**HOME WORK**

**3.1,3.2,3.3**

**2020380029**

**RAIHAN MD RAKIBUL ISLAM**

* 3.1

2. Determine which characteristics of an algorithm described in the text (after Algorithm 1) the following procedures have and which they lack.

a) procedure double (n: positive integer) while n > 0 n: = 2n

**Answer:** The procedure has the input, definiteness, correctness, effectiveness and generality properties. It lacks output and finiteness properties.

**Input:** The input is the positive integer n.

**Output:** There is no return statement.

**Definiteness:** All steps are precisely defined.

**Correctness:** There is no output.

**Finiteness:** The procedure has a loop with no stopping condition.

**Effectiveness:** Each individual step is executable in a finite amount of time.

**Generality:** The procedure is "equally applicable" (it never ends nor it has an output, but it "does what it was meant to do" regardless of the subset of the chosen input set).

b) procedure divide (n: positive integer) while n ≥ 0 m: = 1/n n: = n − 1

**Answer:** The procedure has the input, definiteness, correctness, finiteness and generality properties. It lacks output and effectiveness properties.

**Input:** The input is the positive integer n.

**Output:** There is no return statement.

**Definiteness:** All steps are precisely defined.

**Correctness:** There is no output. But if there was one at the end of the given procedure, this property would not be satisfied since (for any!) input n ∈ ℕ, n > 0, there comes a step after which n = 0 becomes true, after which another cycle of the while loop begins, with the first statement m: = ℝ

**Finiteness:** The procedure obviously ends in a finite number of steps. Ends with an error, but still ends.

**Effectiveness:** It is not possible to execute step m: = exactly.

**Generality:** The procedure is "equally applicable" (it always ends with an error, but on the way there, it does what it was meant to do" regardless of the subset of the chosen input set).

c) procedure sum (n: positive integer) sum: = 0 while i < 10 sum: = sum + i

**Answer:** The procedure has the input. correctness, effectiveness and generality properties. It lacks output, definiteness and finiteness properties.

**Input:** The input is the positive integer n.

**Output:** There is no return statement.

**Definiteness:** The initial value of the variable i is not set.

**Correctness:** There is no output.

**Finiteness\*:** Depending of the initial value of variable i, the procedure will. either run indefinitely or it will stop immediately. On a real computer, both scenarios are possible. **Effectiveness:** Each individual step is executable in a finite amount of time.

**Generality:** The input of this procedure has no impact on its steps, so the input can be literally anything.

d) procedure choose (a, b: integers) x: = either a or b

**Answer:** The procedure has the input. definiteness, correctness, effectiveness and generality properties. It lacks out put and definiteness properties.

**Input:** The input is two integers a and b.

**Output:** There is no return statement.

**Definiteness:** A choice is not a precise step.

**Correctness:** There is no output.

**Finiteness:** The procedure has a single step.

**Effectiveness:** Each individual step is executable in a finite amount of time. Unless someone incredibly indecisive is put to this task.

**Generality:** The procedure is applicable for any a, b ∈ ℤ, including the subset a =b. As in c), the input set could be anything.

9. A palindrome is a string that reads the same forward and backward. Describe an algorithm for determining whether a string of n characters is a palindrome.

**Answer:**

procedure palindrome(a a...a,: string with n≥ 1)

k:=True for i:=1 to ⌊n/2⌋

if then k:=False

return k

14. List all the steps used to search for 7 in the sequence given in Exercise 13 for both a linear search and a binary search.

**Answer:**

\*LINEAR SEARCH

Given sequence: 1, 3, 4, 5, 6, 8, 9, 11

We need to search for 7. We first comnpare the first element with 7. If the element is not equal to 7, then we compare the next element with 7 and so on.

Step 1: 1 is different from 7

Step 2: 3 is different from 7

Step 3: 4 is different from 7

Step 4: 5 is different from 7

Step 5: 6 is different from 7

Step 6: 8 is different from 7

Step 7: 9 is different from 7

Step 8: 11 is different from 7

We obtained that 7 is different from each value in the given list and thus 7 is not present in the list.

\*BINARY SEARCH

Given sequence: 1, 3, 4, 5, 6, 8, 9, 11

Split the sequence in half and determine which one could contain 7. We repeat this process until we obtain the subsequence 7 or subsequences of length 1 that do not contain 7.

Step 1: Divide the sequence in half: 1,3,4,5 and 6,8,9,11

Step 2: 5 is the maximum of the sequence of 1,3,4,5 and 5 is less than 7, thus 7 is not in the sequence 1,3,4,5 (so it could only be in the sequence 6,8,9,11)

Step 3: Divide the sequence 6,8,9,11 in half: 6,8 and 9,11

Step 4: 9 is the minimum of the sequence 9,11 and 9 is more than 7, thus 7 is not in the sequence 9,11.

Step 5: Divide the sequence 6,8 in half: 6 and 8

Step 6: 6 is the maximum of the sequence 6 and 6 is less than 7, thus 7 is not in the sequence.

Step 7: 8 is the minimum of the sequence 8 and 8 is more than 7, thus 7 is not in the sequence.

We then obtained that 7 is not in any subsequence of the given sequence and thus 7 is not contained in the sequence

46. How many comparisons does the insertion sort use to sort the list n, n − 1, …, 2, 1?

The binary insertion sort is a variation of the insertion sort that uses a binary search technique (see Exercise 44) rather than a linear search technique to insert the i th element in the correct place among the previously sorted elements.

**Answer:** Comparisons = 1+1+1 ……. +1 =

52. Use the greedy algorithm to make change using quarters,dimes, nickels, and pennies for

a) 87 cents. b) 49 cents.

c) 99 cents. d) 33 cents.

**Answer:**

**DEFINITIONS Greedy algorithm:** The total amount n is first compared with the largest denomination. As long as the total amount n is more than (or equal to) the largest denomination, then one such coin is added and the total amount is decreased by the denomination. Then we repeat it for the second largest denomination, third largest denomination and so on (until we checked all denominations).

Quarter: 25 cent

Dime: 10 cent

Nickel: 5 cent

Penny: 1 cent

**(a) 3 quarters, 1 dime, 0 nickels and 2 pennies.**

**(b) 1 quarter, 2 dimes, 0 nickels and 4 pennies.**

**(c) 3 quarters, 2 dimes, 0 nickels and 4 pennies.**

**(d) 1 quarter, 0 dimes, 1 nickel and 3 pennies.**

* **3.2**

2. Determine whether each of these functions is O ().

a) f (x) = 17x + 11

**SOLUTION:**

Given: O(x²), thus g(x) = in the definition of the Big-O Notation.

f(x) = 17x + 11

For convenience sake, we will choose k = 18 and thus use x > 18.

|f(x)| = |17x + 11| < |17x| + |11| = 17x + 11 ≤ 18x < x. x = = |x²|

Thus, we need to choose C to be at least 1. Let us then take C = 1.

By the definition of the Big-O notation, f(x) = O) with k = 18 and C = 1.

b) f (x) = + 1000

**SOLUTION:**

Given: O (), thus g(x) = in the definition of the Big-O Notation.

f(x) = + 1000

For convenience sake, we will choose k = 100 and thus use x > 100

Since x > 100, we also know > 10= 10000

If (x)| = |+ 1000| < || + |1000| = + 1000 < += 2 = 2||

If we then take C = 2, then by the definition of the Big-O notation, f(x) = O () with k = 100 and C = 2.

c) f (x) = x log x

**SOLUTION:**

Given: O (), thus g(x) = in the definition of the Big-O Notation.

f(x) = x log x

The property log x ≤ x holds, when x > 0. Thus, let us choose k = 0.

|f(x)| = |x log x| ≤ |x. x | = ||

If we then take C = 1, then by the definition of the Big-O notation, f(x) = O () with k = 0 and C = 1.

d) f (x) = /2

**Not O ()**

e) f (x) =

**Not O ()**

f) f (x) = ⌊x⌋ · ⌈x⌉

**SOLUTION:**

Given: O (), thus g(x) = in the definition of the Big-O Notation.

f(x) = ⌊x⌋ · ⌈x⌉

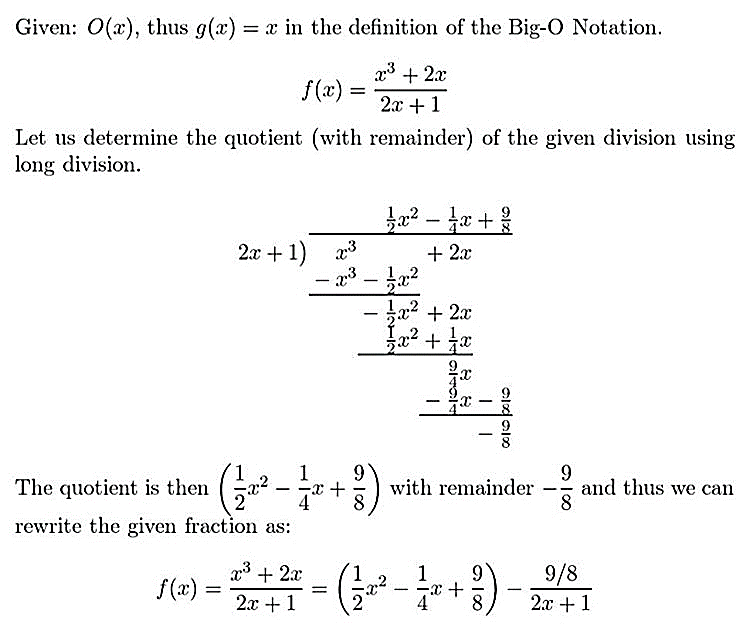
Using the definition of the floor function and ceiling function, while assuming that x >1 (thus k = 1):

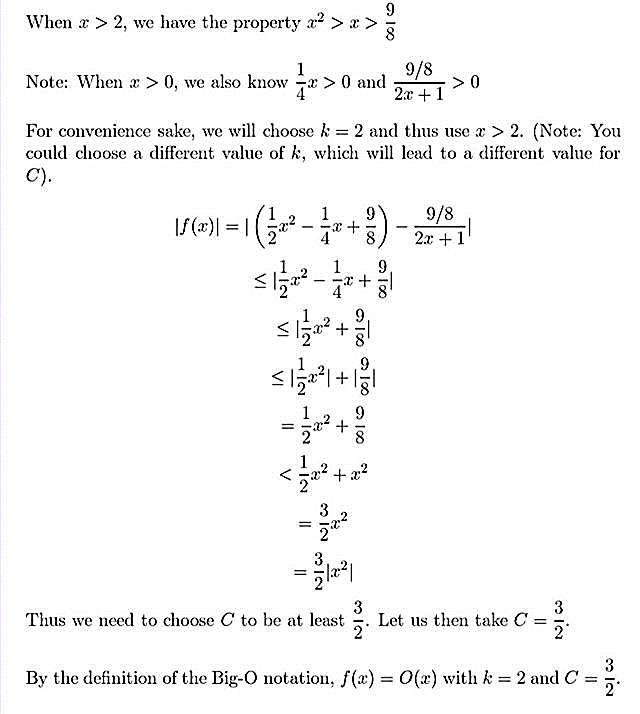
|⌊x⌋ · ⌈x⌉|= ⌊x⌋ · ⌈x⌉ ≤ x. (x+ 1) = + x ≤ + = 2 = 2||

Then we need to choose C to be at least 2. Let us then take C = 2.

By the definition of the Big-O notation, f(x) = O () with k = 1 and C = 2.

6. Show that ( + 2x)/ (2x + 1) is O ()

**Answer:** 



So, ( + 2x)/ (2x + 1) is O () [showed]

16. Show that if f (x) is O(x), then f (x) is O ().

**Answer:**

Given: f(x) is O(x)

To proof: f(x) is O ()

Proof Since f (x) is O(x), there exist constants C and k such that:

|f(x)| ≤ |, when x>k.

Let us assume k > 1, which is a safe assumption as k is a lower boundary. Since k > 1, we then also obtain that x> 1.

|f(x)| ≤ C |x| = C x = Cx.1 < C x. x= C = C|

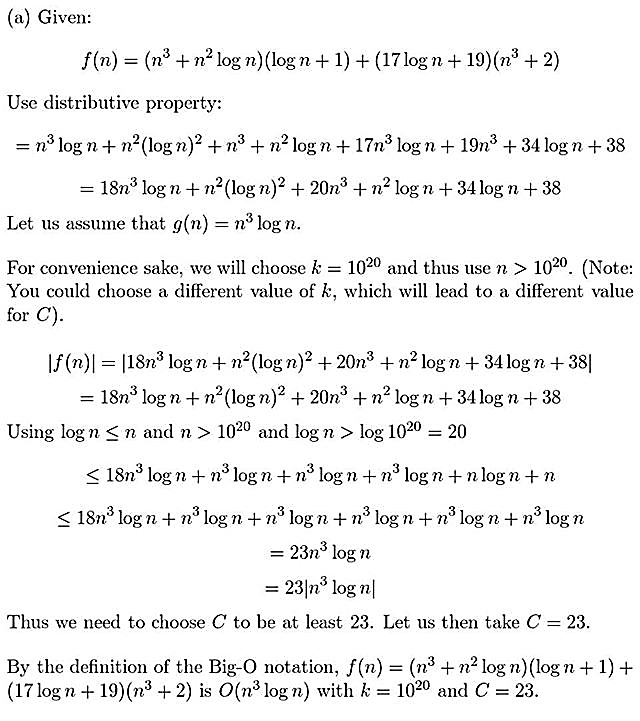
By the Big-O definition, we then obtained that f(x) is O () with constants k (k >1) and C.

If f(x) is O(x), then f(x) is O ()

26. Give a big-O estimate for each of these functions. For the function g in your estimate f (x) is O(g(x)), use a simple function g of smallest order.

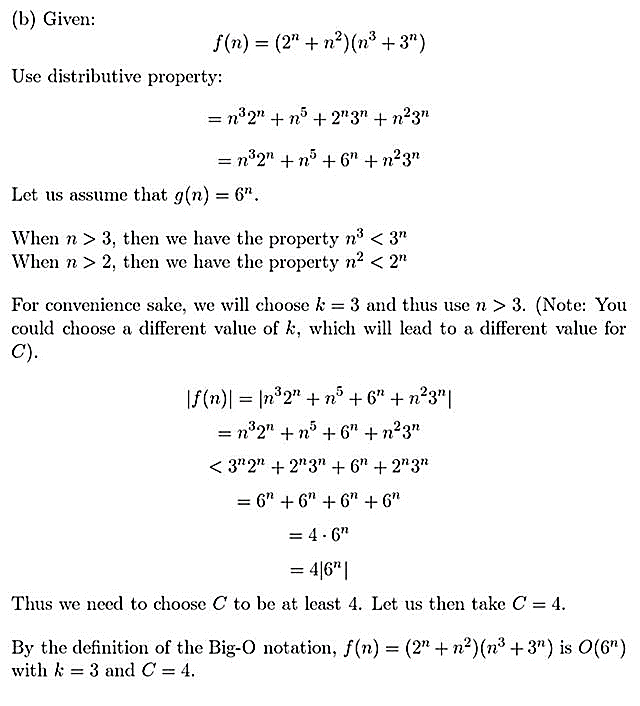
a) (+ log n) (log n+1) + (17 log n+19) (+2)

**Solution:**



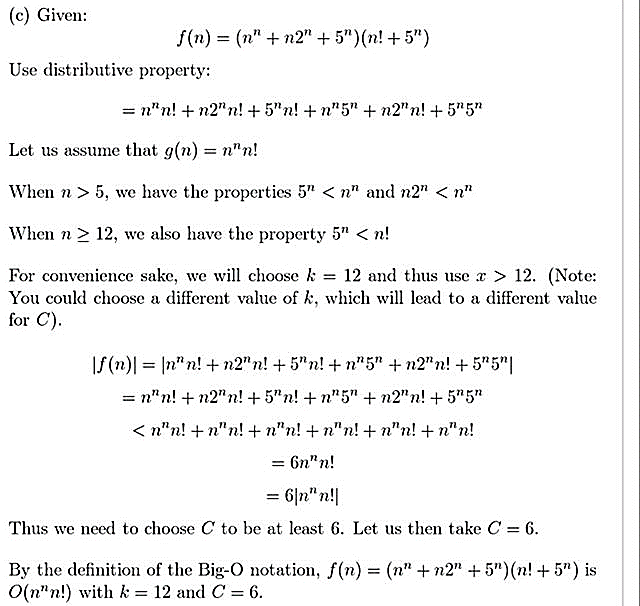
b) ( + ) ( + )

**Solution:**



c) (+ + )(n! +)

**Solution:**



34. a) Show that 3+ x + 1 is () by directly finding the constants k, , andin Exercise 33.

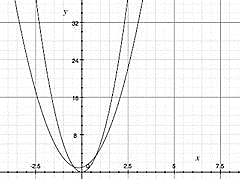
**Answer:**

One such solution is = 0, = 2, k = 1.

Since 0 ≤ |3 +x+ 1| ≤ 6 for x >1

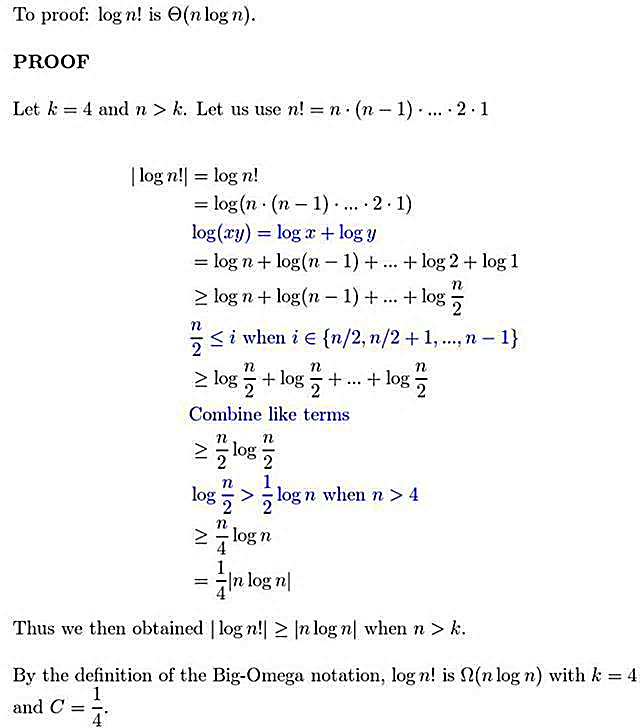
b) Express the relationship in part (a) using a picture showing the functions 3 + x + 1, · 3, and · 3, and the constant k on the x-axis, where ,, and k are the constants you found in part (a) to show that 3 + x + 1 is (3).

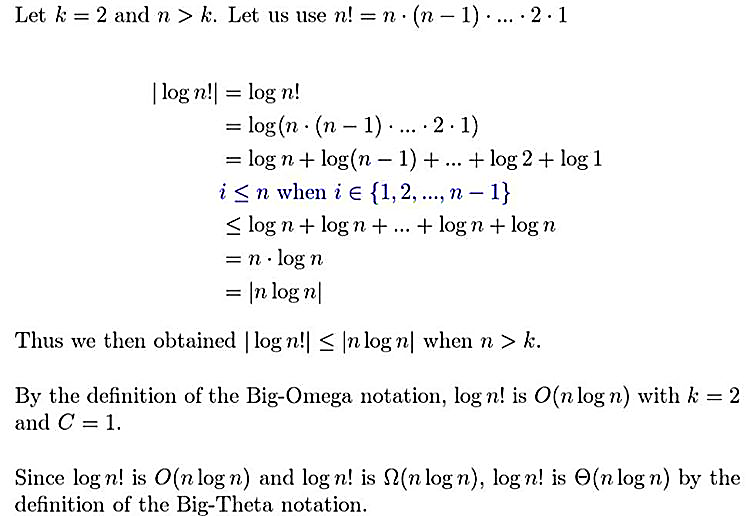
**Answer:**



72. Determine whether log n! is (n log n). Justify your answer.

**Answer:**





* 3.3

18. How much time does an algorithm take to solve a problem of size n if this algorithm uses 2 + operations, each requiring seconds, with these values of n?

a) 10

n = 10

The time that the algorithm takes is the product of the time per operation and the number of operations:

T = 1(2 + ) seconds

= 1(2(1) + ) seconds

= 1224 \*1seconds

= 1224 nanoseconds

b) 20

n = 20

The time that the algorithm takes is the product of the time per operation and the number of operations:

T = 1(2 + ) seconds

= 1(2(2) + ) seconds

= 1049376 \* 1seconds

= 0.001049376 seconds

=1,049,376 nanoseconds

c) 50

n = 50

The time that the algorithm takes is the product of the time per operation and the number of operations:

T = 1(2 + ) seconds

= 1(2(5) + ) seconds

=1\*1, 125, 899, 906, 847, 624 seconds

≈1,125,900 seconds \*

=18765 minutes \*

≈313 hours \*

≈13 days

d) 100

n = 100

The time that the algorithm takes is the product of the time per operation and the number of operations:

T = 1(2 + ) seconds

= 1(2(10) + ) seconds

=1\*1,267,650,600,228,229,401,496,703,225,376 seconds

≈ 1.27\*seconds \*

≈ 2.11 \* minutes \*

≈ 3.52 \* hours \*

≈ 1.47 \* days \*

≈ 4.02 \* Years

Thus about 40.2 trillion years

36. Show that the greedy algorithm for making change for n cents using quarters, dimes, nickels, and pennies has O(n) complexity measured in terms of comparisons needed.

**Answer:**

