UNIT 3

ALGORITHMS, INDUCTION AND RECURSION

SIA GROUP

PART-A
SHORT QUESTIONS WITH SOLUTIONS

Q1. What is an algorithm?

Answer :

An algorithm is a step-by-step process of solving a particular problem. In essence, an algorithm is a set of rules that must be followed when solving a specific problem. Algorithm typically refers to a set of instructions that can be executed by a computer to produce the desired result.

Q2. Write an algorithm for binary search.

Answer : Model Paper-1, Q1(e)

The Pseudocode for the binary search algorithm is as follows,

procedure binary search (x: integer, $p_1, p_2, \ldots p_n$: increasing integers)

$i := 1$ {i is left end point of search interval}

$j := n$ {j is right end point of search interval}

While $i < j$

$m := \lfloor (i + j)/2 \rfloor$

if $x > p_m$ **then** $i := m + 1$

else $j := m$

if $x = p_m$ **then** location $:= i$

else location $:= 0$

return location {location is the subscript i of the term p : equal to x, or 0 if x is not found}

Q3. Define sorting.

Answer :

Sorting is a process of arranging the given list of items or elements in a particular order i.e., either ascending or descending.

Some of the sorting algorithms are,

1. Bubble sort
2. Insertion sort
3. Selection sort
4. Merge sort
5. Quick sort etc.

Q4. Discuss on greedy algorithm.

Answer :

Greedy algorithm is an approach which selects a feasible solution or best choice at each step.

Consider a set of coins with denominations as quarter, dime, nickel and penny. The greedy algorithm for making change using finite set of coins is as follows.

procedure change ($c_1, c_2, \ldots, c_r$: values of denominations of coins, where $c_2 > c_2 > \ldots > c_r$; n: a positive integer)

for $i := 1$ **to** r

$d_i := 0$ {d_i counts the coins of denomination c_i}'

While $n \geq c_i$

$d_i := d_i + 1$

$n := n - c_i$

The greedy algorithm produces change using the fewest coins possible.

Q5. What is halting problem?

Answer :

Halting problem is a problem which eventually stops the computer program when run with an input. It is unsolvable with any procedure.

Q6. 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 : Model Paper-2, Q1(e)

Given,

Input is a string of n characters

Let $a_1, a_2 \ldots\ldots\ldots a_n$ be n characters

The algorithm to determine the given string as palindrome is as follows

procedure palindrome ($a_1, a_2 \ldots\ldots\ldots a_n$: string with $n \geq 1$)

$k :=$ True ; considering the string is a palindrome i.e., true

for $i := 1$ **to** $[n/2]$

If $a_i \neq a_{n-i+1}$ **then** $k :=$ False; If i^{th} symbol is not same as $(n - i + 1)^{th}$ symbol then answer is false i.e., not palindrome.

return k; return the variable k (i.e., True or false)

Q7. Write about Big-O notation.

Answer : Model Paper-1, Q1(f)

Let $f(n)$ and $g(n)$ be two non-negative functions. $f(n)$ is said to be $O(g(n))$ if and only if there exists positive constants 'k' and 'n_0' such that,

$f(n) \leq k * g(n)$ for all non-negative values of n, where, $n \geq n_0$.

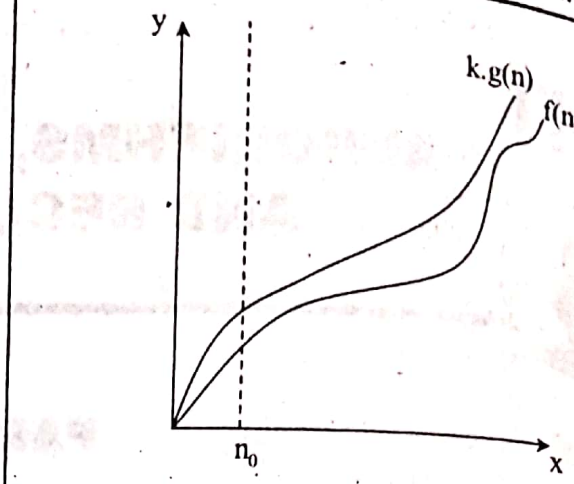


Figure: Big-O Notation

The above definition states that the function 'f' is at most k times the function 'g' when n is greater than 'n_0'.

This notation provides an upper bound for the function f i.e., the function $g(n)$ is an upper bound on the value of $f(n)$ for all n, where $n \geq n_0$.

Big-O estimates of combinations of functions can be provided.

If $f_1(n)$ is $O(g_1(n))$ and that $f_2(n)$ is $O(g_2(n))$, then,

(i) $(f_1 + f_2)(n)$ is $O(\max(|g_1(n)|, |g_2(n)|))$.

(ii) $(f_1 f_2)(n)$ is $O(g_1(n)g_2(n))$

Q8. If $f(n) = 50n^3 - 6n + 23$, then show that $f(n) = O(n^3)$

Answer :

Given that,

$$f(n) = 50n^3 - 6n + 23$$

By triangle inequality,

$$|f(n)| = |50n^3 - 6n + 23| \leq |50n^3| + |-6n| + |23|$$

For $n = 1$,

$$|f(1)| = |50(1)^3 + 6(1) + 23 < 150n^3 + 6n^3 + 23n^3| \text{ (When } n \geq 1)$$
$$= 79\, n^3$$

$\therefore$ By considering $c = 79$ it can be said that,

$$f(n) = O(n^3)$$

Q9. Show that $\lfloor xy \rfloor$ is O(xy).

Answer : Model Paper-3, Q1(e)

Given function is,

$$f(x, y) = \lfloor xy \rfloor$$

Let $k_1 = k_2 = 1$

For $x > k_1$ and $y > k_2$

$$|f(x, y)| = \lfloor \lfloor xy \rfloor \rfloor$$
$$= \lfloor xy \rfloor$$
$$= \leq xy \qquad [\because \lfloor a \rfloor \leq a]$$
$$= |xy|$$
$$= 1|xy|$$

Here $C = 1$.

$\therefore$ $f(x, y) = \lfloor xy \rfloor$ is $O(xy)$ with $C = 1$, $k_1 = 1$ and $k_2 = 1$.

Q10. Define space complexity and time complexity.

Answer : Model Paper-2, Q1(f)

Space Complexity

The space complexity of a computer program is defined as the amount of memory space required to solve a problem as a function of the size of the input.

Time Complexity

Time complexity of an algorithm is defined as, "the execution time of an algorithm". The time complexity of a program is the amount of time it takes to solve a problem as a function of size of the input.

An algorithm with best time complexity can perform its work faster. Besides time complexity, 'space complexity' of an algorithm also matters for its performance. This is essentially the amount of memory that an algorithm requires. A good algorithm keeps this number as small as possible too.

Q11. Write an algorithm for matrix multiplication.

Answer : Model Paper-4, Q1(e)

Consider, $A = [a_{ij}]_{m \times k}$ and $B = [b_{ij}]_{k \times n}$ be two matrices such that,

$$c = [c_{ij}]_{m \times n} = A \times B$$

$$= [a_{ij}]_{m \times k} \times [b_{ij}]_{k \times n}$$

The pseudocode for above matrix product is given as follows,

procedure matrix multiplication (A, B : matrices)

for $i := 1$ **to** m

for $j := 1$ **to** n

$c_{ij} := 0$

for $q := 1$ **to** k

$c_{ij} := c_{ij} + a_{iq} b_{qj}$

return C

Two $(n \times n)$ matrices, when multiplied, requires n^3 multiplications and $n^2(n-1)$ additions.

Q12. Define algorithmic paradigm.

Answer :

Algorithmic paradigm is an approach based on a concept which can be used to construct algorithms for various problems.

Brute-Force algorithm is an algorithmic paradigm which is used to solve a problem in straight forward manner. This is done based on problem statements and definitions while neglecting problem structure or computing resources.

Q13. Give the principle of mathematical induction.

Answer : Model Paper-5, Q1(e)

The principle of mathematical induction states that a proportional function $q(n)$ is true for all positive integers n based on two steps.

Basis Step

Verifying $q(n)$ is true.

Induction Step

To prove that the conditional statement, $q(k) \rightarrow q(k+1)$ is true for all positive integer k.

Q14. Use Mathematical induction to prove that $7^{n+2} + 8^{2n+1}$ is divisible by 57 for every nonnegative integer n.

Answer :

Let $q(n)$ be the proposition, "$7^{n+2} + 8^{2n+1}$ is divisible by 57".

Basis Step

For $n = 0$,

$q(0) \Rightarrow 7^{0+2} + 8^{2 \cdot 0+1} = 7^2 + 8^1 = 57$ is divisible by 57.

Thus, $q(0)$ is true.

Induction Step

For $n = k \Rightarrow 7^{k+2} + 8^{2k+1}$ is divisible by 57.

For $n = k+1$, $q(k+1)$,

$7^{(k+1)+2} + 8^{2(k+1)+1}$ is divisible by 57.

i.e., $7^{(k+1)+2} + 8^{2(k+1)+1} = 7^{k+3} + 8^{2k+3}$

$= 7 \cdot 7^{k+2} + 8^2 \cdot 8^{2k+1}$

$= 7 \cdot 7^{k+2} + 64 \cdot 8^{2k+1}$

$= 7(7^{k+2} + 8^{2k+1}) + 57 \cdot 8^{2k+1}$

$= 7(q(k) + 57 \cdot 8^{2k+1}$

is divisible by 7

$\therefore$ By the principle of mathematical induction, $7^{n+2} + 8^{2n+1}$ is divisible by 57 for every non negative integer n.

Q15. Find the flaw with the following "proof" that $a^n = 1$ for all nonnegative integers n, whenever a is a non zero real number.

Basic Step: $a^n = 1$ is true by the definition of a^n.

Induction Step: Assume that $a^j = 1$ for all nonnegative integers j with $j \leq k$. Then note that

$$a^{k+1} = \frac{a^k \cdot a^k}{a^{k-1}} = \frac{1 \cdot 1}{1} = 1$$

Answer :

Given,

Basis Step

$a^0 = 1$ is true by the definition of a^0.

Inductive Step

Assume that $a^j = 1$ for all non-negative integers j with $j \leq k$. Then,

$$a^{k+1} = \frac{a^k \cdot a^k}{a^{k-1}} = \frac{1 \cdot 1}{1} = 1$$

In the inductive step, $a^k = 1$ and $a^{k-1} \, 1 = 1$. But, in the basis step, only a^0 is considered instead of a^0 and a^1.

This is the flaw with the proof.

Q16. Define structural induction and recursive algorithm.

Answer :

Model Paper-3, Q1(f)

Structural Induction

Structural induction defines all members of a set constructed recursively with a particular property. The proof of structural induction consists of following steps.

Basis Step

The result holds for all elements specified in recursive definition to be in the set.

Recursive Step

The result holds for new elements which are constructed in the recursive step of the definition.

Recursive Algorithm

Recursive algorithm is an algorithm which solves a problem by reducing it to an instance of the same problem with smaller input.

Q17. Give a recursive definition of the set of bit strings that are palindromes.

Answer :

Let S be the set of all bit strings that are palindrome.

Basis Step

The empty string is a palindrome.

$\lambda \in S$

All strings containing only one bit is a palindrome.

$0 \in S$

$1 \in S$

Recursive Step

If the first and last letters of a string are same, it is a palindrome. Also the remaining string is also a palindrome.

$0\ s\ 0 \in S$ whenever $s \in S$

$1\ s\ 1 \in S$ whenever $s \in S$

Q18. Give a recursive algorithm for computing n!, where n is a non-negative integer.

Answer :

The algorithm to compute $n!$ is as follows.

procedure factorial(n : nonnegative integer)

if $n = 0$ **then return** 1

else return n. factorial $(n - 1)$.

Q19. Give an algorithm for a recursive merge sort.

Answer :

Model Paper-4, Q1(f)

The algorithm to compute recursive merge sort is as follows.

procedure mergesort $(L = a_1, \ldots\ldots a_n)$

If $n > 1$ **then**

$m := \left\lfloor \frac{n}{2} \right\rfloor$

$L_1 := a_1, a_2, \ldots\ldots a_m$

$L_2 := a_{m+1}, a_{m+2} \ldots\ldots a_n$

$L :=$ merge(mergesort (L_1), mergesort (L_2))

Q20. Devise a recursive algorithm for finding x^n and m whenever n, x, and m are positive integers based on the fact that $x^n \bmod m = (x^{n-1} \bmod m \cdot x \bmod m) \bmod m$.

Answer :

The recursive algorithm for finding $x^n \bmod m$ is as follows.

procedure modulo (n, x, m : positive integers)

If $n = 1$ **then**

return $r \bmod m$

else

return (modulo $(n-1, x, m)$. $x \bmod m) \bmod m$; $(x^n \bmod m = (x^{n-1} \bmod m \, x \bmod m) \bmod m)$

Q21. Devise an algorithm for evaluating a^n when n is a nonnegative integer. [Hint : Use the binary expansion of n].

Answer :

An algorithm to evaluate a^n when n is a non negative integer is,

procedure power(n: nonzero real number, n: nonnegative integer)

If $n = 0$ **then**

return 1

If n is even **then**

return power $(a, n/2)$.power$(a, n/2)$

else

return a.power$(a, n-1)$

Q22. Prove that the program segment

y := 1

z := x + y

is correct with respect to the initial assertion x = 0 and the final assertion z = 1.

Model Paper-5, Q1(f)

Answer :

Given program segment,

$y := 1$

$z := x + y$

Initial assertion, $x = 0$

Final assertion, $z = 1$

In the program segment, initially, program assigns value 1 to the variable y.

Then the program adds x and y and assigns it to z.

i.e., $z = x + y = 0 + 1 = 1$

$\therefore$ Final assertion, $z = 1$.

Thus the program segment is correct.

PART-B

LONG QUESTIONS WITH SOLUTIONS

3.1 ALGORITHMS

3.1.1 Algorithms

Q23. Define an algorithm. Describe the characteristics of the algorithm.

Answer :

Algorithm

For answer refer Unit-III, Q1.

Characteristics of Algorithm

1. **Input:** The algorithm receives zero or more inputs.
2. **Output:** The algorithm produces at least one output.
3. **Definiteness:** The steps are clearly and unambiguously stated.
4. **Finiteness:** There are finite number of instructions. Hence, the algorithm should terminate after a finite number of instructions have been executed.
5. **Correctness:** The algorithm should produce correct and accurate results.

The study of algorithms deals with,

1. Designing of algorithms
2. Validation of algorithms
3. Analysis of algorithms
4. Testing of a program.

Q24. Explain about searching algorithms.

Answer :

An algorithm or a program that searches several algorithms to locate the element in the list is called searching algorithm. It can be described as follows,

1. Initially, considering an element 'x' that may or may not be present in the list of distinct elements $p_1, p_2, p_3 \ldots p_n$.
2. In the search problem, if x is present or located in the list then, $x = a_i$ (where, i is the solution), otherwise it is 0.

There are two main search algorithms namely,

1. Linear search
2. Binary search

1. Linear Search

In this search algorithm, the required element to be located is compared with each element of the list sequentially, until a match is found. Thus, it is also called as sequential search.

Let $p_1, p_2, p_3 \ldots p_n$ be distinct elements and x be the element to be located in them. According to this algorithm, x compares with p_1. It gives the solution as p_1 if, $x = p_1$ otherwise (i.e., $x \neq p_1$), x compares with next element p_2. In this, the solution is p_2 if $x = p_2$ otherwise (i.e., $x \neq p_2$), x compares with p_3 and so on. This comparison continues until a match is found. If the list does not contain x, then the solution is 0.

The pseudocode for linear search algorithm is given by,

procedure linear search (x: integer, $p_1, p_2, \ldots p_n$: distinct integers)

$i := 1$

While ($i \leq n$ and $x \neq p_i$)

    $i := i + 1$

if $i \leq n$ **then** location := i

else location := 0

return location {location is the subscript of the term that equals x, or is 0 if x is not found}

Binary Search

Binary search technique is implemented on sorted list of elements (i.e., in increasing order). It is faster than linear search. Hence this method is efficient when the size of the list is larger.

The principle followed in binary search is "divide and conquer". The search is initiated by dividing the list into two sublists and computing the middle element. This element is then compared with the element to be searched. If the middle element is same as desired element (say key element), the search is terminated successfully, and the location (or index) of the middle element is returned. On the other hand, if the middle element doesn't match, then it is checked whether the middle element is greater than or less than the key element. If middle element is greater, then search is performed on right half of the sublist or else, the desired element is searched into the left half of the sublist. This process is repeated, until the desired element is found. If the desired element is not found, then the search terminates unsuccessfully.

For remaining answer refer Unit-III, Q2.

Q25. Discuss on binary search with example.

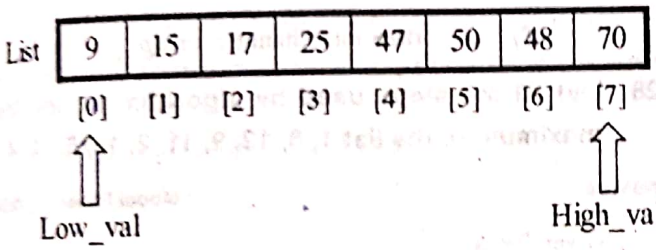
Answer : Model Paper-1, Q6(a)

Binary Search

For answer refer Unit-III, Q24.

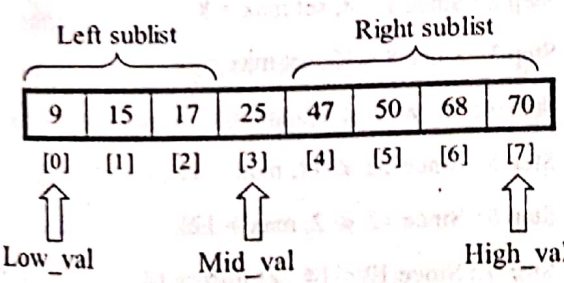
Example

Consider the following ordered list having 8 elements,

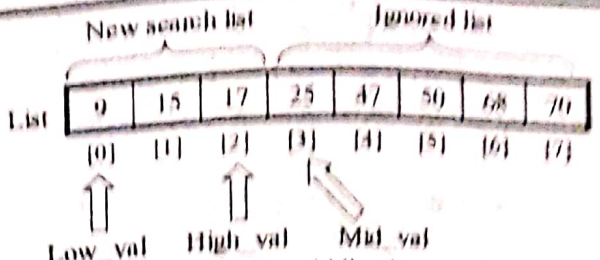


Assuming that the element to be searched is '15'. Initially, the middle (or center) element is determined by using the formula,

$$Mid\_val = \frac{Low\_val + High\_val}{2} = \frac{0+7}{2} = 3$$



Now, compare key element 15, with the value at index 'Mid_val' i.e., 25. Since, 15 < 25, search is restricted to the new list i.e., list [0] to list [2] i.e., left half of sublist. This is done, by moving the High_val pointer to one location before the mid _val pointer.

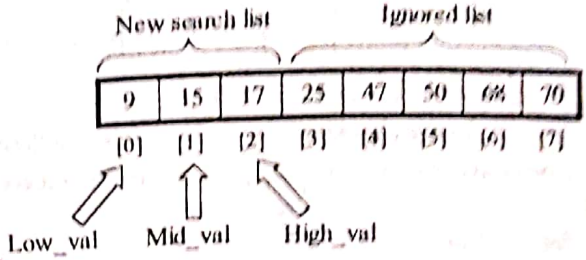


In the next step, calculate the middle element (i.e., mid_val) of the new list as,

$$Mid\_val = \frac{Low\_val + High\_val}{2}$$

$$= \frac{0+2}{2} = 1$$

Therefore, the mid_val found is at index 1, i.e., List[1] = 15.



Now, compare the 'key element' with the value at location mid_val. Since, List[mid_val] is equal to 'KeyEle', then element is found. Hence, its location (i.e., pointed by mid_val) is returned.

Q26. Discuss on Insertion sort.

Answer :

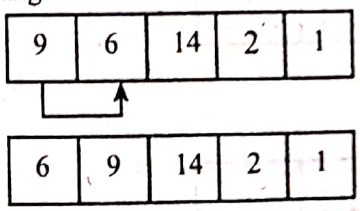
Insertion sort is a sorting technique, which inserts an element at an appropriate location by comparing each element with the corresponding elements present at its left.

Example

Considering the following list having five elements. 9, 6, 14, 2, 1.

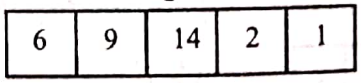
First Pass

Element 9 is compared with element 6. As 9 > 6, hence they are interchanged.



Second Pass

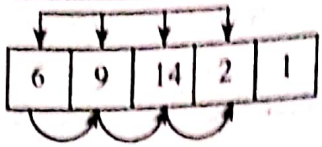
Element 14 is compared with elements 6, 9. As 14 > 6 and 9, hence no change in the series.

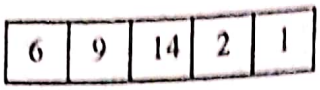


No change

Third Pass

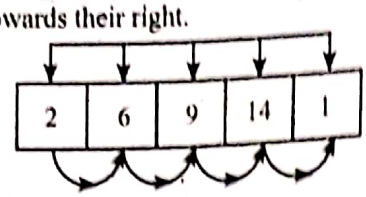
Element 2 is compared with elements 6, 9, 14. As 2 < 6, 9 and 14 hence 2 is placed in the beginning and rest of the elements are moved one position towards their right.

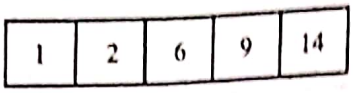




Fourth Pass

Element 1 is compared with elements 2, 6, 9, 14. Hence 1 is placed at the beginning and the rest of the elements are moved one position towards their right.





Thus, the elements are in ascending order.

The pseudocode algorithm of insertion sort is as follows,

procedure insertion sort ($p_1, p_2, \ldots p_n$: real numbers with $n \geq 2$)

for $j := 2$ **to** n

$i := 1$

while $p_j > p_i$

$i := i + 1$

$m := p_j$

for $k := 0$ **to** $j - i - 1$

$p_{j-k} := p_{j-k-1}$

$p_i := m$

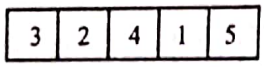
Q27. Explain bubble sort.

Answer :

Bubble sort is a sorting algorithm, which bubble's up the largest element in the correct position after each pass. It uniformly interchanges the adjacent elements in increasing order by adjusting their positions.

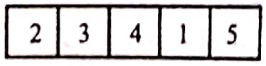
Example

Consider 5 elements namely 3, 2, 4, 1, 5 which are to be bubble sorted in increasing order.

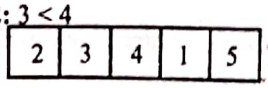


First Pass

Step 1: 3 > 2

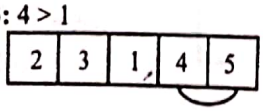


Step 2: 3 < 4

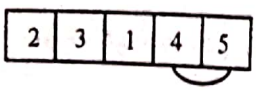


No exchange

Step 3: 4 > 1

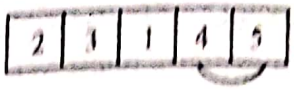


Step 4: 4 < 5



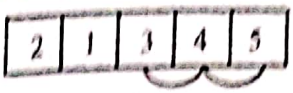
No exchange

Step 1: 2 < 3

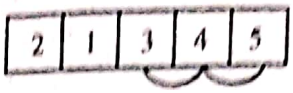


No exchange

Step 2: 3 > 1



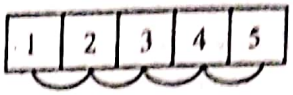
Step 3: 3 < 4



No exchange

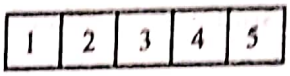
Third Pass

Step 1: 2 > 1



Fourth Pass

Step 1: 1 < 2



No exchange

Thus, bubble sort is completed.

The pseudocode for bubble sort algorithm is shown below,

procedure bubble sort ($p_1, \ldots p_n$: real numbers with $n \geq 2$)

for $i := 1$ **to** $n - 1$

for $j := 1$ **to** $n - i$

if $p_j > p_{j+1}$ **then** interchange p_j and p_{j+1}

Q28. List all the steps used by algorithm to find the maximum of the list 1, 8, 12, 9, 11, 2, 14, 5, 10, 4.

Answer : Model Paper-I, Q6(a)

Given list is,

1, 8, 12, 9, 11, 2, 14, 5, 10, 4. The steps used by algorithm to find the maximum of the given list is as follows.

Step 1 : set max = 1.

Step 2 : Since 1 < 8, set max = 8.

Step 3 : Since 8 < 12, set max = 12.

Step 4 : Since 12 $\not<$ 9, max = 12.

Step 5 : Since 12 $\not<$ 11, max = 12.

Step 6 : Since 12 $\not<$ 2, max = 12.

Step 7 : Since 12 < 14, set max = 14.

Step 8 : Since 14 $\not<$ 5, max = 14.

Step 9 : Since 14 $\not<$ 10, max = 14.

Step 10 : Since 14 $\not<$ 4, max = 14.

$\therefore$ Maximum of given list = 14

Q29. Describe an algorithm that takes as input a list of n integers in nondecreasing order and produces the list of all values that occur more than once. (Recall that a list of integers is nondecreasing if each integer in the list is at least as large as the previous integer in the list).

Answer :

Given,

Input of an algorithm is a list of n integers (non-decreasing order) output is the list of all values that occur more than once.

Let the list of n integers be $a_1, a_2 \ldots\ldots a_n$. The algorithm for these values is as follows.

procedure repeated ($a_1, a_2 \ldots\ldots a_n$: integers with $n \geq 1$ and $a_1 \leq a_2 \leq \ldots\ldots \leq a_n$)

$S := \phi$; Set S is empty set that represents repeated numbers list

$a_{n+1} := a_n - 1$; $a_n < a_{n+1}$

for $i := 2$ **to** n

if $a_i := a_{i-1}$ and $a_i < a_{i+1}$ **then** $S := S \cup \{a_i\}$; If the value is repeated, then add the value a_i to the set S.

return S ; Return the set of all repeated values

Q30. Describe an algorithm to find the longest word in an English sentence (Where a sentence is a sequence of symbols, either a letter or a blank, which can then be broken into alternating words and blanks).

Answer :

Let the input be a list of symbols $a_1, a_2 \ldots\ldots a_n$. Then, the algorithm to find longest word in an english sentence is as follows.

procedure longestword ($a_1, a_2 \ldots\ldots a_n$: symbols with $n \geq 1$)

$i := 1$; First word to be the longest word.

if $a_i \neq$ " " **then** $i := i + 1$

else longest: = $a_1, a_2 \ldots\ldots a_{i-1}$; First word contains all symbols until the first blank

max := $i - 1$; Length of first word

$m := i$; Position of the blank before the next word

for $k := i + 1$ **to** n

if $a_i \neq$ " " **then** $k := k + 1$

else if $[(k-1)-(m+1)+1] >$ max **then**; Length of each next word is compared to the length of first word

longest := $a_{m+1}\, a_{m+2} \ldots a_{k-1}$

max := $(k-1)-(m+1)+1$; length of the longest work

$m := k$

return longest

Q31. The ternary search algorithm locates an element in a list of increasing integers by successively splitting the list into three sublists of equal (or as close to equal as possible) size, and restricting the search to the appropriate piece. Specify the steps of this algorithm.

Answer : Model Paper-2, Q6(a)

Let, the input be a list of increasing integers $a_1, a_2 \ldots\ldots a_n$ and an integer x. The ternary search algorithm is shown as follows

procedure ternary ($a_1, a_2 \ldots\ldots a_n$; integers with $n \geq 1$, x: integer)

i :=1; First element position

$j := n$; Last element position

While $i < j - 1$; For atmost 2 elements, algorithm stops

$l := \lfloor (i+j)/3 \rfloor$; Interval is split from i to 1 (first interval), $l + 1$ to u(second interval)

$u := \lfloor 2(i+j)/3 \rfloor$; $u + 1$ to j(Third interval).

if $x \geq a_u \geq$ **then** $i := u + 1$; Determining the position of x in subinterval

else

if $x \geq a_u$ **then** ; assigning search interval loundaries

$i := l + 1$

$j := u$

else $j := 1$

If $x = a_i$ **then** location := i; x present in search interval

else if $x = a_j$ **then** location := j

else location: = 0; x does not present in search interval.

return location; Return of location in the list

Q32. In a list of elements the same element may appear several times. A mode of such a list is an element that occurs at least as often as each of the other elements; a list has more than one mode when more than one element appears the maximum number of times.

Devise an algorithm that finds all modes. (Recall that a list of integers is nondecreasing if each term of the list is at least as large as the preceding term.)

Answer :

Let the input be a list of non-decreasing integers $a_1, a_2 \ldots\ldots a_n$. The algorithm to find all modes is as follows

procedure modes ($a_1, a_2 \ldots\ldots a_n$: integers with $n \geq 1$ and $a_1 \leq a_2 \leq \ldots\ldots \leq a_n$)

$j := 1$; Frequency of most frequent value

for $i := 1$ **to** n ; Number of times count repeats

count(a_i) := 1 ; Count is 1 for all elements

for $k := 2$ **to** n

if $a_k = a_{k-1}$ **then** count (a_k) := count (a_{k-1}) + 1; if present and previous values are equal then count increments previous elem

if count (a_k) > j **then**; if count is larger than maximum count

$j :=$ count (a_k) ; maximum count is adjusted

$S := \phi$; S is empty set

for $m := 1$ **to** n

If count (a_m) = j **then** $S := S \cup \{a_m\}$; For same count as maximum count, element is added to the list S

return S ; returns S containing all modes.

Q33. (a) Devise a variation of the insertion sort that uses a linear search technique that inserts th element in the correct place by first comparing it with the (j – 1)st element, then the (j – 2)th elem is necessary, and soon.

(b) Use your algorithm to sort 3, 2, 4, 5, 1, 6

(c) Using this algorithm answer how many comparisons does the insertion sort use to sort the li

(i) 1, 2,.....n

(ii) n,n – 1, ...2, 1

Answer :

(a) Let $a_1, a_2 \ldots\ldots a_n$ be input real numbers to insertion sort.

The algorithm that inserts j^{th} element by comparing it with $(j-1)^{st}$ element then $(j-2)^{nd}$ element, and so on is as follo

procedure insertion sort ($a_1, a_2 \ldots\ldots a_n$: real numbers with $n \geq 2$)

for $j := 1$ **to** $n - 1$

begin

$i := n$

while $a_{n-j} < a_i$

$i := i + 1$

$m := a_{n-j}$

for $k := i$ to n

$a_{n+i-k} := a_{n+i-k-1}$

$a_i := m$

end

(b) Given set is,

3, 2, 4, 5, 1, 6

First Pass

The last two elements are compared. Since 1 < 6, 1 should be before the 6. The data set is,

3, 2, 4, 5, 1, 6

Second Pass

5 is compared with sorted 1 and 6

Since 5 > 1 and 5 < 6, it is inserted between 1 and 6 in the data set as,

3, 2, 4, 1, 5, 6

Third Pass

4 is compared with sorted 1, 5 and 6

Since 4 < 6, 4 < 5 and 4 > 1, it is inserted between 1 and 5 as,

3, 2, 1, 4, 5, 6

Fourth Pass

2 is compared with sorted, 4, 5 and 6

Since $2 < 6$, $2 < 5$, $2 < 4$ and $2 > 1$, it is inserted between 1 and 4

3, 1, 2, 4, 5, 6

Fifth Pass

3 is compared with sorted 1, 2, 4, 5, 6

Since $3 < 6$, $3 < 5$, $3 < 4$ and $3 > 2$, it is inserted between 2 and 4 as,

1, 2, 3, 4, 5, 6

Thus, all numbers are in correct order. Therefore, insertion sort algorithm stops.

(i) Given list is 1, 2, 3, ... n

In insertion sort, initially, last two elements are compared for first pass. Thus it makes 1 comparison. Then, in second pass, the third last element is compared with the last two elements. Thus, it makes 1 comparison.

Similarly for third pass, it makes 1 comparison and so on.

Therefore, for k^{th} pass, $(k+1)^{th}$ last element compares with the last k terms and thus makes 1 comparison.

Then $(n-1)$ terms are compared with n^{th} term, the insertion algorithm stops.

Number of comparisons $= 1 + 1 + 1 + \ldots + 1$

$$= \sum_{i=1}^{n-1} 1 = n - 1$$

(ii) Given list is,

$n, n-1, \ldots, 2, 1$

In first pass, the last two elements are compared. Thus making 1 comparison.

In Second pass, The third last element is compared with the last two elements, thus making 2 comparisons.

For third pass, the forth last element is compared with the last three elements, making 3 comparisons.

Thus, for k^{th} pass, $(k+1)^{th}$ last element is compared with the last k terms, making k comparisons.

Then, the n^{th} element is compared with the last $n-1$ terms, the insertion algorithm stops.

Number of comparisons $= 1 + 2 + 3 + \ldots + (n-1) = \sum_{i=1}^{n-1} i = \frac{(n-1)n}{2} = \frac{n^2 - n}{2}$

Q34. Use the greedy algorithm to make change using quarters, dimes, nickels, and pennies for

(a) 51 cents.

(b) 69 cents

Answer : Model Paper-4, Q5(a)

(a) Given,

Number of cents, $n = 51$

To change 51 cents, selecting particularly

For quarter, $c_1 = 25$

For dime, $c_2 = 10$

For Nickel, $c_3 = 5$

For penny, $c_4 = 1$

First Pass (Quarters)

$c_1 = 25$

Since $n \geq 25$, 1 quarter is added which decreases the total amount by 25,

$d_1 = 1$

$n_1 = n_{old} - 25 = 21 - 25 = 26 \geq 25$

Here, $n = 26 \geq 25$, again 1 quarter is added to decrease $n = 26$ by 25

then, $d_1 = 1 + 1 = 2$

$n = n_{old} - 25 = 26 - 25 = 1 < 25$

Thus the first pass ends.

Second Pass (Dimes)

$c_2 = 10$

Since, $n = 1 < 10$, the second pass ends.

Third Pass (Nickels)

$c_3 = 5$

Since, $n = 1 < 5$, the third pass ends.

Fourth Pass (Pennies)

$c_4 = 1$

Since $n = 1 \geq 1$, penny is added to decrease the total amount by 1.

$d_1 = 2$

$d_4 = 1$

$n = n_{old} - 1 = 1 - 1 = 0 < 1$

Thus the forth pass ends.

$\therefore$ 2 quarters, 0 dimes, 0 nickels and 1 penny are required to change 51 cents.

(b) Given,

Number of cents, $n = 69$

First Pass (Quarters)

$c_1 = 25$

Since, $n \geq 25$, 1 quarter is added to decrease the total amount by 25.

$d_1 = 1$

$n = n_{old} - 25 = 69 - 25 = 44$

As $n = 44 \geq 25$, again 1 quarter is added to decrease $n = 44$ by 25

$d_1 = 1 + 1 = 2$

$n = n_{old} - 25 = 44 - 25 = 19 < 25$

Thus the first pass ends.

Second Pass (Dimes)

$c_2 = 10$

Since $n = 19 \geq 10$, 1 dime is added to decrease the total amount by 10.

$d_1 = 2$

$d_2 = 1$

$n = n_{old} - 10 = 19 - 10 = 9 < 10$

Thus the second pass ends.

Third Pass (Nickels)

$c_3 = 5$

Since $n = 9 \geq 5$, 1 nickel is added to decrease the total amount by 5.

$d_1 = 2$

$d_2 = 1$

$d_3 = 1$

$n = n_{old} - 5 = 9 - 5 = 4 < 5$

Thus the third pass ends.

Fourth Pass (Pennies)

$c_4 = 1$

Since $n = 4 \geq 1$,1 penny is added to decrease the total amount by 1

$d_1 = 2$

$d_2 = 1$

$d_3 = 1$

$d_4 = 1$

$n = n_{old} - 1 = 4 - 1 = 3 \geq 1$

Since, $n = 3 \geq 1$, again 1 penny is added to decrease $n = 3$ by 1.

$d_1 = 2$

$d_2 = 1$

$d_3 = 1$

$d_4 = 1 + 1 = 2$

$n = n_{old} - 1 = 3 - 1 = 2 \geq 1$

Since, $n = 2 \geq 1$, again 1 penny is added to decrease $n = 2$ by 1.

$d_1 = 2$

$d_2 = 1$

$d_3 = 1$

$d_4 = 2 + 1 = 3$

$n = n_{old} - 1 = 2 - 1 = 1 \geq 1$

Again, $n \geq 1$, 1 penny is added

$d_1 = 2$

$d_2 = 1$

$d_3 = 1$

$d_4 = 3 + 1 = 4$

$n = n_{old} - 1 = 1 - 1 = 0 < 1$

Thus the forth pass ends.

$\therefore$ To change 69 cents, 2 quarters, 1 dime, 1 nickel and 4 pennies are required.

Q35. Show that the deferred acceptance always terminates with a stable assignment.

Answer :

Assuming that the algorithm does not end with a stable assignment.

Consider there exists, a man m and a woman w such that m prefers w over his assigned partner and w prefers m over her assigned partner. That means m would have proposed to w and w would have rejected the proposal of that assigned partner m in an iteration of the algorithm.

But, from the algorithm, w cannot reject a proposal of her assigned partner. Thus, it is contradict to the assumption.

$\therefore$ The deferred acceptance algorithm always terminates with a stable assignment.

3.1.2 The Growth of Functions

Q36. Write a short notes on,

(i) Big - O notation

(ii) Big-omega (Big-Ω) notation

(iii) Big-theta (Big-Θ) notation.

Answer :

1. **Big-O Notation**

For answer refer Unit-III, Q7.

2. **Big-Omega (Big-Ω)Notation**

Let $f(n)$ and $g(n)$ be two non-negative functions. $f(n)$ is said to be $\Omega(g(n))$ if and only if there exists positive constants 'k' and 'n_0' such that,

$f(n) \geq k * g(n)$ for all non-negative values of n where $n \geq n_0$.

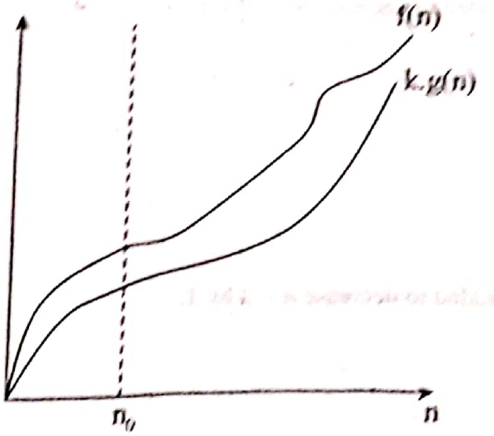


Figure (1): Omega Notation

The above definition states that the function 'f' is at least 'k' times the function 'g' when n is greater than 'n_0'.

This notation provides a lower bound for the function 'f' i.e., the function $g(n)$ is a lower bound on the value of $f(n)$ for all n, where $n \geq n_0$.

3. Big-Theta (big-Θ) Notation

Let $f(n)$ and $g(n)$ be two non-negative functions. $f(n)$ is said to be $\Theta(g(n))$ if and only if there exists positive constants 'k_2' and 'n_0' such that,

$k_1 g(n) \leq f(n) \leq k_2 g(n)$ for all non-negative values of n, where $n \geq n_0$.

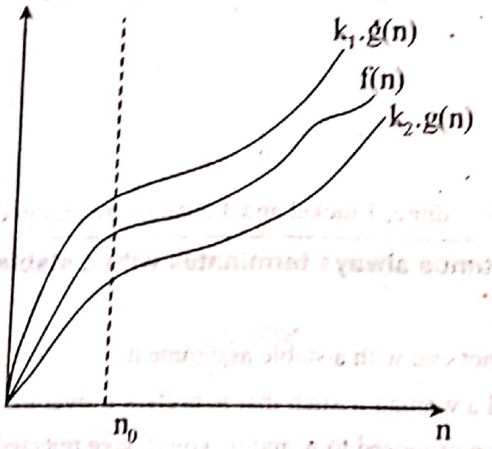


Figure (2): Theta Notation

The above definition states that the function $f(n)$ lies between k_1 times the function $g(n)$ and k_2 times the function $g(n)$, where k_1 and k_2 are positive constants.

This notation provides both lower and upper bound for the function $f(n)$ i.e., $g(n)$ is both lower and upper bound on the value of $f(n)$, for large n. In other words theta notation says that $f(n)$ is both $O(g(n))$ and $\Omega(g(n))$ for all n, where $n \geq n_0$.

If $f(n) = a_p n^p + a_{p-1} n^{p-1} + \ldots + a_1 n + a_0$, where $a_0, a_1 \ldots a_p$ are real numbers with $a_p \neq 0$. Then $f(n)$ is of order n^p.

Q37. Show that $n! = O(n^n)$ and $\lg n! = O(n \lg n)$.

Answer :

Consider,

$n! = 1.2.3.4 \ldots\ldots n(n-1)$

$\Rightarrow$ $n! = 1.2.3.4 \ldots\ldots n(n-1) \leq n.n.n \ldots\ldots n = n^n$; $n > 1$

By considering $c = 1$ it can be said that $n! = O(n^n)$

$\therefore$ $n! = O(n^n)$

Also, $n! \leq n^n$

Applying log on both sides,

$\log n! \leq \log n^n$

$\log n! \leq n \log n$

$\therefore$ $\log n! = O(n \log n)$ with $C = 1$

Q38. If $f_1(n) = O(g_1(n))$ and $f_2(n) = O(g_2(n))$ then prove that $(f_1 + f_2)(n) = O(\max\{|g_1(n)|, |g_2(n)|\})$

Answer : Model Paper-3, Q6(a)

Given that,

$f_1(n) = O(g_1(n))$

$f_2(n) = O(g_2(n))$

From the definition of Big-O notation,

$|f_1|n|| \le c_1 |g_1(n)| \; \forall \; n > n_1$ and

$|f_2|n|| \le c_2 |g_2(n)| \; \forall \; n > n_2$

Where,

c_1, c_2, n_1, n_2 – Positive constants

Consider,

$C = \text{Max}\{c_1, c_2\}$

$n_0 = \text{Max}\{n_1, n_2\}$

and $g(n) = \text{Max}\{|g_1(n)|, |g_2(n)|\}$

Then,

$$|f_1(n) + f(n_2)| \le c_1 |g_1(n)| + c_2 |g_2(n)|$$
$$\le c\,|g(n)| + c\,|g(n)| \quad (\text{when } n > n_0)$$
$$= 2c\,|g(n)|$$
$$= 2\,O(\text{Max}\{|g_1(n)|, g_2(n)|\})$$

$[\because c = \text{Max}\, c_1, c_2, g(n) = \text{Max}\{|g_1 n|, |g_2(n)|\}]$

$\therefore f_1(n) + f_2(n) = O(g(n))$

$\Rightarrow (f_1 + f_2)(n) = O(\max\{|g_1(n)|, |g_2(n)|\})$

Q39. If $f_1(n) = O(g_1(n))$ and $f_2(n) = O(g_2(n))$ then $(f_1 f_2)(h) = O(g_1(n).\, g_2(n))$.

Answer : Model Paper-2, Q6(b)

Given that,

$f_1(n) = O(g_1(n))$

$f_2(n) = O(g_2(n))$

The above functions f_1 and f_2 contains constant c_1, c_2, n_1 and n_2 such that

$|f_1(n)| \le c_1 |g_1(n)|$ for all $n > n_1$

$|f_1(n)| \le c_1 |g_1(n)|$ for all $n > n_2$

Consider,

$c = c_1 c_2$

$n_0 = \text{Max}\{n_1, n_2\}$

Hence,

$$|(f_1 f_2)(n)| = |f_1(n).\, f_2(n)|$$
$$= |f_1(n)|.\, |f_2(n)| \quad \ldots (1)$$

Since,

$|f_1(n)| \le c_1 |g_1(n)|$ and $|f_2(n)| \le c_2 |g_2(n)|$

Substituting the corresponding values in equation (1)

$$\Rightarrow |(f_1 f_2)(n)| = |f_1(n)|\,.\,|f_2(n)|$$
$$= \le c_2 |g_1(n)|.\, c_2 |g_2(n)|$$
$$= O|g_1(n).g_2(n)| \quad (\text{when } n \ge n_0)$$

$\therefore \; (f_1.f_2)(n) = O(g_1(n).\, g_2(n))$

Q40. Determine whether each of these functions is $O(x^2)$.

(a) $f(x) = 17x + 11$

(b) $f(x) = x^2 + 1000$

(c) $f(x) = x \log x$

(d) $f(x) = x^4/2$

(e) $f(x) = 2^x$

(f) $f(x) = \lfloor x \rfloor . \lceil x \rceil$

Answer : Model Paper-5, Q6(a)

Given,

$O(x^2)$

i.e., $g(x) = x^2$

[$\because$ From the definition of the Big-O Notation]

(a) Given function is,

$f(x) = 17x + 11$

Let $k = 18$ such that $x > 18$

Then,

$$|f(x)| = |17x + 11|$$
$$\le |17x| + |11|$$
$$= 17x + 11 \le 18x < x.x = x^2 = |x^2| = 1|x^2|$$

Here, $C = 1$

$\therefore \; f(x) = O(x^2)$ with $k = 18$ and $C = 1$

(b) Given function is,

$f(x) = x^2 + 1000$

Let $k = 100$

Such that $x > 100$

$\Rightarrow \; x^2 > 100^2 = 10000$

Then,

$|f(x)| = |x^2 + 1000| \le |x^2| + |1000| = x^2 + x^2 = 2x^2 = 2x^2$

Here, $C = 2$

$\therefore \; f(x) = O(x^2)$ with $k = 100$ and $C = 2$

(c) Given function is,

$f(x) = x \log x$

Since, $\log x \le x$ for $x > 0$,

$\Rightarrow \; |f(x)| = |x.\log x| \le |x.\, x| = |x^2| = 1|x^2|$

Here, $C = 1$

$\therefore \; f(x) = O(x^2)$ with $k = 0$ and $C = 1$

(d) Given function is,

$f(x) = \dfrac{x^4}{2}$

Since, $x^4 > x$

$\Rightarrow \; \left|\dfrac{x^4}{2}\right| \le C|x|$

Thus, C holds many values (i.e., not one value) for all large values of x

$\therefore \; f(x)$ is not $O(x^2)$

(e) Given function is,

$f(x) = 2^x$

Assuming $f(x) = O(x^2)$

Then,

$f(x) = |2^x| = 2^x \le Cx^2 = C|x^2|; x > k,$

Where,

C - constant

$2^x \le Cx^2$

$\Rightarrow \quad C \ge \frac{2^x}{x^2}$

$\lim_{x \to +\infty} C \ge \lim_{x \to +\infty} \frac{2^x}{x^2}$

$\Rightarrow \quad C \ge +\infty$

Since, C is not a constant,

$\therefore$ $f(x)$ is not $O(x^2)$

(f) Given function is,

$f(x) = \lfloor x \rfloor . \lfloor x \rfloor$

For $x > 1$,

$|\lfloor x \rfloor . \lfloor x \rfloor| = \lfloor x \rfloor . \lfloor x \rfloor \le x . (x+1) = x^2 + x \le x^2 + x^2 = 2x^2 = 2|x^2|$

Here, $C = 2$

$\therefore$ $f(x) = O(x^2)$ with $k = 1$ and $C = 2$

Q41. Find the least integer n such that f(x) is $O(x^n)$ for each of these functions

(a) $f(x) = 2x^2 + x^3 \log x$

(b) $f(x) = 3x^5 + (\log x)^4$

(c) $f(x) = (x^4 + x^2 + 1)/(x^4 + 1)$

(d) $f(x) = (x^3 + 5 \log x)/(x^4 + 1)$

Answer :

Model Paper-1, Q7(a)

(a) Given function is,

$f(x) = 2x^2 + x^3 \log x$

Since,

$\log x \le x$ for $x > 0$

$f(x) = 2x^2 + x^3 \log x \le 2x^2 + x^3 . x \le 2x^2 + x^4$

For $f(x)$ is $O(x^n)$, largest power of x is the smallest n.

i.e., $n = 4$

Let $k = 2$ such that $x > 2$,

Then,

$|f(x)| = |2x^2 + x^3 \log x| \le |2x^2 + x^4| \le |2x^2| + |x^4|$

$= 2x^2 + x^4 \le x^4 + x^4$

$= 2x^4$

$= 2|x^4|$

Here, $C = 2$

$\therefore$ $f(x) = O(x^4)$ with $k = 2$ and $C = 2$

(b) Given function is,

$f(x) = 3x^5 + (\log x)^4$

For $f(x)$ is $O(x^n)$, largest power of x is the smallest n

i.e., $n = 5$

Let $k = 1$ such that $x > 1$

Then,

$|f(x)| = |3x^5 + (\log x)^4| \le |3x^5| + |(\log x)^4|$

$= 3x^5 + (\log x)^4 \le 3x^5 + x^4 < 3x^5 + x^5$

$= 4x^5$

$= 4|x^5|$

Here, $C = 4$

$\therefore$ $f(x) = O(x^5)$ with $k = 1$ and $C = 4$

(c) Given function is,

$f(x) = \frac{x^4 + x^2 + 1}{x^4 + 1}$

From long division method,

$x^4 + 1 \overline{)\, x^4 + x^2 + 1 \,(\, 1}$

$x^4 + 1$

$(-) \quad (-)$

x^2

$\Rightarrow \quad f(x) = \frac{x^4 + x^2 + 1}{x^4 + 1} = 1 + \frac{x^2}{x^4 + 1}$

For $f(x)$ is $O(x^n)$, largest power of x (of quotient) is the smallest n

i.e., $n = 0$

Let $k = 0$ such that $x > 0$

$|f(x)| = \left|1 + \frac{x^2}{x^4 + 1}\right| \le |1| + \left|\frac{x^2}{x^4 + 1}\right|$

$= 1 + \frac{x^2}{x^4 + 1} \le 1 + 1 < 2$

$= 2.|1|$

$= 2|x^0|$

Here, $C = 2$

$\therefore$ $f(x) = O(x^0)$ with $k = 0$ and $C = 2$

(d) Given function is,

$f(x) = \frac{x^3 + 5 \log x}{x^4 + 1}$

Since, the function is smaller than 1 for large values of x, then for $f(x)$ is $O(x^n)$

$n = 0$

Let $k = 2$ such that $x > 2$

Then,

$|f(x)| = \left|\frac{x^3 + 5 \log x}{x^4 + 1}\right| \le \left|\frac{x^3 + 5x}{x^4 + 1}\right| \le 1$

$= |x^0|$

Here, $C = 1$

$\therefore$ $f(x) = O(x^0) = O(1)$ with $k = 3$ and $C = 1$

Q42. Arrange the functions $\sqrt{n}$, 1000 log n, n log n, 2n!, 2^n, 3^n, and $n^2/1,000,000$ in a list so that each function is big-O of the next function.

Answer : Model Paper-4, Q6(b)

Given functions are,

$\sqrt{n}$, 1000 log$n$, n log$n$, $2n!$, 2^n, 3^n and $\frac{n^2}{1,000,000}$

The given functions are represented as,

$\sqrt{n}$ - Polynomial function of degree $\frac{1}{2}$

1000 log$n$ - Logarithmic function

n log n - Logarithmic and polynomial function of degree 1

$2n!$ - Factorial function

2^n - Exponential function with base 2

3^n - Exponential function with base 3

$\frac{n^2}{100000}$ - Polynomial function of degree 2

The increasing order of functions in Big-O notations is,

$10001\log n < \sqrt{n} < n \log n < \frac{n^2}{1000000} < 2^n < 3^n < 2n!$

$\therefore$ The order of arranging the list such that each function is Big-O of next function is,

1000 log n, $\sqrt{n}$, n log n, $\frac{n^2}{1000000}$, 2^n, 3^n, $2n!$

Q43. Suppose that you have two different algorithms for solving a problem. To solve a problem of size n, the first algorithm uses exactly n (log n) operations and the second algorithm uses exactly $n^{3/2}$ operations. As n grows, which algorithm uses fewer operations?

Answer :

Given,

First Algorithm uses n log n operations

Second Algorithm uses $n^{3/2}$ operations

$f(n) = n \log n$

$g(n) = n^{3/2}$

For $x > 0$, $\log n < \sqrt{n}$

For $k > 1$ $\log(n) > 0$

Let $k = 1$ such that $n > 1$.

Then, $|f(n)| = |n \log n| = n \log n < n\sqrt{n} = n \cdot n^{1/2} = n^{3/2} = |n^{3/2}| = 1\ |n^{3/2}|$

Here, $C = 1$

$\therefore$ $f(n) = n \log n$ is $O(n^{3/2})$ with $k = 1$ and $C = 1$

As n grows, n log n grows slower than $n^{3/2}$. Thus first algorithm uses fewer operations.

Q44. (a) Show that $3x^2 + x + 1$ is $\Theta(3x^2)$ by directly finding the constants k, C_1 and C_2 in Exercise 33.

(b) Express the relationship in part (a) using a picture showing the functions $3x^2 + x + 1$, C_1, $3x^2$, and C_2. $3x^2$, and the constant k on the x-axis, where C_1, C_2 and k are the constants you found in part (a) to show that $3x^2 + x + 1$ is $\Theta(3x^2)$.

Answer :

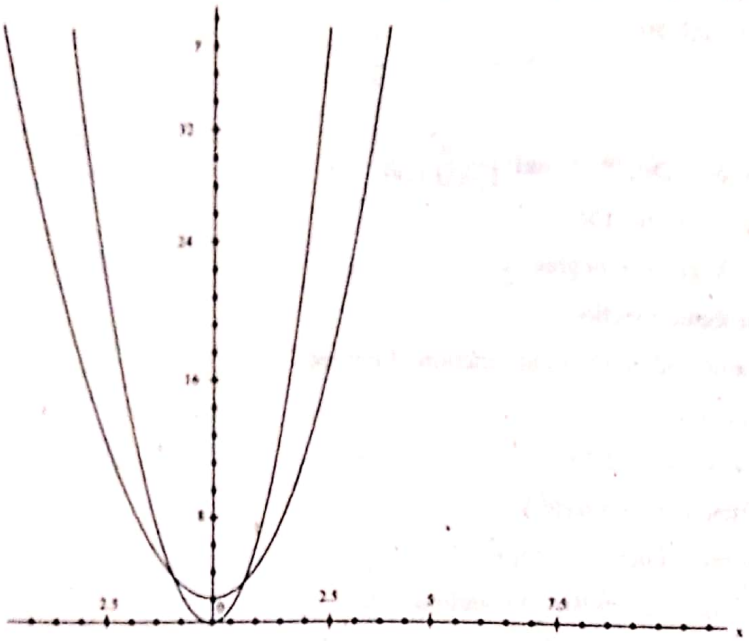
(a) Given function is,

$3x^2 + x + 1$

For $f(x)$ is $\Theta(g(x))$, if $C_1|g(x)| \le f(x)| \le C_2|g(x)|$ then $C_1 = 0$, $C_2 = 2$, $k = 1$

Since $0 \le |3x^2 + x + 1| \le 6x^2$ for $x \ge 1$

(b) The graph of the functions that show $3x^2 + x + 1$ is $\Theta(3x^2)$ is as follows.



Figure

Q45. Show that $x^6y^3 + x^4y^4 + x^3y^5$ is $\Omega(x^3y^3)$.

Answer :

Given function is,

$$f(x, y) = x^6y^3 + x^4y^4 + x^3y^5$$

Let $k_1 = k_2 = 1$

For $x > 1$ and $y > 1$,

$$|f(x, y)| = |x^6y^3 + x^4y^4 + x^3y^5|$$
$$= x^6y^3 + x^4y^4 + x^3y^5$$

Since $x > 1$ and $y > 1$,

$$x^6 > x^3,\ x^4 > x^3,\ y^4 > y^3 \text{ and } y^5 > y^4$$

$$\Rightarrow \quad |f(x, y)| > x^3y^3 + x^3y^3 + x^3y^3$$
$$= 3x^3y^3$$
$$= 3|x^3y^3|$$

Here, $C = 3$.

$\therefore$ $f(x, y) = x^6y^3 + x^4y^4 + x^3y^5$ is $\Omega(x^3y^3)$ with constants $C = 3$, $k_1 = 1$ and $k_2 = 1$.

Q46. Let H_n be the n^{th} harmonic number

$$H_n = 1 + \frac{1}{2} + \frac{1}{3} + \ldots\ldots + \frac{1}{n}$$

Show that H_n is O (log n). [Hint : First establish the inequality

$$\sum_{j=2}^{n} \frac{1}{j} = \int_1^n \frac{1}{x}\,dx$$

by showing that the sum of the areas of the rectangles of height 1/j with base from j – 1 to j for j = 2, 3, ... n, is less than the area under the curve y = 1/x from 2 to n].

Answer :

Given,

n^{th} Harmonic number,

$$H_n = 1 + \frac{1}{2} + \frac{1}{3} + \ldots\ldots + \frac{1}{n}$$

$$= 1 + \sum_{j=2}^{n} \frac{1}{j} \qquad \ldots(1)$$

And $y = f(x) = \frac{1}{x}$

And, also given $\frac{1}{j}$ is height and $j-1$ to (j) is base of rectangle and $j = 2, 3, \ldots\ldots n$

For a non negative x, $f(x)$ is strictly decreasing.

Area of rectangle < Area of curve

$\Rightarrow$ (base × height) < area under a curve

$\Rightarrow \frac{1}{j} \times (j-(j-1)) < \int_{j-1}^{j} \frac{1}{x} dx$

$\Rightarrow \frac{1}{j} < \int_{j-1}^{j} \frac{1}{x} dx$

$\Rightarrow \sum_{j=2}^{n} \frac{1}{j} < \sum_{j=2}^{n} \int_{j=1}^{j} \frac{1}{x} dx \quad [\because j = 2, 3, \ldots\ldots n]$

$\Rightarrow \sum_{j=2}^{n} \frac{1}{j} < \int_{1}^{n} \frac{1}{x} dx$

$\Rightarrow 1 + \sum_{j=2}^{n} \frac{1}{j} < 1 + \int_{1}^{n} \frac{1}{x} dx$

$\Rightarrow H_n \le 1 + \int_{1}^{n} \frac{1}{x} dx \quad [\because$ From equation (1)]

$\Rightarrow H_n \le 1 + \left[ln x\right]_1^n = 1 + [ln\, n - ln\, 1] = 1 + ln\, n$

$\Rightarrow H_n \le ln\, n + ln\, n$, when $n > 3$

$\Rightarrow H_n \le 2\, ln\, n$

$\Rightarrow H_n \le \frac{2}{\log e} \log n$, when $n > 3$

$\Rightarrow |H_n| \le \frac{2}{\log e} |\log n|$, when $n > 3$

Here, $C = \frac{2}{\log e}$

$\therefore$ H_n is $O(ln\, n)$ with $k = 3$ and $C = \frac{2}{\log e}$

Q47. For each of these pairs of functions, determine whether f and g are asymptotic.

(a) $f(x) = \log(x^2 + 1)$, $g(x) = \log x$

(b) $f(x) = 2^{x+3}$, $g(x) = 2^{x+7}$

(c) $f(x) = 2^{2x}$, $g(x) = 2^{x^2}$

(d) $f(x) = 2^{x^2+x+1}$, $g(x) = 2^{x^2+2x}$

Model Paper-2, Q7(a)

Answer :

(a) Given functions are,

$f(x) = \log(x^2 + 1)$

$g(x) = \log x$

Dividing $f(x)$ by $g(x)$

$\Rightarrow \frac{f(x)}{g(x)} = \frac{\log(x^2+1)}{\log x}$

Applying $\lim_{x \to \infty}$ on both sides

$\lim_{x \to \infty} \frac{f(x)}{g(x)} = \lim_{x \to \infty} \frac{\log(x^2+1)}{\log x}$

From L'Hospital's rule,

$\lim_{x \to \infty} \frac{f(x)}{g(x)} = \lim_{x \to \infty} \frac{d\left(\log(x^2+1)\right)}{d(\log x)}$

$= \lim_{x \to \infty} \frac{\frac{1}{x^2+1}}{\frac{1}{x}}$

$= \lim_{x \to \infty} \frac{x}{x^2+1}$

Dividing numerator and denominator by x

$\Rightarrow \lim_{x \to \infty} \frac{f(x)}{g(x)} = \lim_{x \to \infty} \frac{1}{x + \frac{1}{x}}$

$= \frac{1}{\infty + 0}$

$= 0 \ne 1$

$\therefore$ f and g are not asymptotic.

(b) Given functions are,

$f(x) = 2^{x+3}$

$g(x) = 2^{x+7}$

Dividing $f(x)$ by $g(x)$

$\frac{f(x)}{g(x)} = \frac{2^{x+3}}{2^{x+7}}$

$= 2^{(x+3)-(x+7)}$

$= 2^{-4}$

$= 16$

Applying $\lim_{x \to \infty}$ on both sides

$\lim_{x \to \infty} \frac{f(x)}{g(x)} = \lim_{x \to \infty} 16 = 16 \ne 1$

$\therefore$ f and g are not asymptotic.

(c) Given functions are,

$f(x) = 2^{2x}$

$g(x) = 2^{x^2}$

Dividing $f(x)$ by $g(x)$

$\frac{f(x)}{g(x)} = \frac{2^{2x}}{2^{x^2}}$

$= 2^{2x - x^2}$

Applying $\lim_{x \to \infty}$ on both sides

$\lim_{x \to \infty} \frac{f(x)}{g(x)} = \lim_{x \to \infty} 2^{2x - x^2}$

$= 2^{\infty}$

$= \infty \ne 1$

$\therefore$ f and g are not asymptotic.

(d) Given functions are,

$f(x) = 2^{x^2+x+1}$

$g(x) = 2^{x^2+2x}$

Dividing $f(x)$ by $g(x)$

$\frac{f(x)}{g(x)} = \frac{2^{x^2+x+1}}{2^{x^2+2x}}$

$= 2^{(x^2+x+1)-(x^2+2x)}$

$= 2^{-x+1}$

$= 2^{1-x}$

$= \frac{2}{2^x}$

Applying $\lim_{x \to \infty}$ on both sides

$$\lim_{x \to \infty} \frac{f(x)}{g(x)} = \lim_{x \to \infty} \frac{2}{2^x}$$

$$= \frac{2}{\infty}$$

$$= 0 \neq 1$$

$\therefore$ f and g are not asymptotic.

3.1.3 Complexity of Algorithms

Q48. What are the different measures of time complexity?

Answer :

The measures of time complexity are as follows,

1. **Average Case**
   The amount of time the algorithm takes on an average set of inputs.
2. **Worst Case**
   The amount of time the algorithm takes on the worst possible set of inputs.
3. **Best Case**
   Inputs are provided in such a way that the minimum time is required to process them. It is very easy to cheat with the best case running time. Therefore it is considered the least, worst and average case complexity of an algorithm is a numerical function of the size of the input.
4. **Size of Input**
   A program's running time is expressed as a function of the size of the input. The size of the input is typically denoted as n and the running time of the algorithm as $T(n)$. By plugging in the size of the input, a rough estimate of the cost of executing the algorithm can the obtained.

The terminology that describes the time complexity of algorithm is shown as follows,

Complexity	Terminology
$\Theta(1)$	Constant complexity
$\Theta(n)$	Linear complexity
$\Theta(n^b)$	Polynomial complexity
$\Theta\, n!$	Factorial complexity
$\Theta(\log n)$	Logarithmic complexity
$\Theta(n \log n)$	Linearithmic complexity
$\Theta(b^n)\ (b > 1)$	Exponential complexity

The problems that can be solved using an algorithm with worst case polynomial complexity is called tractable, otherwise they are called intractable.

Q49. Compute best, worst and average complexity of linear search.

Answer :

Let,

a_1 be the average case complexity of linear search.

b_1 be the best case complexity of linear search and

w_1 be the worst case complexity of linear search

(i) Average time complexity (a_1) can be computed considering two cases.

(i) Whether the key element occur in the list and

(ii) Whether the key element does not occur in the list

Suppose, if the key occur and is present at some position k then the, comparisons required to search the element is $n - k + 1$ where $1 < k < n$.

Similarly, if the key element does not occur in list then the total number of comparison required is, $n + 1$.

$\therefore$ The average time taken to search a key is given by

$$a_1 = \frac{(1+2+3+4+\ldots n)(n+1)}{n+1}$$

$$\leq \frac{(n+1)(n+2)}{2(n+1)}$$

$$= \frac{n+2}{2}$$

$$= \frac{n}{2} + \frac{2}{2}$$

$$= \frac{n}{2} + 1$$

$$= O(n) \text{ (by definition of Big } O\text{)}$$

$\therefore$ The average time complexity of linear search is $O(n)$

(ii) For best case, consider a list A containing n elements if the key element to be searched is equal to a_n (i.e., a_n = key) then only single comparison is required to search the element.

$\therefore$ The best case complexity (b_1) of linear search by keeping the execution time constant is $O(1)$.

(iii) Worst case time (w_1) can be computed only if the key element to be searched does not exist in the given list. Hence, search is performed on such list $n + 1$ times

$\therefore$ $w_1 = n + 1$

$\leq n + n$ (when the value of $n \geq 1$)

$= 2n$

$= O(n)$ (by assuming c = 2)

$\therefore$ The worst time complexity (w_1) of linear search is $O(n)$

Q50. Give a big-O estimate for the number of operations, where an operation is an addition or a multiplication, used in this segment of an algorithm (ignoring comparisons used to test the conditions in the while loop).

```
i := 1
t := 0
while i ≤ n
    t := t + i
    i := 2i
```

Answer : Model Paper-3, Q5

Given algorithm is,

```
i := 1
t := 0
while i ≤ n
t := t + i

i := 2i
```

In above algorithm i starts from 1 and t from 0. While loop iterates for $i \le n$ times. i takes values from 1 to $2i$

i.e., 1, 2, 4, ... such that,

$2^x \le n$

$\Rightarrow \quad x = \lfloor \log_2(n) \rfloor$

Then, i takes values from 1, 2, 4; ..... $2^{\lfloor \log_2(n) \rfloor}$

i.e., $\lfloor \log_2(n) \rfloor + 1$ values

Total number of operations = Number of values for i ×

Number of operations per iteration of the while-loop

$$= \left(\lfloor \log_2(n) \rfloor + 1 \right) \times 2$$

$$= 2 \lfloor \log_2(n) \rfloor + 2$$

Hence, there are $2 \lfloor \log_2(n) \rfloor + 2$ operations, while $2 \lfloor \log_2(n) \rfloor + 2$ is $O(\log n)$

Q51. (a) Show that this algorithm determines the number of 1 bits in the bit string S:

procedure bit count (S: bit string)
count := 0
while S ≠ 0
count := count + 1
S := S ∧ (S – 1)
return count {count is the number of 1s in S}

Here S–1 is the bit string obtained by changing the rightmost 1 bit of S to a 0 and all the 0 bits to the right of this to 1s. [Recall that S ∧ (S – 1) is the bitwise AND of S and S – 1].

(b) How many bitwise AND operations are needed to find the number of 1 bits in a string S using the algorithm in part (a) ?

Answer : **Model Paper-5, Q6(b)**

(a) Given algorithm is,

Procedure bit count (S: bit string)
count := 0
while $S \neq 0$
count := count + 1
$S : S \wedge (S-1)$
Return count

When bit string, $S \neq 0$ or zero string, the algorithm stops. Thus, S does not contain any 1's.

In while loop, for each iteration, S is reassigned to $S \wedge (S-1)$.

For $1 \wedge 0 = 0 \Rightarrow S \wedge (S-1)$ is the same as $S - 1$.

Thus, in every iteration of the loop, one of the 1's is changed to 0 and the count is increased by 1. The count tracks the number of ones in the bit string.

∴ This algorithm determines the number of 1 bits in the bit string S.

(b) A bit wise AND operation is done by $S := S \wedge (S-1)$ in every iteration of the while-loop.

For every iteration of the loop, one of the 1's is changed to a 0.

Therefore, the number of bitwise AND operations = number of 1's in the bit string.

Q52. Consider the following algorithm, which takes as input a sequence of n integers $a_1, a_2, \ldots a_n$ and produces as output a matrix $M = \{m_{ij}\}$ where m_{ij} is the minimum term in the sequence of integers $a_i, a_{i+1}, \ldots a_j$ for $j \ge i$ and $m_{ij} = 0$ otherwise.

initialize M so that $m_{ij} = a_i$ if $j \ge i$ and $m_{ij} = 0$ otherwise
for i := 1 to n
for j := i + 1 to n
for k := i + 1 to j
m_{ij} := min(m_{ij}, a_k)
return M {m_{ij}} {m_{ij} is the minimum term of a_i, a_{i+1} ..... a_j}

(a) Show that this algorithm uses $O(n^3)$ comparisons to compute the matrix M.

(b) Show that this algorithm uses $\Omega(n^3)$ comparisons to compute the matrix M. Using this fact and part (a), conclude that the algorithms uses $\Theta(n^3)$ comparisons. [Hint : Only consider the cases where i ≤ n/4 and j ≥ 3n/4 in the two outer loops in the algorithm].

Answer :

Given algorithm is,

Initialize M so that $m_{ij} = a_i$ if $j \ge i$ and $m_{ij} = 0$

Otherwise

for $i := 1$ **to** n
for $j := i + 1$ **to** n
for $k : i + 1$ **to** j
$m_{ij} := \min(m_{ij}, a_k)$
return $M = \{m_{ij}\}$

m_{ij} is the minimum term of $a_i, a_{i+1}, \ldots a_j$

(a) In the above algorithm, comparisons occur at

$$m_{ij} := \min(m_{ij}, a_k)$$

In this, i takes n values i.e. 1 to n, j takes $(n-1)$ values i.e. $(i+1)$ to n and k takes $(n-2)$ values i.e. $(i+1)$ to j

Total number of comparisons= product of the (maximum) number of values for i, j and k in the for-loops.

$$= n \times (n-1) \times (n-1)$$
$$= n(n-1)^2$$
$$= n(n^2 - 2n + 1) = n^3 - 2n^2 + n$$

Thus, to compute the matrix M, the number of comparisons is $n^3 - 2n^2 + n$ which is $O(n^3)$.

(b) Since the number of comparisons is $n^3 - 2n^2 + n$ which is $\Omega(n^3)$ and $O(n^3)$

∴ The number of comparisons is also $\Theta(n^3)$.

Q53. What is the largest n for which one can solve within a day using an algorithm that requires f(n) bit operations, where each bit operation is carried out in 10^{-11} second, with these functions f(n) ?

(a) log n

(b) 1000n

(c) n^2

(d) $1000n^2$

(e) n^3

(f) 2^n

(g) 2^{2n}

(h) 2^{2^n}

Answer :

Given,

$f(n)$ is number of bit operations.

Time period of each bit operation, $T = 10^{-11}$ seconds

Since the algorithm takes 1 day for its solving,

i.e. t = 1 day = 24 hours × 60 min × 60 sec

= 86400 seconds

Then, number of possible bit operations,

$$f(n) = \frac{t}{T} = \frac{86400}{10^{-11}}$$

$$\Rightarrow \quad f(n) = 8.65 \times 10^{15} \qquad ...(1)$$

(a) Given, $f(n) = \log n$ ...(2)

From equations (1) and (2),

$$\log n = 8.64 \times 10^{15}$$

$$\log_2 n = 8.64 \times 10^{15}$$

[$\because$ Bits have 2 possible values i.e. 1 and 0]

$$\therefore \quad n = 2^{\log 2n}$$

$$= 2^{8.64 \times 10^{15}}$$

(b) Given,

$$f(n) = 1000n \qquad ...(3)$$

From equations (1) and (3),

$$1000n = 8.64 \times 10^{15}$$

$$\therefore \quad n = 8.64 \times 10^{12} = 8{,}640{,}000{,}000{,}000$$

(c) Given,

$$f(n) = n \qquad ...(4)$$

From equations (1) and (4),

$$n^2 = 8.64 \times 10^{15}$$

$$\therefore \quad n = \sqrt{8.64 \times 10^{15}} \approx 9.295 \times 10^7 = 9{,}295{,}000$$

(d) Given,

$$f(n) = 1000n^2$$

From equations (1) and (5)

$$1000n^2 = 8.64 \times 10^{15}$$

$$\Rightarrow \quad n^2 = 8.64 \times 10^{12}$$

$$\therefore \quad n = \sqrt{8.64 \times 10^{12}} \approx 2.93939 \times 10^6 = 293{,}939$$

(e) Given,

$$f(n) = n^3$$

From equations (1) and (6)

$$n^3 = 8.64 \times 10^{15}$$

$$\therefore \quad n = \sqrt{8.64 \times 10^{15}} \approx 205{,}197$$

(f) Given,

$$f(n) = 2^n$$

From equations (1) and (7)

$$2^n = 8.64 \times 10^{15}$$

$$\therefore \quad n = \log_2 (8.64 \times 10^{15}) \approx 52$$

(g) Given,

$$f(n) = 2^{2n}$$

From equations (1) and (8)

$$2^{2n} = 8.64 \times 10^{15}$$

$$\Rightarrow \quad 2n = \log_2 (8.64 \times 10^{15})$$

$$\therefore \quad n = \frac{\log_2 (8.64 \times 10^{15})}{2} \approx 26$$

(h) Given,

$$f(n) = 2^{2^n}$$

From equations (1) and (9)

$$2^{2^n} = 8.64 \times 10^{15}$$

$$\Rightarrow \quad 2^n = \log_2 (8.64 \times 10^{15})$$

$$\therefore \quad n = \log_2 (\log_2 (8.64 \times 10^{15}) \approx 5$$

Q54. Give an example and analyze the worst case time complexity of the algorithm you devised in Exercise 33 of Section 3.1 for finding the first term of a sequence less than the immediately preceding term.

Answer :

Consider the following algorithm,

procedure smaller($a_1, a_2, \ldots a_n$: positive integers with $n \geq 2$)

location := 0

$i = 2$

while ($i \leq n$ **and** location = 0)

if $a_i < a_{i-1}$ **then** location := i **else** $i := i + 1$

return location

In worst-case scenerio, no term is less than the immediately preceding term.

In while loop, atmost $(n - 1)$ iterations takes place.

When $i = n + 1$, 2 comparisons occurs

$\therefore$ Number of comparisons = number of iterations Number of comparisons per iteration

$$= (n - 1).3 + 2$$

$$= 3n - 3 + 2$$

$$= 3n - 1 \text{ which is } O(n)$$

3.2 INDUCTION AND RECURSION

3.2.1 Mathematical Induction

Q55. Prove that $2 - 2 \cdot 7 + 2.7^2 - \ldots + 2(-7)^n = (1 - (-7)^{n+1})/4$ whenever n is a nonnegative integer.

Model Paper-3, Q7(a)

Answer :

Let $q(n) = 2 - 2.7 + 2.7^2 - \ldots + 2.(-7)^n = \frac{1-(-7)^{n+1}}{4}$ ... (1)

Basis Step

For $n = 0$

LHS $= 2.(-7)^0 = 2.1 = 2$

RHS $= \frac{1-(-7)^{0+1}}{4} = \frac{1-(-7)^1}{4} = \frac{1-(-7)}{4} = \frac{1+7}{4} = \frac{8}{4} = 2$

Since, LHS = RHS,

$P(0)$ is true.

Induction Step

For $n = k$, equation (1) becomes,

$2 - 2.7 + 2.7^2 - \ldots + 2.(-7)^k = \frac{1-(-7)^{k+1}}{4}$ ... (2)

For $n = (k + 1)$,

$$2 - 2.7 + 2.7^2 - \ldots + 2.(-7)^k + 2.(-7)^{k+1} = \frac{1-(-7)^{k+1}}{4} + 2.(-7)^{k+1} \quad [\because \text{From equation (2)}]$$

$$= \frac{1-(-7)^{k+1} + 8.(-7)^{k+1}}{4}$$

$$= \frac{1+(-1+8).(-7)^{k+1}}{4}$$

$$= \frac{1+7.(-7)^{k+1}}{4}$$

$$= \frac{1-(-7).(-7)^{k+1}}{4}$$

$$= \frac{1-(-7)^{k+2}}{4}$$

$$= \frac{1-(-7)^{(k+1)+1}}{4}$$

Thus, $q(k + 1)$ is also true.

$\therefore$ From the principle of mathematical induction, $q(n)$ is true for all nonnegative integers n.

Q56. Prove that 21 divides $4^{n+1} + 5^{2n-1}$ whenever n is a positive integer.

Answer :

Let $q(n)$ be "21 divides $4^{n+1} + 5^{2n-1}$"

Basis step

For $n = 1$,

$4^{n+1} + 5^{2n-1} = 4^{1+1} + 5^{2(1)-1} = 4^2 + 5^1 = 16 + 5 = 21$

Since 21 is divisible by 21, $q(1)$ is true.

Induction step ... (1)

For $n = k$, $q(n)$ becomes,

21 divides $4^{k+1} + 5^{2k-1}$

$q(k + 1)$ is also true.

For $n = k + 1$, $q(n)$ becomes,

$$4^{(k+1)+1} + 5^{2(k+1)-1} = 4^{k+2} + 5^{2k+1}$$

$$= 4.4^{k+1} + 5^2.5^{2k-1}$$

$$= 4.4^{k+1} + 25.5^{2k-1}$$

$$= 4.4^{k+1} + (21 + 4).5^{2k-1}$$

$$= 4.4^{k+1} + 21.5^{2k-1} + 4.5^{2k-1}$$

$$= 21.5^{2k-1} + 4.(4^{k+1} + 5^{2k-1})$$

Since, 21.5^{2k-1} and $(4 \cdot (4^{k+1} + 5^{2k-1}))$ are divisible by 21, then $q(n)$ is true by the principle of mathematical induction.

Q57. Use mathematical induction to show that n people can divide a cake (where each person gets one or more separate pieces of the cake) so that the cake is divided fairly, that is, in the sense that each person thinks he or she got at least $(1/n)^{th}$ of the cake. [Hint: For the inductive step, take a fair division of the cake among the first k people, have each person divide their share into what this person thinks are k + 1 equal portions, and then have the $(k+1)^{st}$ person select a portion from each of the k people. When showing this produces a fair division for k + 1 people, suppose that person k + 1 thinks that person i got p_i of the cake where $\sum_{i}^{k} p_i = 1$].

Answer :

Let $p(n)$ be "n people can divide a cake so that the cake is divided fairly".

Basis Step

Let $n = 2$.

Then, the first person cuts the cake in half while the second person chooses the slice that is largest.

Both the persons believe that, they received at least one half of the cake.

$\therefore$ $p(2)$ is true.

Induction Step

For $n = k$,

Assuming that, k people can divide a cake fairly.

For $n = k + 1$,

Since, first k people divide the cake fairly, each of them divide their slice in $k+1$ pieces. While the $(k+1)^{th}$ person choose the largest piece of the $k + 1$ pieces from each of the k persons.

The first k people then believe that they received at least $(1/(k+1))^{th}$ of the cake, since they received k of the $k+1$ slices from their $\left(\frac{1}{k}\right)^{th}$ piece.

Then,

$$\frac{1}{k}\cdot\frac{k}{k+1} = \frac{1}{k+1}$$

The $(k+1)^{th}$ person believes that at least $\left(\frac{1}{k+1}\right)^{th}$ of the cake is received as the larger k slices from the $k+1$ pieces of the $\left(\frac{1}{k}\right)^{th}$ slice is choosen.

Then,

$$k.\frac{1}{k+1}\cdot\frac{1}{k} = \frac{1}{k+1}$$

Thus $p(k+1)$ is true.

$\therefore$ By the principle of mathematical induction, $p(n)$ is true for all positive integers $n > 1$.

Q58. Show that,

$[(p_1 \to p_2) \wedge (p_2 \to p_3) \wedge \ldots \wedge (p_{n-1} \to p_n)] \to [(p_1 \wedge p_2 \wedge \ldots \wedge p_{n-1}) \to p_n]$

is a tautology whenever $p_1, p_2, \ldots, p_n$ are propositions, where $n \geq 2$.

Answer :

Let, $q(n)$ be "$[(p_1 \to p_2) \wedge (p_2 \to p_3] \wedge \ldots \wedge (p_{n-1} \to p_n)] \to [(p_1 \wedge p_2 \wedge \ldots \wedge p_{n-1}) \to p_n]$ is a tautology"

Basis Step

For $n = 2$, $(p_1 \to p_2) \to (p_{2-1} - p_2)$

$\Rightarrow$ $(p_1 \to p_2) \to (p_1 \to p_2)$

$\therefore$ $q(2)$ is true

Induction Step

For $n = k$, $q(n)$ becomes,

$[(p_1 \to p_2) \wedge (p_2 \to p_3) \wedge \ldots \wedge (p_{k-1} \to p_k)] \to [(p_1 \wedge p_2 \wedge \ldots \wedge p_{k-1}) \to p_k]$ ...(1)

For $n = k$, L.H.S of $q(n)$ becomes,

$[(p_1 \to p_2) \wedge (p_2 \to p_3) \wedge \ldots \wedge (p_{k-1} \to p_k) \wedge (p_k \to p_{k+1})]$

$\equiv [((p_1 \to p_2) \wedge (p_2 \to p_3) \wedge \ldots \wedge (p_{k-1} \to p_k)) \wedge (p_k \to p_{k+1})]$

$\equiv [([p_1 \wedge p_2 \wedge \ldots \wedge p_{k-1}] \to p_k) \wedge (p_k \to p_{k+1})]$

[$\because$ From equation (1)]

$\equiv [p_1 \wedge p_2 \wedge \ldots \wedge p_{k-1} \wedge p_k] \to p_{k+1}$

Thus $q(k+1)$ is true.

$\therefore$ By the principle of mathematical induction, $q(n)$ is true for all positive integers n.

Q59. Let $a_1, a_2, \ldots, a_n$ be positive real numbers. The arithmetic mean of these numbers is defined by,

$A = (a_1 + a_2 + \ldots + a_n)/n$,

and the geometric mean of these numbers is defined by,

$G = (a_1 a_2 \ldots a_n)^{1/n}$.

Use mathematical induction to prove that $A \geq G$.

Answer :

Given that,

$a_1, a_2, \ldots a_n$ be positive real numbers

Arithmetic mean, $A_n = \frac{a_1 + a_2 + \ldots + a_n}{n}$ ...(1)

Geometric mean, $G_n = (a_1 a_2 \ldots a_n)^{1/n}$ ...(2)

Let $q(n)$ be "$A_n \geq G_n$"

Basis Step

For $n = 1$, equation (1) becomes,

$A_1 = \frac{a_1}{1} = a_1$

Equation (2) becomes, $G_1 = (a_1)^{1/1} = a_1 \Rightarrow A_1 = G_1$

$\therefore$ $q(1)$ is true

Induction Step

For $n = k$, $q(n)$ becomes,

$A_k \geq G_k$

For $n = k+1$, consider a value 'x' such that,

$f(x) = A_{k+1} - G_{k+1}$

$$\Rightarrow \quad f(x) = \frac{a_1 + a_2 + \ldots + a_k + x}{k+1} - (a_1 a_2 \ldots a_k x)^{1/(k+1)} \qquad \ldots (3)$$

Differentiating with respect to x on both sides.

$$f'(x) = \left(\frac{1}{k+1}\right) - \left(\frac{1}{k+1}(a_1 a_2 \ldots a_k)^{1/(k+1)} x^{(1/(k+1))-1}\right)$$

$$= \frac{1}{k+1} - \frac{1}{k+1}(a_1 a_2 \ldots a_k)^{1/(k+1)} x^{-k/(k+1)}$$

$$= \frac{1}{k+1}\left(1 - (a_1 a_2 \ldots a_k)^{1/(k+1)} x^{-k/(k+1)}\right)$$

For x_0, $f'(x_0) = 0$

$$\Rightarrow \quad \frac{1}{k+1}\left(1 - (a_1 a_2 \ldots a_k)^{1/(k+1)} x_0^{-k/(k+1)}\right) = 0$$

$$1 - (a_1 a_2 \ldots a_k)^{1/(k+1)} x_0^{-k/(k+1)} = 0$$

$$1 = (a_1 a_2 \ldots a_k)^{1/(k+1)} x_0^{-k/(k+1)}$$

$$\frac{1}{(a_1 a_2 \ldots a_k)^{1/(k+1)}} = x_0^{-k/(k+1)}$$

$$(a_1 a_2 \ldots a_k)^{1/(k+1)} = x_0^{k/(k+1)}$$

$$(a_1 a_2 \ldots a_k) = x_0^k$$

$$(a_1 a_2 \ldots a_k)^{1/k} = x_0$$

Since f is convex, $x_0 = (a_1 a_2 \ldots a_k)^{1/k}$ is a global minimum of the function f

Substituting $x = x_0$ value in equation (3)

$$\Rightarrow \quad f(x_0) = \frac{a_1 + a_2 + \ldots + a_k + (a_1 a_2 \ldots a_k)^{1/k}}{k+1} - (a_1 a_2 \ldots a_k (a_1 a_2 \ldots a_k)^{1/k})^{1/(k+1)}$$

$$= \frac{a_1 + a_2 + \ldots + a_k}{k+1} + \frac{(a_1 a_2 \ldots a_k)^{1/k}}{k+1} - \left((a_1 a_2 \ldots a_k)^{\frac{1}{k}+1}\right)^{\frac{1}{k+1}}$$

$$= \frac{a_1 + a_2 + \ldots + a_k}{k+1} + \frac{(a_1 a_2 \ldots a_k)^{1/k}}{k+1} - \left((a_1 a_2 \ldots a_k)^{\left(\frac{1+k}{k}\right)}\right)^{\frac{1}{k+1}}$$

$$= \frac{a_1 + a_2 + \ldots + a_k}{k+1} + \frac{(a_1 a_2 \ldots a_k)^{1/k}}{k+1} - (a_1 a_2 \ldots a_k)^{1/k}$$

$$= \frac{a_1 + a_2 + \ldots + a_k}{k+1} + \frac{(a_1 a_2 \ldots a_k)^{1/k}}{k+1} - \frac{(k+1)(a_1 a_2 \ldots a_k)^{1/k}}{k+1}$$

$$= \frac{a_1 + a_2 + \ldots + a_k}{k+1} - \frac{k\,(a_1 a_2 \ldots a_k)^{1/k}}{k+1}$$

$$= \frac{k}{k+1}\left(\frac{a_1 + a_2 + \ldots + a_k}{k} - (a_1 a_2 \ldots a_k)^{1/k}\right)$$

$$= \frac{k}{k+1}(A_k - G_k)$$

Since $q(k)$ is true for $A_k \geq G_k$, thus $f(x_0) \geq 0$.

$\therefore$ $q(k+1)$ is true

$\therefore$ $A \geq G$

Q60. Use the well-ordering property to show that the following form of mathematical induction is a valid method to prove that P(n) is true for all positive integers n.

Basis step: P(1) and P(2) are true.

Inductive step: For each positive integer k, if P(k) and P(k + 1) are both true, then P(k + 2) is true.

Answer : Model Paper-4, Q7(a)

Given that,

Basis step:

$P(1)$ and $P(2)$ are true

Inductive step:

For each positive integer k,

If $P(k)$ and $P(k+1)$ are both true, then $P(k+2)$ is true.

Assuming that this induction is not valid.

Then,

Let S be the set of counter examples, $S = \{n \mid \neg P(n)\}$

$\Rightarrow \quad S \neq \phi$

By well-ordering property, S has a least element x.

Then, from basis step, $P(1)$ and $P(2)$ are true,

$1 \notin S$, $2 \notin S$ and $x \neq 1$, $x \neq 2$.

So, $x > 2$ and $P(x)$ is false, since $x \in S$.

Also, $\forall j < x$, $P(j)$ is true

From inductive step, for $k = x - 2$,

$P(x-2) \wedge P(x-1) \Rightarrow P(x)$

By modus ponens,

$P(x-2) \wedge P(x-1)$ is true.

$\therefore \quad P(x)$ is true.

But, $x \in S$ which is contradict to the assumption,

$\therefore \quad S = \phi$. Thus the proof is valid.

Q61. Show that if $I_1, I_2, \ldots I_n$ is a collection of open intervals on the real number line, $n \geq 2$, and every pair of these intervals has a nonempty intersection, that is, $I_i \cap I_j \neq \phi$ whenever $1 \leq i \leq n$ and $1 \leq j \leq n$, then the intersection of all these sets is nonempty, that is $I_1 \cap I_2 \cap \ldots \cap I_n \neq \phi$. (Recall that an open interval is the set of real numbers x with a < x < b where a and b are real numbers with a < b.

Answer :

Given,

$I_1, I_2, \ldots, I_n$ is a collection of open intervals on the real number line $n \geq 2$.

Every pair of these intervals I_i and I_j have a nonempty intersection, $I_i \cap I_j = \phi$

Let $q(n)$ be "$I_1 \cap I_2 \cap \ldots \cap I_n$".

Basis Step

Let $n = 2$,

Since, $I_i \cap I_j = \phi$

$\Rightarrow \quad I_1 \cap I_2 = \phi$

$\therefore \quad q(2)$ is true.

Inductive Step

For $n = k$

$I_1 \cap I_2 \cap \ldots \cap I_k = \phi$

For $n = k + 1$, ...(1)

$$I_1 \cap I_2 \cap \ldots \cap I_k \cap I_{k+1} = (I_1 \cap I_2 \cap \ldots \cap I_k) \cap I_{k+1}$$
$$= \phi \cap I_{k+1} \quad [\because \text{From equation (1)}]$$
$$= \phi$$

$\therefore \quad q(k+1)$ is true.

$\therefore \quad$ The intersection of all these sets is non empty.

3.2.2 Strong Induction and Well-Ordering

Q62. What are the steps to be followed for proving the statement q(n) by strong induction. Define well-ordering property.

Answer :

The steps to be followed for proving a statement $q(n)$ for all positive integers n by strong induction are,

Basis step

Proposition $q(1)$ is verified such that it is true.

Induction step

The conditional statement $[q(1) \wedge q(2) \wedge \ldots\ldots \wedge q(k)] \rightarrow q(k+1)$ is proved for all positive integers k.

Strong induction is also called as second principle of mathematical induction or complete induction.

Infinite ladder also explains strong induction by considering rungs as,

All rungs can be reached if,

1. First rung is reached and
2. For every integer k, if first k rungs are reached then, $(k+1)^{st}$ rung can be reached.

The difference between mathematical induction and strong mathematical induction is that, in former $q(k)$ is assumed to be true, but in latter $q(k)$, $q(k-1)$, $q(k-2)$ ..... $q(1)$ are assumed to be true.

Well-ordering property

This property states that, every non-empty set of non negative integers has a least element.

Q63. A jigsaw puzzle is put together by successively joining pieces that fit together into blocks. A move is made each time a piece is added to a block, or when two blocks are joined. Use strong induction to prove that no matter how the moves are carried out, exactly n – 1 moves are required to assemble a puzzle with n pieces.

Answer : Model Paper-5, Q7(a)

Given,

A jigsaw puzzle is put together by successively joining pieces that fit together into blocks.

A move is the joining of two blocks or adding a piece to a block.

Let $q(n)$ be $n-1$ moves required to assemble a puzzle with n pieces.

Basis Step

For $n = 1$,

Puzzle of 1 piece requires 0 moves

$\therefore$ $q(1)$ is true

Inductive Step

For $n = k$,

$q(1), q(2), \ldots q(k)$ are all true

For $n = k + 1$,

Joining two blocks with j pieces and $k - j + 1$ pieces finishes the moves.

Since $q(j)$ is true, j pieces block requires $j-1$ pieces and $k-j+1$ pieces block requires $k-j+1-1 = k-j$ pieces.

$\Rightarrow$ $(j-1)+(k-j)+1 = k$

Thus $q(k+1)$ is true, since it requires k moves to assemble a puzzle with k pieces.

Q64. The well-ordering property can be used to show that there is a unique greatest common divisor of two positive integers. Let a and b be positive integers, and let S be the set of positive integers of the form as + bt, where s and t are integers.

a. Show that S is non empty.

b. Use the well-ordering property to show that S has a smallest element c.

c. Show that if d is a common divisor of a and b, then d is a divisor of c.

d. Show that c|a and c|b. [Hint : First, assume tht c|a. Then a=qc+r, where 0<r<c. Show that r S, contradicting the choice of c].

e. Conclude from (c) and (d) that the greatest common divisor of a and b exists. Finish the proof by showing that this greatest common divisor is unique.

Answer :

Given,

a and b are positive integers

S = set of positive integers of the form $as + bt$.

Where s and t are integers.

When $s = 1$ and $t = 0 \Rightarrow as + bt = a$

When $s = 0$ and $t = 1$, $as + bt = b$

(a) Since, S contains both a and b, it is nonempty.

(b) As S consists of set of positive integers, it contains a least element say c.

Thus, from well-ordering principle, c is the smallest element of s.

(c) Given,

d is a common divisor of a and b.

Then, there exists integers x and y such that,

$$a = dx$$
$$b = dx$$

Since c is the smallest element of S, then,

$$c = ae + bf$$
$$\Rightarrow \quad c = dxe + dyf = d(ex + yf)$$

Where,

e and f are integers.

Here, $ex + yf$ is an integer.

Thus d is a divisor.

(d) Assuming that c does not divide a. Then, there exist integers q and r with $0 < r < s$ such that,

$$a = cq + r$$
$$a - cq = r$$

Since, cq and a are elements of S,

$$r \in S$$

i.e., r is an element of S that is smaller than c, which is in contradiction that the c is the smallest element in S.

Thus the assumption is wrong.

$\therefore$ c divides a. Similarly $c|b$.

(e) Since, $c|a$ and $c|b$. Then $c \le d$. Also every common divisor d of a and b divides c.

Thus, c is the greatest common divisor of a and b.

However, $c \le d$ and $d \le c$ implies $c = d$. Thus the greatest common divisor is unique.

Q65. A stable assignment, defined in the preamble is called optimal for suiters if no stable assignment exists in which a suiter is paired with a suitee whom this suiter prefers to the person to whome this suitor is paired in this stable assignment. Use strong induction to show that the deferred acceptance algorithm produces a stable assignment that is optimal for suiters.

Answer :

Given,

A stable assignment is optimal for suitors if no stable assignment exists in which a suitor is paired with a suitee.

Let $q(n)$ be the statement. "The deferred acceptance algorithm produces a stable assignment that is optimal for suitors, when there are n suitors and n suitees".

Basis Step

For $n = 1$, stable assignment is optimal as it assigns 1 suitor to 1 suitee.

Thus $q(1)$ is true.

Inductive Step

Let $q(1), q(2), \ldots q(k)$ be true.

For $n = k + 1$,

Let $k + 1$ suitors and $k + 1$ suitees exists.

$k + 1$ suitors are divided into two groups P and Q, such that the deferred acceptance algorithm prouduces a stable optimal assignment for them as $q(1), q(2) \ldots q(k)$ is true.

$\therefore$ $q(k + 1)$ is true.

Thus, $q(n)$ is true.

Q66. Pick's theorem says tht the area of a simple polygon P in the plane with vertices that are all lattice points that is, points with integer coordinates) equal I(P) + B(P)/2 – 1, where I(P) and B(P) are the number of lattice points in the interior of P and on the boundary of P, respectively. Use strong induction on the number of vertices of P to prove Pick's theorem.

Answer : Model Paper-1, Q7(b)

Given,

Area of a simple polygon $P = I(P) + \frac{B(P)}{2} - 1$.

Where,

$I(P)$ - Number of lattice points in the interior of P

$B(P)$ - Number of lattice points on the boundary of P.

Let, $q(n)$ be equation (1), when P has n vertices.

Basis Step

Let P be a rectangle with vertices $(a, c), (b, d).(a, d), (b, c)$.

Then, length $= c - d$

Width $= a - b$

Then, area of a rectangle = Width × Length

Area $P = (a - b)(c - d)$

Boundary of P, $B(P)$ = 2(lattice points on horizontal side) + 2(lattice points on vertical side) ...(1)

$\Rightarrow$ $B(P) = 2(a - b + 1) + 2(c - d - 1) = 2(a - b + c - d)$

Interior lattice points, $I(P)$ = Horizontal lattice points × vertical lattice points

$\therefore$ $I(P) = (a - b - 1)(c - d - 1)$

Then,

$$I(P) + \frac{B(P)}{2} - 1 = (a - b - 1)(c - d - 1) + \frac{2(a - b + c - d)}{2} - 1$$

$$= (a - b - 1)(c - d - 1) + a - b + c - d - 1$$

$$= ac - ad - a - bc + bd + b - c + d + 1 + a - b + c - d - 1$$

$$= ac - ad - bc + bd$$

$$= a(c - d) - b(c - d)$$

$$= (a - b)(c - d)$$

$$= \text{Area } P$$

[$\because$ From equation (1)]

Inductive Step

Let $q(3), q(4), q(5) \ldots q(k)$ be true.

Let P be a polygon with $k + 1$ vertices or lattice points.

If diagonal d divides the polygon P into two simple polygons with less vertices each, Pick's formula holds for the two simple polygons.

Thus, $P(k + 1)$ is true.

Thus, Pick's theorem is proved.

3.2.3 Recursive Definitions and Structural Induction

Q67. Explain recursively defined functions with various definitions.

Answer :

Recursively defined functions define a function with the set of non-negative integers as its domain.

Basis step

The value of the function is specified at zero.

Recursive step

Considering smaller integer values and a rule is given to find its value at an integer.

❖ For finite sequence of string, the set Σ^* of string over the alphabet Σ is defined recursively by,

Basis step

$\lambda \in \Sigma^*$, where λ - empty string

Recursive step

If $w \in \Sigma^*$ and $x \in \Sigma$, then $wx \in \Sigma^*$

❖ The concatenation of two strings is recursively defined as follows

Basis step

If $w \in \Sigma^*$ then $w.\lambda = w$

Recursive step

If $w_1 \in \Sigma^*$ and $w_2 \in \Sigma^*$ and $x \in \Sigma$, then

$w_1.(w_2x) = (w_1 . w_2)x$

❖ For a set of rooted trees, recursive defined as,

Basis step

A single vertex r is a rooted tree.

Recursive step

Considering $T_1, T_2, \ldots T_n$ are disjoint rooted trees with roots $r_1, r_2, \ldots r_n$ respectively. A graph is said to be rooted tree if, a root r exists which is not present in disjoint rooted trees and adds an edge from r to each of the vertices.

❖ The set of extended binary trees can be defined recursively by following steps

Basis step

The empty set is an extended binary tree.

Recursive step

Let T_1 and T_2 be disjoint extended non-empty binary trees and $T_1 . T_2$ as extended binary tree. Then, $T_1 . T_2$ consists of a root r together with edges connecting to each of the roots of the left subtree T_1 and the right subtree T_2.

❖ The set of full binary trees can be defined recursively by following steps,

Basis step

There is a full binary tree consisting only of a single vertex r.

Recursive step

Let T_1 and T_2 are disjoint full binary trees and $T_1 . T_2$ as full binary tree. Then, $T_1 . T_2$ consists of a root r together with edges connecting to each of the roots of the left subtree T_1 and the right subtree T_2.

Q68. Give a recursive definition of the function max and min so that max($a_1, a_2, \ldots a_n$) and min($a_1, a_2, \ldots a_n$) are the maximum and minimum of the n numbers $a_1, a_2, \ldots a_n$ respectively.

Answer :

Given,

$a_1, a_2 \ldots a_n$ are n numbers,

$\max(a_1, a_2 \ldots a_n)$

$\min(a_1, a_2 \ldots a_n)$

When $n = 1$,

$\max(a_1) = a_1$

$\min(a_1) = a_1$

i.e., the maximum and minimum of a number is the number itself.

When $n = 2$,

$$\max(a_1, a_2) = \begin{cases} a_1 & \text{if } a_1 \geq a_2 \\ a_2 & \text{Otherwise} \end{cases}$$

$$\max(a_1, a_2) = \begin{cases} a_2 & \text{if } a_1 \geq a_2 \\ a_1 & \text{Otherwise} \end{cases}$$

When $n > 2$,

$$\max(a_1, a_2 \ldots a_n, a_{n+1}) = \max(\max(a_1, a_2 \ldots a_n), a_{n+1})$$

$$\min(a_1, a_2 \ldots a_n, a_{n+1}) = \min(\min(a_1, a_2, \ldots a_n), a_{n+1})$$

(or)

$$\max(a_1, a_2 \ldots a_n) = \max(\max(a_1, a_2 \ldots a_{n-1}), a_n)$$

$$\max(a_1, a_2 \ldots a_n) = \max(\max(a_1, a_2 \ldots a_{n-1}), a_n)$$

Q69. Let $a_1, a_2 \ldots\ldots a_n$, and $b_1, b_2, \ldots\ldots\ldots b_n$ be real numbers. Use the recursive definitions from Q68 to prove these.

(a) $\max(-a_1, -a_2, \ldots\ldots -a_n) = -\min(a_1, a_2, \ldots\ldots a_n)$

(b) $\max(a_1 + b_1, a_2 + b_2 \ldots\ldots a_n + b_n) \leq \max(a_1, a_2, \ldots\ldots a_n) + \max(b_1, b_2, \ldots\ldots b_n)$

(c) $\min(a_1 + b_1, a_2 + b_2 \ldots\ldots a_n + b_n)) \leq \min(a_1, a_2, \ldots\ldots a_n) + \min(b_1, b_2, \ldots\ldots b_n)$

Model Paper-2, 7(b)

Answer :

For answer refer Unit-3, Q68.

(a) Let, $q(n)$ be $\max(-a_1, -a_2, \ldots -a_n) = -\min(a_1, a_2, \ldots\ldots a_n)$

Basis Step

For $n = 1$ and $n = 2$,

$$\max(-a_1) = -a_1 = -\min(a_1)$$

$$\max(-a_1, -a_2) = \begin{cases} -a_2 & \text{if } a_1 \geq a_2 \\ -a_1 & \text{Otherwise} \end{cases}$$

$$= -\begin{cases} a_2 & \text{if } a_1 \geq a_2 \\ a_1 & \text{Otherwise} \end{cases} = -\min(a_1, a_2)$$

Thus $q(1)$ and $q(2)$ are true,

Inductive Step

For $n = k$,

$$\max(-a_1, -a_2, \ldots\ldots -a_k) = -\min(a_1, a_2, \ldots\ldots a_k) \quad \ldots(1)$$

For $n = k + 1$,

$$\max(-a_1, -a_2, \ldots\ldots -a_k, -a_{k+1}) = \max(\max(-a_1, -a_2, \ldots -a_k), -a_{k+1})$$

$$= \max(-\min(a_1, a_2, \ldots\ldots -a_k), -a_{k+1}) \quad [\because \text{From equation (1)}]$$

$$= -\min(\min(a_1, a_2, \ldots\ldots a_k), a_{k+1})$$

$$= -\min(a_1, a_2, \ldots\ldots a_k, a_{k+1})$$

Thus, $q(k + 1)$ is true.

$\therefore$ $q(n)$ is proved.

(b) Let $q(n)$ be $\max(a_1 + b_1, a_2 + b_2 \ldots\ldots a_n + b_n) \leq \max(a_1, a_2, \ldots\ldots a_n) + \max b_1, b_2 \ldots b_n)$

Basis Step

For $n = 1$ and $n = 2$,

$$\max(a_1 + b_1) = a_1 + b_1 = \max(a_1) + \max(b_1)$$

$$\max(a_1 + b_1, a_2 + b_2) = \begin{cases} a_1 + b_1 & \text{if } a_1 + b_1 \geq a_2 + b_2 \\ a_2 + b_2 & \text{otherwise} \end{cases}$$

$$\leq \begin{cases} \max(a_1, a_2) + \max(b_1, b_2) & \text{if } a_1 + b_1 \geq a_2 + b_2 \\ \max(a_1 a_2) + \max(b_1 b_2) & \text{otherwise} \end{cases}$$

$$= \max(a_1, a_2) + \max(b_1, b_2)$$

Thus $q(1)$ and $q(2)$ are true

Inductive Step

For $n = k$,

$$\max(a_1 + b_1, a_2 + b_2, \ldots, a_k + b_k) \leq \max(a_1, a_2, \ldots a_k) + \max(b_1, b_2, \ldots b_k) \quad \ldots (2)$$

For $n = k + 1$,

Then,

$$\max a_1 + b_1, a_2 + b_2 \ldots a_k + b_k, (a_{k+1} + b_{k+1}) = \max(\max(a_1 + b_1, a_2 + b_2, \ldots a_k + b_k), a_{k+1} + b_{k+1})$$

[$\because$ From recursive definitions]

$$\leq \max(\max a_1, a_2, \ldots a_k) + \min(b_1, b_2 \ldots b_k), a_{k+1} + b_{k+1})$$

[$\because$ From equation (2)]

$$\leq \max(\max(a_1, a_2, \ldots a_k), a_{k+1}) + \max(\max(b_1, b_2 \ldots b_k), b_{k+1})$$

[$\because$ $q(2)$ is true]

$$= \max(a_1, a_2 + b_2, \ldots a_k, a_{k+1}) + \max(b_1, b_2 \ldots b_k, b_{k+1})$$

$\therefore$ $q(k + 1)$ is true.

Hence, $q(n)$ is proved.

(c) Let, $q(n)$ be $\min(a_1 + b_1, a_2 + b_2, \ldots a_n + b_n) \geq \min(a_1, a_2 \ldots a_n) + \min(b_1, b_2 \ldots b_n)$

Basis Step

For $n = 1$ and $n = 2$,

$$\min(a_1 + b_1) = a_1 + b_1 = \min(a_1) + \min(b_1)$$

$$\min(a_1 + b_1, a_2 + b_2) = \begin{cases} a_1 + b_1 & \text{if } a_1 + b_1 \geq a_2 + b_2 \\ a_2 + b_2 & \text{Otherwise} \end{cases}$$

$$\geq \begin{cases} \min(a_1, a_2) + \min(b_1, b_2) & \text{if } a_1 + b_1 \geq a_2 + b_2 \\ \max(a_1, a_2) + \max(b_1, b_2) & \text{Otherwise} \end{cases}$$

$$= \min(a_1, a_2) + \min(b_1, b_2)$$

Thus $q(1)$ and $q(2)$ are true

Inductive Step

Let $n = k$,

$$\Rightarrow \quad \min(a_1 + b_1, a_2 + b_2, \ldots a_k + b_k) \geq \min(a_1, a_2 \ldots a_k) + \min(b_1, b_2 \ldots b_k) \quad \ldots (3)$$

Let,

$n = k + 1$,

Then, $\min(a_1 + b_1, a_2 + b_2, \ldots a_k + b_k, a_{k+1} + b_{k+1}) = \min(\min(a_1 + b_1, a_2 + b_2, \ldots a_k + b_k), a_{k+1} + b_{k+1})$

[$\because$ From recursive definitions]

$$\geq \min(\min(a_1, a_2, \ldots a_k) + \min(b_1, b_2 \ldots b_k), a_{k+1} + b_{k+1})$$

[$\because$ From equation (3)]

$$\geq \min(\min(a_1, a_2, \ldots a_k), a_{k+1}) + \min(\min(b_1, b_2 \ldots b_k), b_{k+1})$$

[$\because$ $q(2)$ is true]

$$= \min(a_1, a_2, \ldots a_k, a_{k+1}) + \min(b_1, b_2 \ldots b_k, b_{k+1})$$

Thus $q(k + 1)$ is true.

$\therefore$ $q(n)$ is proved.

Q70. Show that $f_{n+1} f_{n-1} - f_n^2 = (-1)^n$ when n is a positive integer.

Answer :

Let, $q(n)$ be $f_{n+1} f_{n-1} - f_n^2 = (-1)^n$

Basis Step

For $n = 1$,

Considering L.H.S $= f_{1+1} f_{1-1} - f_1^2 = f_2 f_0 - f_1^2$ ... (1)

From fibonacci definition,

$f_0 = 0$, ... (2)

$f_1 = 0$,

$f_n = f_{n-1} + f_{n-2}$ ... (3)

From equations (2) and (3)

$$\text{L.H.S} = (f_{2-1} + f_{2-2}) f_0 - f_1^2$$
$$= (f_1 + f_0) f_0 - f_1^2$$
$$= (1 + 0)\, 0 - 1^2 - 1 = -1$$

Thus $q(1)$ is true

Inductive Step

For $n = k$, equation (1) becomes,

$$f_{k+1} f_{k-1} - f_k^2 = (-1)^k$$

For $n = k + 1$, L.H.S of equation (1) becomes, ... (4)

$$f_{(k+1)+1} + f_{(k+1)-1} - f_{k+1}^2 = f_{k+2} f_k - f_{k+1}^2$$
$$= (f_k + f_{k+1}) f_k - f_{k+1}^2 \quad [\because \text{From equation (3)}]$$
$$= f_k^2 + f_{k+1} f_k - f_{k+1}^2$$
$$= f_k^2 - f_{k+1} (f_{k+1} + f_k)$$
$$= f_k^2 - f_{k+1} f_{k-1} \quad [\because \text{From equation (3)}]$$
$$= -(f_{k+1} f_{k-1} - f_k^2)$$
$$= -(-1)^k \quad [\because \text{From equation (4)}]$$
$$= (-1)(-1)^k$$
$$= (-1)^{k+1}$$

Thus, $q(k + 1)$ is true.

$\therefore$ By the principle of mathematical induction, $f_{n+1} f_{n-1} - f_n^2 = (-1)^n$, whenever n is a positive integer.

Q71. Consider an inductive definition of a version of Ackermann's function. This function was named after Wilhelm Ackermann, a German mathematician who was a student of the great mathematician David Hilbert. Ackermann's function plays an important role in the theory of recursive functions and in the study of the complexity of certain algorithms involving set unions. (There are several different variants of this function All are called Ackermann's function and have similar properties even though their values do not always agree).

$$A(m, n) = \begin{cases} 2n & \text{if } m = 0 \\ 0 & \text{if } m \geq 1 \text{ and } n = 0 \\ 2 & \text{if } m \geq 1 \text{ and } n = 1 \\ A(m-1, A(m, n-1)) & \text{if } m \geq 1 \text{ and } n \geq 2 \end{cases}$$

Find A(3, 4).

Answer :

Given,

$$A(m, n) = \begin{cases} 2n; & \text{if } m = 0 \\ 0; & \text{if } m \geq 1 \text{ and } n = 0 \\ 2; & \text{if } m \geq 1 \text{ and } n = 1 \\ A(m-1, A(m, n, -1)); & \text{if } m \geq 1 \text{ and } n \geq 2 \end{cases} \quad \ldots (1)$$

And $A(1, n) = 2n$

For $A(3, 4)$,

$3 \geq 1$ and $4 \geq 2$

Then,

$A(3, 4) = A(3 - 1, A(3, 4 - 1))$

$= A(2, A(3, 3))$

$= A(2, A(3 - 1, A(3, 3 - 1)))$

$= A(2, A(2, A(3, 2)))$

$= A(2, A(2, A(3 - 1, A(3, 2 - 1))))$

$= A(2, A(2, A(2, A(3, 1))))$

$= A(2, A(2, A(2, 2)))$

[$\because$ From equation (1)]

$= A(2, A(2, A(2 - 1, A(2, 2 - 1))))$

$= A(2, A(2, A(1, A(2, 1))))$

$= A(2, A(2, A(1, 2)))$

$= A(2, A(2, 2^2))$ [$\because A(1, n) = 2^n$]

$= A(2, A(2, 4))$

$= A(2, A(1, A(2, 3))$

$= A(2, A(1, A(1, A(2, 2)))$

$= A(2, A(1, 2^{A(2,2)}))$

$= A(2, A(1, 2^{A(1,A(2,1))}))$

$= A(2, A(1, 2^{A(1,2)}))$

$= A(2, A(1, 2^2))$

$= A(2, A(1, 16))$

$= A(2, 2^{16})$

$= A(2, 65536)$

$= A(2 - 1, A(2,65536 - 1))$

$= A(1, A(2,65535))$

$= 2^{A(2,65535)}$

$= 2^{A(1, A(2,65534))}$

$= 2^{2^{A(2,65534)}}$

$= 2^{2^{A(1,A(2,65533))}}$

$= 2^{2^{2^{A(2,65533)}}}$

$\vdots$

$= 2^{2^{2^{2^{\cdots 2^2}}}}$

$\therefore \quad A(3,4) = 2^{2^{2^{2^{\cdots 2^2}}}}$

Q72. Let a and b be positive integers with a ≥ b. Then the number of divisions used by the Euclidean algorithm to find gcd(a, b) is less than or equal to five times the number of decimal digits in b.

Answer : Model Paper-3, Q7(b)

According to Euclidean algorithm of $gcd\ (a, b)$ with $a \geq b$, (where $a = r_n$ and $b = r_1$) the sequence of equations are,

$$r_0 = r_1 q_1 + r_2 \quad ; \quad 0 \leq r_2 < r_1$$

$$r_1 = r_2 q_2 + r_3 \quad ; \quad 0 \leq r_3 < r_2$$

$$r_{n-2} = r_{n-1} q_{n-1} + r_n \quad ; \quad 0 \leq r_n < r_{n-1}$$

$$r_{n-1} = r_n q_n$$

Here, quotients $q_1, q_2, \ldots\ldots q_{n-1}$ are all at least 1.

Moreover, $q_n \geq 2$, because $r_n < r_{n-1}$.

i.e., $r_n \geq 1 = f_2$

$r_{n-1} \geq 2r_n \geq 2f_n = f_3$

$r_{n-2} \geq r_{n-1} + r_n \geq f_3 + f_2 = f_4$

$\vdots$

$r_2 \geq r_3 + r_4 \geq f_{n-1} + f_{n-2} = f_n$

$b = r_1 \geq r_2 + r_3 \geq f_n + f_{n-1} = f_{n+1}$

Thus, for n divisions in the Euclidean algorithm to obtain $gcd\ (a, b)$, $b \geq f_{n+1}$

Since, $b > \alpha^{n-1}$,

$$\log_{10} b > (n-1) \log_{10} \alpha > \frac{(n-1)}{5}$$

$$\left[\because \log_{10} \alpha \approx 0.208 > \frac{1}{5}\right]$$

Hence, $n - 1 < 5 \cdot \log_{10} b$

or $n - 1 < 5k$ and $n \leq 5k$

$\therefore$ $O(\log b)$ divisions are used by the Euclidean algorithm to find $gcd\ (a, b)$ whenever $a > b$.

3.2.4 Recursive Agorithms

Q73. Prove if T is a full binary tree, then $n(T) \leq 2^{h(T)+1} - 1$.

Answer : Model Paper-4, Q7(b)

Given, T is a full binary tree.

Basis step

For the full binary tree with root r, the result is true.

Since, $n(T) = 1$ and $h(T) = 0$

i.e., $n(T) = 1 \leq 2^{0+1} - 1 = 1$

Recursive step

Let $T = T_1 \Rightarrow n(T_1) \leq 2^{h(T_1)+1} - 1$ and $T = T_2 \Rightarrow n(T_2) \leq 2^{h(T_2)+1} - 1$

From recursive formulae,

$n(T) = 1 + n(T_1) + n(T_2)$ and $h(T) = 1 + \max(h(T_1), h(T_2))$

i.e., $n(T) = 1 + n(T_1) + n(T_2)$

$\leq 1 + (2^{h(T_1)+1} - 1) \cdot (2^{h(T_2)+1} - 1)$

$\leq 2 \cdot \max(2^{h(T_1)+1} \cdot 2^{h(T_2)+1}) - 1$

$= 2 \cdot 2^{\max(h(T_1), h(T_2))+1} - 1$

$= 2 \cdot 2^{h(T)} - 1$

$= 2^{h(T)+1} - 1$

Thus, $n(T) \leq 2^{h(T)+1} - 1$

3.2.5 Program Correctness

Q74. What is program correctness ?

Answer :

A program is said to be correct if, it takes all legal input and produces a correct result as an output without any computation and execution error. Partial correctners of a program can be proved only if the program terminates correctly.

In initial insertion part (i) the properties of input values are specified while, final insertion part (f) specifies the output properties of the program.

A program or program segment S is said to be partially correct with respect to 'i' and 'f'. If 'i' is true, S terminates making q as true.

- According to composition rule, if p is true and $S = S_1 ; S_2$ is executed and terminates, then r is true.

i.e.
$$\frac{p\{S_1\}q \quad q\{S_2\}r}{\therefore \; p\{S_1 ; S_2\}r}$$

- For conditional statements, rule of interference is,

$$\frac{(p \wedge \text{condition})\ \{S_1\}q \quad (p \wedge \neg \text{condition})\ \{S_2\} \rightarrow q}{\therefore \; p\{\text{if condition then } S_1 \text{ else } S_2\}q}$$

- For loop invariants (assertion remains true for each execution of S), the rule of interference is,

$$\frac{(p \wedge \text{condition})\ \{S\}p}{\therefore \; p\{\text{while condition } S\}\ (\neg \text{condition} \wedge p)}$$

Q75. Prove that the iterative program for finding f_n given in program is correct.

```
    If n = 0 then
    return 0
else
    x := 0
    y := 1
for  i := 1 to n – 1
    z := x + y
    x := y
    y := z
return y
```

Model Paper-5, Q7(b)

Answer :

Given program segment,

```
If n = 0 then
    return 0
else
    x := 0
    y := 1
for  i := 1 to n − 1
    z := x + y
    x := y
    y := z
return y
```

Proof

Let $q(n)$ be the statement "returned statement for n is f_n".

Basis Step

When $n = 0$, then 0 is returned.

i.e., returned statement for 0 is f_0

When $n = 1$, then $y = 1$ is returned.

returned statement for 1 is f_1

$\therefore$ $q(0)$ and $q(1)$ is true.

Induction Step

For $n = k - 1$,

returned statement for $k - 1$ is f_{k-1}

For $n = k$,

returned statement for k is f_k

Then, for $n = k + 1$,

$z = f_{k-1} + f_k = f_{k+1}$ [$\because$ From fibonacci numbers]

Then x is assigned to y,

$y = f_k$

And x is assigned to y,

$y = f_{k+1}$

Then returned statement for $k + 1$ is f_{k+1}

$\therefore$ $q(k + 1)$ is true.

Hence, $q(n)$ is true for all nonnegative integers n.

Q76. This program computes quotients and reminders

```
r := a
q := 0
while r ≥ d
    r := r − d
    q := q + 1
```

Verify that it is partially correct with respect to the initial assertion "a and d are positive intergers" and the final assertion "q and r are integers such that a = dq + r and 0 ≤ r < d".

Answer :

Given program;

$r := a$

$q := 0$

while $r \geq d$

$r := r - d$

$q := q + 1$

Loop Invariant

Let m be the statement "$a = dq + r$ and $r \geq 0$".

At the beginning of the while-loop

$$a = dq + r$$

$$r \geq d$$

$$r_{new} = r - d \geq d - d = 0$$

But,

$$q_{new} = q + 1$$

Then, $a = dq + r = dq + d - d + r = d(q + 1) + (r - d) = dq_{new} + r_{new}$

At the end of the while-loop, m is true.

$\therefore$ The program is partially correct with respect to the initial assertion "a and b are positive integers" and the final assertion "q and r are integers such that $a = dq + r$ and $0 \leq r < d$".

EXERCISE QUESTIONS

1. **Find the least integer n such that f(x) is $O(\chi^n)$ of the given function $f(x) = (x^4 + x^2 + 1)$ of given function $f(x) = (x^4 + x^2 + 1)(x^3 + x^2 + 1)$**

   Ans : : $n = 1, c = 2, k = 1$

2. **How much time does an algorithm using 2^{50} bit operations need if each bit operation takes these amount of 10^{-12} second?**

   Ans : 19 min

3. **Evaluate –221 mod 23.**

   Ans : 9

4. **Find the prime factorization of 729**

   Ans : 3

5. **Convert ABBA from its hexadecimal expansion to its binary expansion**

   Ans : 1010 10111011 1010

6. **Find an inverse of 19 modulo 141**

   Ans : 52