CSCI803 Assignment

Yao Xiao
SID 2019180015

October 29, 2020

1 Solution 1

$$\begin{aligned} log_b(x^n) &= log_b(x \cdots x) \\ &= log_b x + \cdots + log_b x \\ &= n log_b x \end{aligned} \tag{1}$$

2 Solution 2

2.1 solution 2-a

when $k > 0$,

$$\begin{aligned} k > 2 + k^{-1} &\Rightarrow k^2 > 2k + 1 \\ &\Rightarrow k^2 + k^2 > k^2 + 2k + 1 \\ &\Rightarrow 2k^2 > (k+1)^2 \end{aligned} \tag{2}$$

From the above, we can conclude that if and only if k > 0, the proof is valid.

2.2 solution 2-b

First, for k = 7, $\frac{2^k}{2k} = 9.14 > k$, the proof is valid.

Second, suppose $k = n - 1,\ 2^{(n-1)} > 2(n-1)^2$ is valid.

Third, so for k = n,

$$\begin{aligned} &2^{n-1} > 2(n^2 - 2n + 1) \\ &2^n > 4n^2 - 8n + 4 \\ &\text{Because } 4n^2 - 8n + 4 > 2n^2 \\ &\Rightarrow 2^n > 2n^2 \end{aligned} \tag{3}$$

So $2n^2 - 8n + 4 > 0 \Rightarrow n > 2 + \sqrt{2}$ *or* $n < 2 - \sqrt{2}$, and $n > 6, 2n^2 - 8n + 4 > 0$

From the first, second and third, we can conclude that $\frac{2^n}{2n} > n$

2.3 solution 2-c

$$\lim_{m \to \infty} \frac{2^n}{2n} = \lim_{m \to \infty} \frac{2^n ln2}{2} = \infty_+ \tag{4}$$

2.4 solution 2-d

Assume $log_2 m = q, q \to \infty, m = 2^q, \sqrt{m} = 2^{\frac{q}{2}}$

$$\begin{aligned} \lim_{m\to\infty} \frac{log_2 m}{\sqrt{m}} &= \lim_{q\to\infty} \frac{q}{2^{\frac{q}{2}}} \\ \text{assume } \frac{q}{2} &= t,\ q = 2t \\ \lim_{q\to\infty} \frac{q}{2^{\frac{q}{2}}} &= \lim_{t\to\infty} \frac{2t}{2^t} = 0 \end{aligned} \tag{5}$$

3 Solution 3

3.1 solution 3-a

Suppose the length of array A is m, the worst case is the length of B is $\frac{m}{2}$, and for each epoch value is the half of the previous value, so space complexity for this algorithm:

$$S(n) = \frac{n}{2} + \frac{n}{2^2} + \cdots + 1 = n - 1 \tag{6}$$

3.2 solution 3-b

$$\begin{aligned} O(n) &= 2n - 1 \\ \Omega(n) &= \frac{n}{2} \\ \Theta(n) &= n \end{aligned} \tag{7}$$

For the worst case, from the section 3.1, we can select the candidate who got the most votes which costs $T(n) = n - 1$. And we should figure out whether the candidate gets more than the half which costs n. So the so the worst case space complexity for this algorithm is $T(n) = 2n - 1$.

For the best case which will cost $\frac{n}{2}$.

3.3 solution 3-c

When the votes are consecutive and the candidate gets more than half of the votes. At least one pair of identical votes is consecutive, or no more than 50%. The algorithm is effective. When the candidate gets less than half of the votes, the score is divided into two situations:

1. The algorithm works when no one gets a relatively large number of votes, and B is empty.
2. There are some candidates who have won overwhelmingly dominant votes. In this case, regardless of whether the selection result is correct or not, the result will retain the old leader.

3.4 solution 3-d

The step set as 2 and i + 1 will be out of range will cause the problem, here is the alternative algorithm:

```
1       array_A = [5, 3, 7, 1, 7, 7, 2, 7, 7]
2       # B.length == the number of candidate
3       array_B = [0...0]
4       def vote(array_A):
5           for i in range(len(array_A) - 1):
```

```
            array_B[array_A[i] + 1] += 1
        return array_B

    def get_max(array):
        for j in range(len(array) - 1):
            if array[j+1] > array[j]:
                max = array[j+1]
                k = j
        return max, k

    def get_result(array):
        array_C = vote(array_A)
        max, k = get_max(array_C)
        if max > len(array_A) * 0.5:
            print("The new leader is:", str(k))
        esle:
            print("Retain the old leader.)
```

3.5 solution 3-e

The Big(O) is O(n), and because adding new array(C), the space complexities is O(n).

In contrast, the choice of the new algorithm is appropriate because it can satisfy all situations and has less time and space complexity than the previous algorithm.