is one.

   B. The length of the next step is either one less than, equal to, or one greater than the length of the previous step.

   For example, moving from the integer 10 to 15 takes four steps (i.e., 1+2+1+1). Write a pseudocode algorithm that finds the smallest number of steps when moving from the integer x to y (2 points).

We can find some pattern through some examples :

3: 1 1 1
4: 1 2 1
5: 1 2 1 1
6: 1 2 2 1
7: 1 2 2 1 1
8: 1 2 2 2 1

9: 1 2 3 2 1
10: 1 2 3 2 1 1
11: 1 2 3 2 2 1
12: 1 2 3 3 2 1
13: 1 2 3 3 2 1 1
14: 1 2 3 3 2 2 1

15: 1 2 3 3 3 2 1
16: 1 2 3 4 3 2 1
17: 1 2 3 4 3 2 1 1
. . .

There's the following rules :

1. It forms a perfect pyramid at square numbers.
2. A top number of the pyramid is a square root of remaining length.
3. It can step the top number until getting next top number (the root of the next square number)

We can write down the pseudocode that satisfies the above rules as follows : Next page.

```
def   min_steps (x, y) :
    n = y - x                                  // remaining length.
    array = Stack (n)                          // new stack array.
    top = floor ( sqrt (n) )                   // maximum step length.
    while  n > 0 :
        if   n - (top * top) ≥ top :
             q = floor ((n - (top * top)) / top )
             for (i = 0 ; i ≤ q ; i++)
                 array . push_back ( top )
                 n = n - top
             top = top - 1
        else :
             array. push_back (top)
             n = n - top .
             top = top - 1 .
    cnt = 0
    for (i = 1 ; i ≤ n ; i++)
        if  array [i-1] > 0 :
            cnt = cnt + 1
    return  cnt                                // minimum steps of
                                               // a given distance.
```