CSPB 3104 - Park - Algorithms

Started on	Thursday, 18 January 2024, 10:42 PM
State	Finished
Completed on	Thursday, 18 January 2024, 10:49 PM
Time taken	6 mins 43 secs
Marks	8.80/8.80
Grade	**10.00** out of 10.00 (**100**%)

Question **1**

Correct

Mark 0.80 out of 0.80

The code below searches whether $\lfloor \log_2(n) \rfloor$ consecutive zeros occur in an input list of n binary numbers 0s and 1s.

```
def consecutiveZeros (lst):
      n = len(lst)
      k = floor(log2(n)) # Assume this calculate floor of lg(n)
      rLength = 0
      for i in range (0,n):
            if (lst[i] == 0):
                  rLength = rLength + 1
            else:
                  rLength = 0
            if (rLength >= k):
                   return True
      return False
```

What is the **worst case complexity** as a function of n: select the tightest bound possible ?

- $O(n^2)$
- $O(n \log(n))$
- $O(\log(n))$
- $O(n)$ Correct!

Mark 1.00 out of 1.00

The correct answer is: $O(n)$

What is the **best case complexity** as a function of \(n\)?

- $O(n^2)$
- $O(n \log(n))$
- $O(\log(n))$ Correct: this is the best case
- $O(n)$

Mark 1.00 out of 1.00

The correct answer is: $O(\log(n))$

Which of the following inputs causes the worst case running time over lists of size \(n\)?

- An input of all zeros
- $n/2$ ones followed by all zeros
- An input of all ones Correct
- An input of $n-1$ zeros with a one bit someplace in the middle

Mark 1.00 out of 1.00

The correct answer is: An input of all ones

Which of the following inputs causes the best running time behavior over lists of size \(n\)?

- (selected) An input of all zeros Correct
- $n/2$ ones followed by all zeros
- An input of all ones
- None of the choices provided

Mark 1.00 out of 1.00

The correct answer is: An input of all zeros

Correct

Marks for this submission: 0.80/0.80.

Question **2**

Correct

Mark 0.80 out of 0.80

The function isPalindrome shown below takes an input string "*inpstr*" and two indices "i", "j" :

```
def isPalindrome(inpstr,i,j):

   # Check if the substring starting from i and ending at j is a palindrome.

    assert(i <= j)
    assert( i >= 0 and j < len(inpstr)) # check that i,j are valid indices into inpstr
    while (i < j):
        if (inpstr[i] == inpstr[j]):
            i  = i +1
            j = j -1
        else:
            return False # Not a palindrome
    # Otherwise, a palindrome
    return True
```

What is the number of times the loop in isPalindrome executes as a function of i and j, in the worst case?

- (selected) $\lceil (j-i)/2 \rceil$ Correct
- $\lceil (j+i)/2 \rceil$
- $\lceil i \times j/2 \rceil$
- $\lceil n/2 \rceil$
- n

Mark 2.00 out of 2.00

The correct answer is: $\lceil (j-i)/2 \rceil$

Next, the function below, uses the isPalindrome function to find the longest Palindrome inside a given string inpstr.

It works as follows:

- Iterate through each i from 0 to end of string
- Iterate through j from i+1 to end of string
- Check if the string starting at i and ending at j is a palindrome using the isPalindrome method

```
def findLongestPalindrome(inpstr):
    longestSoFar = 1 # So far longest is trivially just a single letter
    longestStart = 0 # The start point for the longest palindrome so far
    longestEnd = 0   # The end point
    n = len(inpstr)
    for i in range(0,n): # Iterate for all starting points i
        for j in range(i+1,n): # Iterate starting from i+1 to the end
            if (isPalindrome(inpstr,i,j)): # Is it a palindrome from i to j ?
                pLength = j - i +1 # If yes,
                if (pLength > longestSoFar): # Is it the longest so far?
                    longestSoFar = pLength # Store information for the longest
                    longestStart = i       # starting point
                    longestEnd = j         # end point
    return inpstr[longestStart:longestEnd+1] # return longest found.
```

What is the tightest bound on the method's running time?

- $O(n^2)$
- $O(n^4)$
- $O(n^3)$ Correct
- $O(n)$

Mark 1.00 out of 1.00

The correct answer is: $O(n^3)$

Now, Prof. Stan Nats wishes to improve the complexity of finding the longest palindrome through a divide and conquer method. He formulates the following scheme

- If the string is of size 0 or 1, return the entire string as the largest palindrome.
- Divide the given string into two substrings a1, a2 of roughly half the size of the original string.
- Find the largest palindrome in a1
- Find the largest palindrome in a2
- Return the bigger of the two palindromes.

- The algorithm is incorrect for finding the largest palindrome Correct
- The algorithm correctly discovers the largest palindrome
- The algorithm never terminates
- The algorithm can be fixed simply by considering an extra base case for size 2 strings.

Mark 1.00 out of 1.00

The correct answer is: The algorithm is incorrect for finding the largest palindrome

Correct

Marks for this submission: 0.80/0.80.

Question **3**

Correct

Mark 0.80 out of 0.80

Select all functions below that belong to $O(n^2)$. Make sure that functions that do not belong to $O(n^2)$ are not selected.

Each correct selection will give you positive points but each wrong selection will take away some points. So do pay attention to each choice.

Select one or more:

- [ ] a. $n^{2.4} + 2n^{1.3}$
- [x] b. $100000000000000 \times n^2 + 10349891898128419819081908098$ Do not let the constant terms fool you.
- [ ] c. $e^n - \log(n)$
- [x] d. $1006n + 2 \times 10^9 \log(n^{10})$ Do not let the form of the function fool you. After some simplification, you should be able to find the leading term quite easily.
- [x] e. $26.3 \log_6(n)$ Note that all logarithms can be converted to the standard natural logarithms after dividing by a constant factor: $\log_a(n) = \frac{\log_e(n)}{\log_e(a)}$.
- [x] f. $0.000000001n^2 - 2n + 3$ The leading term is n^2. Now use this information to make up a correct answer.
- [x] g. $\log(n^{500})$ Note that $\log(a^k) = k \log(a)$.
- [x] h. $2n^{1.3} + 3n^{1.5}$ First, select the leading term. Next compare its growth to n^2 as n grows large.

Your answer is correct.

It is important for each choice to understand what the "leading term" will be as n grows larger. Use that insight to understand whether this leading term will eventually dominate n^2.

The correct answers are: $0.000000001n^2 - 2n + 3$
, $1006n + 2 \times 10^9 \log(n^{10})$
, $100000000000000 \times n^2 + 10349891898128419819081908098$
, $2n^{1.3} + 3n^{1.5}$
, $\log(n^{500})$
, $26.3 \log_6(n)$

Correct

Marks for this submission: 0.80/0.80.

Question **4**

Correct

Mark 1.20 out of 1.20

Consider the following list of functions:

$f_1 : \log(n),\ f_2 : n\log(n),\ f_3 : n^2,\ f_4 :\ n^{1.2},\ f_5 :\ n^2\log(n),\ f_6 : 1.2$

Arrange the functions in asymptotically increasing order from asymptotically smallest to largest. Use the boxes below to write down the IDs of the functions from smallest first to the largest last. Just fill in a number: for instance, write "6" to indicate f_6.

6

1

2

4

3

5

Correct

Marks for this submission: 1.20/1.20.

Question **5**

Correct

Mark 1.00 out of 1.00

Consider the list of functions below:

$f_1 : 3^n,\ f_2 : 2^{2n},\ f_3 : (2n)^n,\ f_4 : n^n,\ f_5 : 2^n$

Arrange the functions in asymptotically increasing order,

from the asymptotically smallest first to the asymptotically largest last.

5

1

2

4

3

Correct

Marks for this submission: 1.00/1.00.

Question **6**

Correct

Mark 1.60 out of 1.60

Consider the following list of functions:

$f_1 : \log(n),\ f_2 : n\log(n),\ f_3 : n^2,\ f_4 :\ n^{1.2},\ f_5 :\ n^2 \log(n),\ f_6 : 1.2$

(A) Consider the function $g_1 : n^2 + 2.5n - \sqrt{n} + 2\log(n) + 5$

What is the "leading term" of g_1 above?

- (selected) n^2 Correct
- $n\sqrt{n}$
- $n^2\log(n)$
- 5
- No leading term

Mark 1.00 out of 1.00

The correct answer is: n^2

(i) Whch of the functions $f_1, \ldots, f_6$ provides the tightest possible upper bound $O(?)$ for g_1? **Write down the index j of the function f_j or "NONE" if none of the functions provides a bound.**

3

(ii) Tightest possible lower bound $\Omega(?)$ for g_1 (above)? **Write down the index j of the function f_j or "NONE" if none of the functions provides a bound.**

3

(iii) Big Theta bound $\Theta(?)$ for g_1 (above)? **Write down the index j of the function f_j or "NONE" if none of the functions provides a bound.**

3

Here is the list of functions, again:

$f_1 : \log(n),\ f_2 : n\log(n),\ f_3 : n^2,\ f_4 :\ n^{1.2},\ f_5 :\ n^2 \log(n),\ f_6 : 1.2$

(B) Consider the function $g_2 : n\sqrt{n} + 10n\log(n) + 5$

What is the "leading term" of g_2 above?

- (selected) $n\sqrt{n}$ Correct
- $n\log(n)$
- $n^2\log(n)$
- 5
- No leading term

Mark 1.00 out of 1.00

The correct answer is: $n\sqrt{n}$

(i) Tightest upper bound $O(?)$ for g_2? **Write down the index j of the function f_j or "NONE" if none of the functions provides a bound.**

3

(ii) Tightest lower bound $\Omega(?)$ for g_2 (above)? **Write down the index j of the function f_j or "NONE" if none of the functions provides a bound.**

4

(iii) Big Theta bound $\Theta(?)$ for g_2 (above)? **Write down the index j of the function f_j or "NONE" if none of the functions provides a bound.**

NONE

Correct

Marks for this submission: 1.60/1.60.

Question **7**

Correct

Mark 1.20 out of 1.20

Consider the following list of functions:

$f_1 : \log(n),\ f_2 : n\log(n),\ f_3 : n^2,\ f_4 : n^{1.2},\ f_5 : n^2\log(n),\ f_6 : 1.2$

(A) Consider the function $g_3 : n\log(n) + \sqrt{n}$

What is the "leading term" of g_3 above?

- (•) $n\log(n)$ Correct
- ( ) $n\sqrt{n}$
- ( ) $n\sqrt{n}\log(n)$
- ( ) 5
- ( ) No leading term

Mark 1.00 out of 1.00

The correct answer is: $n\log(n)$

(i) Which of the functions $f_1, \ldots, f_6$ provides the tightest possible upper bound $O(?)$ for g_3? **Write down the index j of the function f_j or "NONE" if none of the functions provides a bound.**

2

(ii) Tightest possible lower bound $\Omega(?)$ for g_3 (above)? **Write down the index j of the function f_j or "NONE" if none of the functions provides a bound.**

2

(iii) Big Theta bound $\Theta(?)$ for g_3 (above)? **Write down the index j of the function f_j or "NONE" if none of the functions provides a bound.**

2

Here is the list of functions again:

$f_1 : \log(n),\ f_2 : n\log(n),\ f_3 : n^2,\ f_4 : n^{1.2},\ f_5 : n^2\log(n),\ f_6 : 1.2$

(B) Consider the function $g_4 : n^4$

(i) Tightest upper bound $O(?)$ for g_4? **Write down the index j of the function f_j or "NONE" if none of the functions provides a bound.**

NONE

(ii) Tightest lower bound $\Omega(?)$ for g_4 (above)? **Write down the index j of the function f_j or "NONE" if none of the functions provides a bound.**

5

(iii) Big Theta bound $\Theta(?)$ for g_4 (above)? **Write down the index j of the function f_j or "NONE" if none of the functions provides a bound.**

NONE

Correct

Marks for this submission: 1.20/1.20.

Question **8**

Correct

Mark 1.40 out of 1.40

This problem involves exponential functions. Think carefully before answering!

Consider the list of functions below:

$f_1 : 2^n,\ f_2 : 2^{2n},\ f_3 : 3^{2n},\ f_4 : 2^{2\log_2(n)},\ f_5 : n^n,\ f_6 : n!,\ f_7 : n^{1.2n}$

(A) Consider the function $g_1 : 5^n + 3 \times 4^n + 2 \times 3^n + 2^n + n$

What is the leading term for g_1 (above)?

- (•) 5^n Correct
- ( ) 4^n
- ( ) 3^n
- ( ) 2^n
- ( ) n
- ( ) No leading terms

Mark 1.00 out of 1.00

The correct answer is: 5^n

Select a function from $f_1, \ldots, f_7$ above that forms the tightest upper bound $O(?)$ to g_1. Simply enter the the index i for the function f_i in the box below. If no upper bound is found, enter "NONE".

3

Select a function from $f_1, \ldots, f_7$ above that forms the tightest lower bound $\Omega(?)$ to g_1. Simply enter the the index i for the function f_i in the box below. If no upper bound is found, enter "NONE".

2

Select a function from $f_1, \ldots, f_7$ above that is asymptotically equivalent: $\Theta(?)$ to g_1. Simply enter the the index i for the function f_i in the box below. If no upper bound is found, enter "NONE".

NONE

(B) Consider the function $g_2 : 5n^2$

Select a function from $f_1, \ldots, f_7$ above that forms the tightest upper bound $O(?)$ to g_2. Simply enter the the index i for the function f_i in the box below. If no upper bound is found, enter "NONE".

4

Select a function from $f_1, \ldots, f_7$ above that forms the tightest lower bound $\Omega(?)$ to g_2. Simply enter the the index i for the function f_i in the box below. If no upper bound is found, enter "NONE".

4

Select a function from $f_1, \ldots, f_7$ above that is asymptotically equivalent: $\Theta(?)$ to g_2. Simply enter the the index i for the function f_i in the box below. If no upper bound is found, enter "NONE".

4

Correct

Marks for this submission: 1.40/1.40.