CSE 3500 hw 1

Casey Provitera

January 2024

1 Question 1 30 Points

Consider the following linear search algorithm that searches for a target element within an array of numbers.

```
1. Function search(array of size n, target):
2.     for each element in array:
3.         if element == target:
4.             return "Element found"
5.     return "Element not found"
```

Analyze the best-case, average-case, and worst-case scenarios for this linear search algorithm based on the position of the target element within the array.

(a) Best Case: What is the best-case scenario for this linear search algorithm? Explain the situation where it would perform most efficiently.
Answer –
The best case scenario runtime for is O(1), This would happen when the target is in the first index in the array.

(b) Average Case: How does the average-case scenario consider different possible positions of the target element within the array? Provide an explanation of how the algorithm performs on average.
Answer –
The average case scenario runtime is $\Theta(n/2)$. This can be seen by imagining the searched array as different distances away from a location, if these distances are 0, 1, 2, 3, 4, 5, and 6 units away the average distance away is 3 units. Now if the distances stretch to "n" units away the average distance is n/2 units away. This is because for every search that takes the algorithm towards the end of the list there is a search that only needs to check against the start of the list. For example, if you are given the array [1,4,6,2,7,5], and you are searching for 2 it will take 4 checks in the array to find 2 since it is t

the third index. But if you are looking for 4 it will only take 2 checks in the array both of which are close to the average search time in this array of 3.

(c) Worst Case: What is the worst-case scenario for this linear search algorithm? Describe the input arrangement that would lead to the highest time complexity.
Answer –
The worst-case scenario runtime is $\Omega(n+1)$. This only happens when the target that is being searched for does not exit in the array.

2 Question 2 25 Points

Consider a scenario where you need to find the sum of all even numbers between 1 and a given positive integer n.

(a) Write pseudocode to solve this problem. Make sure to clearly define the variables and provide step-by-step instructions for your algorithm.
Answer –

```
i.    Function SumEvens(positive integer n)
ii.       Integer sum = 0
iii.      for each even integer i between 0 and n
iv.           sum = sum + i
v.        return sum
```

(b) After writing the pseudocode, analyze the time complexity of each line in terms of the input size n. Provide a brief explanation for each line's time complexity analysis.
Answer –

	Time Complexity
Function SumEvens(positive integer n)	1
Integer sum = 0	1
for each even integer i between 0 and n	n
sum = sum + i	1
return sum	1

(ii) Finally, calculate and provide the overall time complexity of your algorithm using big-O notation.
Answer –
The final time complexity is the sum of the individual time complexities of each line. Final time complexity = 1+1+ n(1+1) = 2n+2.

3 Question 3 20 Points

For each group of functions, sort the functions in increasing order of asymptotic (big-O) complexity, and explain your answer.
Note: "increasing order" refers to arranging the functions from the one that grows at the slowest rate to the one that grows at the fastest rate. In other words, it means placing the functions in ascending order based on how quickly their time complexity increases as the input size grows.

Group 1 unsorted

1. $f_1(\text{n}) = \text{O}(2^n)$
2. $f_2(\text{n}) = \text{O}(\text{n!})$
3. $f_3(\text{n}) = \text{O}(n^3)$
4. $f_4(\text{n}) = \text{O}(n^2)$
5. $f_5(\text{n}) = \text{O}(\text{n log n})$

Group 1 Sorted

1. $f_5(\text{n}) = \text{O}(\text{n log n})$
2. $f_4(\text{n}) = \text{O}(n^2)$
3. $f_3(\text{n}) = \text{O}(n^3)$
4. $f_1(\text{n}) = \text{O}(2^n)$
5. $f_2(\text{n}) = \text{O}(\text{n!})$

Explanation - O(n log n) has the smallest time complexity because after $n_0 = 0$, $\text{O}(n^2)$ grows faster. $\text{O}(n^2)$ is next because after $n_0 = 1$, $\text{O}(n^3)$ grows faster. $\text{O}(n^3)$ is next because after $n_0 = 2$, $\text{O}(2^n)$ grows faster. And O(n!) is the largest time complexity because after $n_0 = 4$, O(n!) grows faster then $\text{O}(2^n)$.

Group 2 unsorted

1. $f_1(\text{n}) = 2^{2^{1000000}}$
2. $f_2(\text{n}) = 2^{100000n}$
3. $f_3(\text{n}) = \binom{n}{2}$
4. $f_4(\text{n}) = n\sqrt{n}$

Group 2 Sorted

1. $f_1(\text{n}) = 2^{2^{1000000}}$
2. $f_4(\text{n}) = n\sqrt{n}$

3. f_3(n) = $\binom{n}{2}$
4. f_2(n) = $2^{100000n}$
   Explanation – The reason f_1 is the fasted is because it is a constant running time so every equation will eventually take longer than it. f_4 is next because it can be simplified to $n^{3/2}$ this is a lower order than n^2 but higher than a constant runtime. f_3 is next, the equation for n choose 2 is $\frac{n(n+1)}{2}$ which is asymptotic with O(n^2), giving it a larger runtime than O($n^{3/2}$). And finally f_2 has the largest asymptotic runtime since after n_0 = 4 O(n!) grows faster than O(n^2).

4 Question 4 25 Points

(a) Prove that f(n) = $3n^2$+ 2n + 1 is in O(n^2). Show the constants c and n_0 that satisfy the definition of Big O.
Answer -
By definition f(n) is O(n^2) if there exists some c>0, and n_0 >= 0 such that 0< $f(n)$ <= $c * n^2$ for all n >= n_0. To prove this we will substitute f(n) into the expression. 0< $3n^2 + 2n + 1$ <= $c * n^2$, choose c to be 4, 0< $3n^2 + 2n + 1$ <= $4 * n^2$ choose n_0 to be 3, 0< $3(3^2) + 2(3) + 1$ <= $4 * 3^2$, 0 < 34 <= 36, since there is some c>0 and n_0 >= 0 which validates the expression for all n >= 3 f(n) is O(n^2).

(b) Prove that f(n) = $n^3 + 4n^2 + 2n$ is in $\Theta(n^3)$. Show the constants c_1, c_2, and n_0 that satisfy the definition of Theta notation.
Answer -
By definition for f(n) to be $\Theta(n^3)$ there must exist $c_1 > 0$, $c_2 > 0$, and n_0 >= 0 such that $0 < c_1 * n^3 <= f(n) <= c_2 * n^3$ for all n >= n_0. Substitute f(n), $0 < c_1 * n^3 <= n^3 + 4n^2 + 2n <= c_2 * n^3$ choose c_1 to be 1 and c_2 to be 5, $0 < n^3 <= n^3 + 4n^2 + 2n <= 5 * n^3$, choose n_0 to be 2, $0 < 8 <= 8 + 16 + 4 <= 5 * 8$, $0 < 8 <= 28 <= 40$, since there is a $c_1 > 0$, $c_2 > 0$, and n_0 >= 0 that validates the expression for all n >= 2 f(n) is $\Theta(n^3)$.

(c) Prove that f(n) = $2n^2 + 7n$ is in $\Omega(n^2)$. Show the constants c and n_0 that satisfy the definition of Omega notation.
Answer -
By definition f(n) is $\Omega(n^2)$ if there exists a c>0 and n_0 >= 0 such that $f(n)$ >= $c * n^2$ >= 0 for all n >= n_0. Now substitute f(n) into the expression. $2n^2 + 7n$ >= $c * n^2$ >= 0 choose c to be 1, $2n^2 + 7n$ >= n^2 >= 0 choose n_0 to be 1, $2 + 7 >= 1 >= 0, 9 >= 1 >= 0$. Since there is a c>0 and n_0 >= 0 that validate the expression for all n >= 1 f(n) is $\Omega(n^2)$.

(d) Given two functions g(n) = n^2 and h(n) = n^3, determine whether g(n) = O(h(n)). Provide a proof for your answer.

Answer -
By definition for g(n) to be O(h(n)) then there must exist some $c > 0$ and $n_0 >= 0$ such that $0 < g(n) <= c * h(n)$. Substitute g(n) and h(n) into the expression. $0 < n^2 <= c * n^3$, choose c to be 1, $0 < n^2 <= n^3$, choose n_0 to be 2, $0 < 2 <= 8$. Since there is a $c > 0$ and $n_0 >= 0$ that validate the expression for all $n >= 2$ g(n) to be O(h(n)).

(e) For the function f(n) = $5n^2$ + 3n log n + 10, determine the tightest possible bound using Big O, Omega, and Theta notations. Explain your reasoning.
Answer -
Big O(n^2) c=6
By definition f(n) is O(n^2) if there exists some c>0, and $n_0 >= 0$ where $0 < 5n^2$+ 3n log n + 10 <= $c * n^2$ for all $n >= n_0$, choose c to be 6. $0 < 5n^2$+ 3n log n + 10 <= $6n^2$, choose n_0 to be 5, $0 < 135 +$ 15 log 5 <= 150, $0 < 135 + 10.485 <= 150$, $0 < 145.485 < 150$. Since there exists some $c > 0$ and $n_0 > 0$ that validate the expression for all $n >= 5$ f(n) is O(n^2).

Big $\Theta(n^2)$ $c_1 = 5, c_2 = 6$
By definition f(n) is (n^2) if there exists some $c_1 > 0$, $c_2 > 0$, and $n_0 >= 0$ where $0 < c_1 * n^2 <= 5n^2$+ 3n log n + 10 <= $c_2 * n^2$ for all $n >= n_0$, choose c_1 to be 4 and c_2 to be 6.
$0 < 4 * n^2 <= 5n^2$+ 3n log n + 10 <= $6n^2$, choose n_0 to be 5,
$0 < 100 <= 135 + 15$ log 5 <= 150, $0 < 100 <= 145.485 <= 150$.
Since there exists some $c_1 > 0$, $c_2 > 0$ and $n_0 > 0$ that validate the expression for all $n >= 5$ f(n) is $\Theta(n^2)$.

Big $\Omega(n^2)$ c=5
By definition f(n) is $\Omega(n^2)$ if there exists some $c > 0$ and $n_0 >= 0$ where $0 < c * n^2 <= 5n^2$+ 3n log n + 10 for all $n >= n_0$, choose c to be 4.
$0 < 4 * n^2 <= 5n^2$+ 3n log n + 10, choose n_0 to be 5,
$0 < 100 <= 135 + 15$ log 5, $0 < 100 <= 145.485$. Since there exists some $c > 0$ and $n_0 > 0$ that validate the expression for all $n >= 5$ f(n) is $\Omega(n^2)$.

Remember, proofs for asymptotic notations often involve finding appropriate constants (c and n_0) that satisfy the definitions of Big O, Theta, and Omega. Provide clear explanations and reasoning in your answers.