Assignment 1

Rahul Motan[1]

[1]Affiliation not available

January 18, 2018

1) $\lim_{n\to\infty}\ \frac{\sqrt{n}}{n\sqrt{\log n}} = \lim_{n\to\infty}\ \frac{(n)^{0.5}}{n(\log\ n)^{0.5}} = \lim_{n\to\infty}\ \frac{(n)^{-0.5}}{(\log\ n)^{0.5}}$ (Applying L-Hopital rule)

$= \lim_{n\to\infty} \frac{n^{-0.5}}{(\log\ n)^{-0.5}} = \lim_{n\to\infty} \frac{\sqrt{\log\ n}}{\sqrt{n}} = \lim_{n\to\infty} \frac{1}{\sqrt{n\ \log\ n}} = 0$ (Simplifying and applying L-Hopital Rule)

$\Rightarrow\ n\sqrt{\log n} > \sqrt{n}$

Applying the same principle to another set of functions, $2^{\sqrt{\log n}}$ &amp; $(\log n)^2$ we get,

$\equiv \lim_{n\to\infty}\ \frac{2^{\sqrt{\log n}}}{(\log\ n)^2}$

Taking Log of numerator and denominator we get,

$\equiv \lim_{n\to\infty}\ \frac{\sqrt{\log n}}{2\log\log n}$

Now Applying L- Hopital Rule, we get

$\equiv \lim_{n\to\infty} \frac{1}{4}\log n$

Applying limits we get

$\Rightarrow \infty$

Hence $2^{\sqrt{\log n}} >\ (\log n)^2$

Now we compare $\sqrt{n}$ &amp; $(\log n)^2$

$\lim_{n\to\infty}\ \frac{\sqrt{n}}{(\log n)^2}$ (Apply L-Hopital Rule we get)

$\Rightarrow \lim_{n\to\infty}\ \frac{\sqrt{n}}{\log n}$

$\equiv \frac{1}{8}\ \lim_{n\to\infty}\ \frac{n}{\sqrt{n}}$, Simplifying & Applying the limits, we get

$\Rightarrow\ \infty$

Hence, $(\log n)^2 < \sqrt{n} < n\sqrt{\log n} < 2^{\sqrt{\log n}}$

2a)

From the given relation we can infer,

$g\,(n)\ \le\ c_1 h\,(n)\ \forall\ n\ \ge n_0$

$\Rightarrow h\,(n) \ge \frac{1}{c_1} g\,(n)$

Also, $f\,(n) \ge c_2 h\,(n)\ \forall\ n \ge n_0$

therefore from the above relations we can say that,

$f(n) \geq \frac{c_2}{c_1} g(n) \Rightarrow f(n) \geq c_3 g(n) \ \forall \ n \geq n_0$

hence, $f(n) \ = \ \Omega(g(n))$ is TRUE

2b)

$f(n) \ = \ O(g(n))$

$3^{f(n)} = O\left(3^{g(n)}\right)$ if this hold true then,

$\Rightarrow \ \ 3^{f(n)} \leq \ 3^{C_1 g(n)}$

Taking Log on both sides we get,

$\Rightarrow \ f(n) . \log 3 \leq C_1 g(n) . \log 3$,

Dividing by $\log 3$ we get,

$\Rightarrow f(n) \leq C_1 g(n) \ \Rightarrow \ f(n) = \ O(g(n))$

So we can say this relationship is TRUE

3. a)

The problem here is of searching but not the exact match rather finding the number of elements lesser than the given B.M.I, additionally the exact match could be present, so we will have to check for that as well.

In the problem statement its mentioned that at the end of every year the list is prepared for the next year. Assuming the list is sorted according to the B.M.I, we can therefor apply the algorithm of Binary-Search as my friend Polly suggested.

The catch here is that we never return "-1" when the given B.M.I we are searching for is not found, instead we return the lower bound of the deduced sub-array we are searching in at that point. This could very well be the case when the given B.M.I is not present in the sub-array or list.

3. b)

Assumptions:

1. The given sub-array & the parent array are in the non-increasing order of the B.M.I and the B.M.I in the list are of the same age as of the patient.
2. Let the list of patients who visited in the last year be β, where $\beta.length \ > \ 0$
3. Let the given sub-array be α, where $\alpha.length \ > 0 \ : \ \alpha[i] \in \beta \ for \ i \ = \ 0, 1, 2, 3....n \ where \ n < \beta.length$
4. Let the given B.M.I be κ

$input \ : \ A \ sequence \ of \ n \ numbers \ \langle a_1, \ a_2, a_3... \ a_n \rangle$

$output \ : \ A \ number \ n_1 : \ 0 \leq n_1 \leq 1$

```
compute-Percentile($\kappa$, $\alpha$, $\beta.length$){
    $l \leftarrow 0$
    $r \leftarrow \alpha.length - 1$
    $\lambda \leftarrow$ search-BMI($\alpha$, $l$, $r$, $\kappa$)
    return $\frac{\lambda}{\beta.length}$
```

```
search-BMI($\alpha$, $l$, $r$, $\kappa$){
 if  $l \leq r$
      $m \leftarrow \frac{(l\ +r)}{2}$
      if $\alpha[m]\ ==\ \kappa$
            return $m$
      else if $\alpha[m] < \kappa$
            return search-BMI($\alpha$, $m+1$, $r$, $\kappa$)
        else
            return search-BMI($\alpha$, $l$, $m-1$, $\kappa$)
return $l$
```

3. c) We prove the correctness of our algorithm by induction on the size of sub-array $n\ =\ r\ -\ l\ +1$

Base case: $n\ =\ 1\ \Rightarrow l\ =\ r,\ m\ =\ \frac{(l+r)}{2}\ =\ l = r$

if $\alpha[m]\ ==\ \kappa$, then it will return $m\ = l\ = r$

if $\alpha[m]\ \neq\ \kappa$,

- $\alpha[m] < \kappa$, it will call ***search-BMI(**α, $m+1$, r, κ**)***

  this call will return $l\ \Rightarrow 1$

- $\alpha[m] > \kappa$, it will call ***search-BMI(**α, l, $m-1$, κ**)***

  this call will return l $\Rightarrow$ 0

Assumption: ***search-BMI*** works correctly for any sub-array of size $K\ =\ r-l+1$

Inductive Step:

Let α be of size $K+1$

$m\ =\ \frac{(l+r)}{2}$

there are 3 cases:

1. $\alpha[m]\ ==\ \kappa$, it returns m
2. $\alpha[m] < \kappa$, it will call ***search-BMI(**α, $m+1$, r, κ**)***

   size of sub-array now is $r-(m+1)+1 = r-m$

   a) if $l+r$ *is even*

   $r - \frac{(l+r)}{2} = \frac{(2r-l-r)}{2} = \frac{(r-l)}{2} < K$

   by our assumption, this returns the correct answer as our algorithm works for size K

   b) if $l+r$ *is odd*, $m\ =\ \frac{(l+r-1)}{2}$

   size of sub-array is $r - \frac{(l+r-1)}{2}\ =\ \frac{(2r-l-r+1)}{2} = \frac{(r-l+1)}{2} < K$

   by our assumption, this returns the correct answer as our algorithm works for size K

3. $\alpha[m] > \kappa$, it will call ***search-BMI(**α, l, $m-1$, κ **)***

the size of the sub-array now is $m - 1 - l + 1 \ = m - l$

a) if $l + r$ *is even*

$\frac{(l+r)}{2} - l = \frac{(r-l)}{2} < K$

by our assumption this returns the correct answer.

b) if $l + r$ *is odd*, $m \ = \frac{(l+r-1)}{2}$

$\Rightarrow \frac{(l+r-1)}{2} - l \ = \frac{(l+r-1-2l)}{2} = \frac{(r-l-1)}{2} = \frac{(K+1-1)}{2} = \frac{K}{2} < K$

by our assumption this returns the correct answer .

|