Assignment 1

Elite Lu

October 5, 2022

1 Question 1

1.1

Algorithm 1 Determining if the equation's brackets are correct

```
procedure Brackets(L, S)          ▷ L is the inputted string, S is an empty stack
    for i from 1 to L.length do
        if L[i] = '(' then
            S.push('(')
        else if L[i] = ')' then
            if S.IsEmpty() then
                return FALSE
            else
                S.pop()
            end if
        end if
    end for
    return S.isEmpty()
end procedure
```

1.2

The time complexity for the worst case scenario would be $O(n)$. This i because the worst case scenario is where the for loop runs an n number of times, where n represents the length of the string. This string would only contain the brackets, and the brackets are arranged such that there are the same number of left and right braces. The time function that would be derived would be $T(n) = c_f + c_{13} + (c_{2:12})n$. From this time function, I determined that the time complexity for the worst case scenario would be $O(n)$.

1.3

The code will be provided as A1Q1.java.

2 Question 2

According to the definition of $\Theta(f(n))$, this is only true when there exists two arbitrary coefficients, $c_1 > 0$ and $c_2 > 0$ and an n_0 point, T(n) is sandwiched in between the function multiplied by the coefficients when n is larger than n_0. Because of this, I will prove this by showing this property and disprove by contradiction. When proving, I will look at the coefficients with the largest n term since that term would dominate and find some coefficient to fit this scenario and an n_0 value to make the statements true after that term. For the limits, if the value from the limits is between 0 and ∞ exclusive, then it is $\Theta(n)$. However, if it is 0 or ∞, then it is not.

2.1

$$\frac{64n^4 + 2n + 3}{n + 1} \in \Theta(n^3)$$

2.1.1

To prove that the the function is in $\Theta(n^3)$, I will find $c_1 > 0$ and $c_2 > 0$ and n_0 points. This is shown below:

$$c_1 \cdot f(n) \leq T(n) \leq c_2 \cdot f(n)$$

$$\begin{aligned}
c_1 \cdot f(n) &\leq T(n) \\
c_1 \cdot n^3 &\leq \frac{65n^4 + 2n + 3}{n + 1} \\
c_1 &\leq \frac{65n^4 + 2n + 3}{n^3(n + 1)} \\
c_1 &\leq \frac{65n^4 + 2n + 3}{n^4 + n^3} \\
c_1 &\leq 65 \\
c_1 &= 63
\end{aligned}$$

$$\begin{aligned}
T(n) &\leq c_2 \cdot f(n) \\
\frac{65n^4 + 2n + 3}{n + 1} &\leq c_2 \cdot n^3 \\
\frac{65n^4 + 2n + 3}{n^3(n + 1)} &\leq c_2 \\
\frac{65n^4 + 2n + 3}{n^4 + n^3} &\leq c_2 \\
65 &\leq c_2 \\
c_2 &= 70 \\
n_0 &= 1
\end{aligned}$$

Therefore, because there exists c_1, c_2, n_0, therefore the function provided is in $\Theta(n^3)$

2.1.2

$$
\begin{aligned}
\lim_{n\to\infty} \frac{\frac{65n^4+2n+3}{n+1}}{n^3} &= \lim_{n\to\infty} \frac{\frac{65n^4}{n^3} + \frac{2n}{n^3} + \frac{3}{n^3}}{n+1} \\
&= \lim_{n\to\infty} \frac{65n + \frac{2}{n^2} + \frac{3}{n^3}}{n+1} \\
&= \lim_{n\to\infty} \frac{\frac{65n}{n} + \frac{\frac{2}{n^2}}{n} + \frac{\frac{3}{n^3}}{n}}{\frac{n}{n} + \frac{1}{n}} \\
&= \lim_{n\to\infty} \frac{65 + \frac{2}{n^3} + \frac{3}{n^4}}{1 + \frac{1}{n}} \\
&= \frac{65 + \frac{2}{\infty} + \frac{3}{\infty}}{1 + \frac{1}{\infty}} \\
&= \frac{65}{1} \\
&= 65
\end{aligned}
$$

Because the limit evaluates to 64, the time complexity is in $\Theta(n^3)$

2.2

$$45n\log(n) + 2n + 1 \in \Theta(n\log(n))$$

2.2.1

To prove the function is in $\Theta(n\log(n))$, I will need to prove that the function is first in $O(n\log(n))$ and n $\Omega(n\log(n))$. I will determine two arbitrary constants and an n_0 value. I will look at the dominating value and find an appropriate constant.

$$c_1 \cdot f(n) \leq T(n) \leq c_2 \cdot f(n)$$

$$\begin{aligned}
c_1 \cdot f(n) &\leq T(n) \\
c_1 \cdot n\log(n) &\leq 45n\log(n) + 2n + 1 \\
c_1 &\leq \frac{45n\log(n) + 2n + 1}{n\log(n)} \\
c_1 &\leq 45 + \frac{2}{\log(n)} + \frac{1}{n\log(n)} \\
c_1 &\leq 45 \\
c_1 &= 40
\end{aligned}$$

$$\begin{aligned}
T(n) &\leq c_2 \cdot f(n) \\
45n\log(n) + 2n + 1 &\leq c_2 \cdot n\log(n) \\
\frac{45n\log(n) + 2n + 1}{n\log(n)} &\leq c_2 \\
45 + \frac{2}{\log(n)} + \frac{1}{n\log(n)} &\leq c_2 \\
45 &\leq c_2 \\
c_2 &= 55 \\
n_0 &= 2
\end{aligned}$$

Therefore, because there exists c_1, c_2, n_0, therefore the function provided is in $\Theta(n\log(n))$

2.2.2

$$\begin{aligned}
\lim_{n\to\infty} \frac{45n\log(n) + 2n + 1}{n\log(n)} &= \lim_{n\to\infty} \frac{\frac{45n\log(n)}{n\log(n)} + \frac{2n}{n\log(n)} + \frac{1}{n\log(n)}}{\frac{n\log(n)}{n\log(n)}} \\
&= \lim_{n\to\infty} (45 + \frac{2}{\log(n)} + \frac{1}{n\log(n)}) \\
&= 45 + \frac{2}{\infty} + \frac{1}{\infty} \\
&= 45
\end{aligned}$$

Because the limit evaluates to 45, the time complexity is in $\Theta(n\log(n))$

2.3 $n^2 \notin \Theta(\log(n))$

2.3.1

To prove by contradiction, I will examine the upper bound, which is where $T(n) \leq c_2 \cdot f(n)$. Let c_2 and n_0 be the least constants in the equation $n^2 \leq c_2 \cdot \log(n)$ Suppose when $n = e \cdot c_2$. With this, the equation simplifies to $c_2^2 \cdot e^2 \leq c_2^2$, which is a contradiction for the upper bound. Because the upper bound does not hold, then therefore $n^2 \notin O(\log(n))$. Because of this, $n^2 \notin \Theta(\log(n))$ since n^2 must be in both $O(\log(n))$ and $\Omega(\log(n))$ to be in $\Theta(\log(n)$.

2.3.2

$$\begin{aligned}\lim_{n\to\infty} \frac{n^2}{log(n)} &\stackrel{\text{H}}{=} \lim_{n\to\infty} \frac{2n}{\frac{1}{n}} \\ &= \lim_{n\to\infty} 2n^2 \\ &= \infty\end{aligned}$$

Because the limit evaluates to ∞, the time complexity is not in $\Theta(\log(n))$

2.4 $n^n \notin \Theta(2^n)$

2.4.1

To prove by contradiction, I will examine the upper bound, which is where $T(n) \leq c_2 \cdot f(n)$. Let c_2 and n_0 be the least constants in the equation $n^n \leq c_2 \cdot 2^n$ Suppose when $n = c_2$. With this, the equation simplifies to $c_2^{c_2} \leq 2^{c_2}$, which is a contradiction for the upper bound since c_2 is larger than 2. Because the upper bound does not hold, then therefore $n^n \notin O(2^n)$. Because of this, $n^n \notin \Theta(2^n)$ since n^n must be in both $O(2^n)$ and $\Omega(2^n)$ to be in $\Theta(2^n)$.

2.4.2

$$\begin{aligned}\lim_{n\to\infty} \frac{n^n}{2^n} &= \lim_{n\to\infty} (\frac{n}{2})^n \\ &= (\frac{\infty}{2})^\infty \\ &= \infty^\infty \\ &= \infty\end{aligned}$$

Because the limit evaluates to ∞, the time complexity is not in $\Theta(2^n)$

3 Question 3

3.1

Note: if asc was 1, then it would just be normal comparisons. However, when asc is -1, multiplying the numbers by -1 reverses the order since the larger magnitude would result in a smaller negative number. This is why I used -1 and 1. Also other note: The pseudocode may appear on the next page.

Algorithm 2 Selection sort algorithm

```
1: procedure SelectionSort(D, S)    ▷ D is the doubly linked list, S is a Boolean for ascending or descending; if S, then ascending and vice versa
2:     asc = 1
3:     if S then
4:         asc = -1                 ▷ Making asc -1 allows for reverse order
5:     end if
6:     temp = D.head
7:     while temp ≠ NIL do
8:         current = temp
9:         swap = temp
10:        other = temp.next
11:        while other ≠ NIL do
12:            if other.value * asc < swap.value * asc then
13:                swap = other
14:            end if
15:            other = other.next
16:        end while
17:        if current ≠ swap then
18:            swap(current, swap)
19:        end if
20:    end while
21:    return D
22: end procedure
```

3.2

The worse case scenario would result in $O(n^2)$. This is because when the worst case occurs, that would mean the whole list is not in order and would require an n number of swaps. The outer while loop would run n times and the inner while loop runs n minus the current number of loop runs from the outer while loop. The other statements would be dominated by this since worst case scenario, they run either n number of times from the other loop or run once. The time function would be $T(n) = c_f + c_{2..6} + (c_{7..10} + c_{17..19})n + (c_{11..16})n^2$. As a result, inner and outer loops result in the $O(n^2)$ since the n^2 dominates as n approaches infinity.

3.3

The code will be provided as A1Q3.java. A sample has been given. In addition to this, I realized that instead of saying .next or .prev is null, I can just check if the node is a head or a tail because the only cases where they are null is then the value is a head or a tail. I tried to cut down on lines of code by making auxiliary functions.

Note: I used equals signs for my pseudocode since the TAs were using that.