HSC SECOND YEAR – COMPUTER SCIENCE

CHAPTER 4: Algorithmic Strategies

Prepared by,

J. KAVITHA, B.Sc,B.Ed,M.C.A,M.Phil.,

Computer Instructor Gr - I,

GHSS, S.S.KULAM,

Coimbatore.

https://www.kavikalvi.freeweb.co.in/

Learning Objectives

The students will be able to:

- **Know the basics and technical perspective of algorithms.**
- **Understand the efficiency, time and space complexity of an algorithm.**
- **Develop and analyze algorithms for searching and sorting.**
- **Learn about dynamic programming through algorithmic approach.**

Introduction to Algorithmic strategies

- **An algorithm is a finite set of instructions to accomplish a particular task.**
- **It is a step-by-step procedure for solving a given problem.**
- **An algorithm can be implemented in any suitable programming language.**

Important Activities

Search	**To search an item in a data structure using linear and binary search.**
Sort	**To sort items in a certain order using the methods such as bubble sort, insertion sort, selection sort, etc.**
Insert	**To insert an item (s) in a data structure.**
Update	**To update an existing item (s) in a data structure.**
Delete	**To delete an existing item (s) in a data structure.**

Characteristics of an algorithm

Input	Zero or more quantities to be supplied.
Output	At least one quantity is produced.
Finiteness	Algorithms must terminate after finite number of steps.
Definiteness	All operations should be well defined.
Effectiveness	Every instruction must be carried out effectively.
Correctness	The algorithms should be error free.
Simplicity	Easy to implement.
Unambiguous	Algorithm should be clear and unambiguous.
Feasibility	Should be feasible with the available resources.
Portable	An algorithm should be generic, independent of any programming language or an operating system able to handle all range of inputs.
Independent	An algorithm should have step-by-step directions, which should be independent of any programming code.

Analysis of Algorithm

- Computer resources are limited.
- Efficiency of an algorithm is defined by the utilization of time and space complexity.
- Analysis of algorithms and performance evaluation can be divided into two different phases:
  - A Priori estimates: This is a theoretical performance analysis of an algorithm. Efficiency of an algorithm is measured by assuming the external factors.
  - A Posteriori testing: This is called performance measurement. In this analysis, actual statistics like running time and required for the algorithm executions are collected.

Complexity of an Algorithm

- The two main factors, which decide the efficiency of an algorithm are,
  - Time Factor: Time is measured by counting the number of key operations like comparisons in the sorting algorithm.
  - Space Factor: Space is measured by the maximum memory space required by the algorithm.

Complexity of an Algorithm

- The complexity of an algorithm f(n) gives the running time and/or the storage space required by the algorithm in terms of n as the size of input data.

Types of complexity:

1. Time Complexity: The Time complexity of an algorithm is given by the number of steps taken by the algorithm to complete the process.
2. Space Complexity: Space complexity of an algorithm is the amount of memory required to run to its completion

Efficiency of an algorithm

- The efficiency of an algorithm is defined as the number of computational resources used by the algorithm.
- An algorithm must be analyzed to determine its resource usage.
- The efficiency of an algorithm can be measured based on the usage of different resources.
- For maximum efficiency of algorithm we wish to minimize resource usage.
- The important resources such as time and space complexity cannot be compared directly, so time and space complexity could be considered for an algorithmic efficiency.

Method for determining Efficiency

- The efficiency of an algorithm depends on how efficiently it uses time and memory space.
- For example, write a program for a defined algorithm, execute it by using any programming language, and measure the total time it takes to run.
- The execution time that you measure in this case would depend on a number of factors such as:
  - Speed of the machine
  - Compiler and other system Software tools
  - Operating System
  - Programming language used
  - Volume of data required

Asymptotic Notations

- Asymptotic Notations are languages that use meaningful statements about time and space complexity.
    - Big O - Worst-case of an algorithm.
    - Big Ω - Best -case of an algorithm
    - Big Θ - complexity case of an algorithm (Or) lower bound = upper bound

Algorithm for Searching Techniques

Linear Search:

- **Linear search also called sequential search is a sequential method for finding a particular value in a list. This method checks the search element with each element in sequence until the desired element is found or the list is exhausted.**
- **In this searching algorithm, list need not be ordered.**

Binary Search:

- **Binary search also called half-interval search algorithm. It finds the position of a search element within a sorted array.**
- **The binary search algorithm can be done as divide-and-conquer search algorithm and executes in logarithmic time.**

Linear Search

Procedure:

- Traverse the array using for loop
- In every iteration, compare the target search key value with the current value of the list.
    - If the values match, display the current index and value of the array
    - If the values do not match, move on to the next array element.
- If no match is found, display the search element not found.

Linear Search

- Linear search will go step by step in a sequential order starting from the first element in the given array, if the search element is found that index is returned otherwise the search is continued till the last index of the array.

Example:

Input: values[] = {5, 34, 65, 12, 77, 35}

target = 77

Output: 4

Input: values[] = {101, 392, 1, 54, 32, 22, 90, 93}

target = 200

Output: -1 (not found)

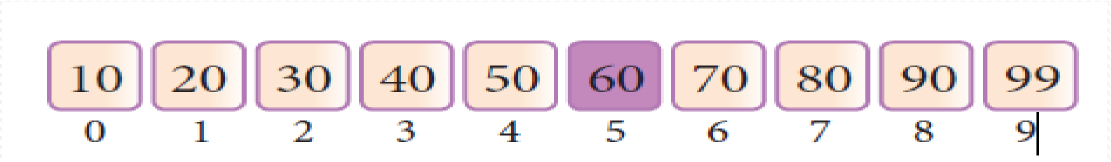
Binary Search

Procedure for Binary search:

1. **Start with the middle element:**
   - If the search element is equal to the middle element of the array, then return the index of the middle element.
   - If not, then compare the middle element with the search value,
   - If the search element is greater than the number in the middle index, then select the elements to the right side of the middle index, and go to Step-1.
   - If the search element is less than the number in the middle index, then select the elements to the left side of the middle index, and start with Step-1.
2. When a match is found, display success message with the index of the element matched.
3. If no match is found for all comparisons, then display unsuccessful message.

Binary Search - EXAMPLE

Let us assume that the search element is 60 and we need to search the index of search element 60 using binary search.



- First, we find index of middle element by using this formula :

  mid = (low + high) / 2 , Here it is, (0 + 9) / 2 = 4.

- Compare the value stored at index 4 with target value, which is not match with search element. As the search value 60 >50.
- Now we change our search range low to mid + 1 and find the new mid value as index 7.

Binary Search - EXAMPLE

- We compare the value stored at index 7 with our target value.
- Element not found because the value in index 7 is greater than search value. (80 > 60)
- Now we change our search range low to mid - 1 and find the new mid value as index 5.
- We compare the value stored at location 5 with our search element.
- We found that it is a match.
- We can conclude that the search element 60 is found at location or index 5.
- If no match is found for all comparisons, then return -1.

Sorting Techniques - Bubble sort algorithm

- Bubble sort is a simple sorting algorithm; it starts at the beginning of the list of values stored in an array.
- It compares each pair of adjacent elements and swaps them if they are in the unsorted order.
- This comparison and passed to be continued until no swaps are needed, which shows the values in an array is sorted.

Sorting Techniques - Bubble sort algorithm

Pseudo code:

- **Start with the first element i.e., index = 0, compare the current element with the next element of the array.**
- **If the current element is greater than the next element of the array, swap them.**
- **If the current element is less than the next or right side of the element, move to the next element.**
- **Go to Step 1 and repeat until end of the index is reached.**

Bubble sort algorithm

Example: Consider an array with values {15, 11, 16, 12, 14, 13}.

15 > 11	So Interchange	**15**	**11**	16	12	14	13
15 > 16	No Swapping	11	**15**	**16**	12	14	13
16 > 12	So Interchange	11	15	**16**	**12**	14	13
16 > 14	So Interchange	11	15	12	**16**	**14**	13
16 > 13	So Interchange	11	15	12	14	**16**	**13**
		11	15	12	14	13	16

Bubble sort algorithm

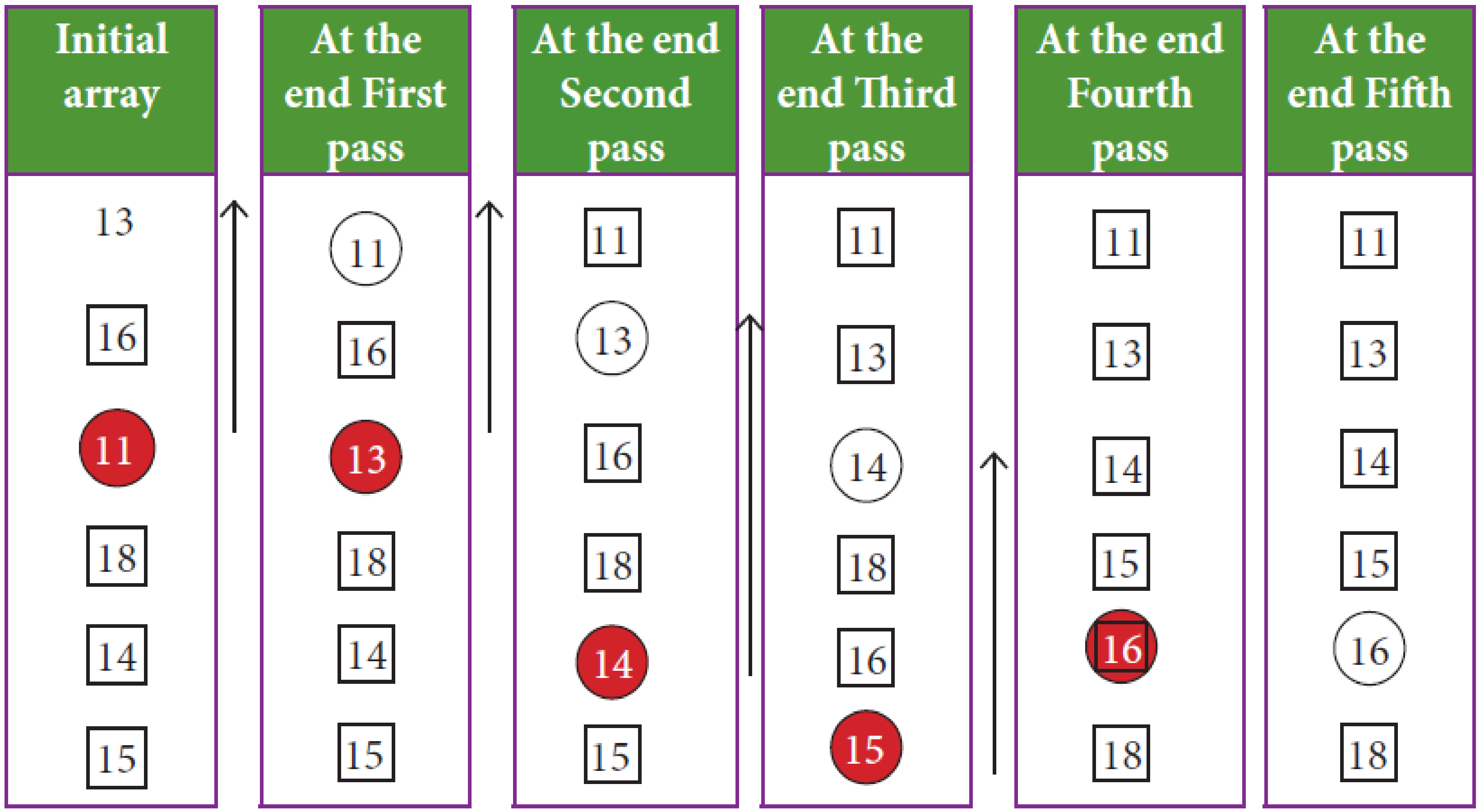
- The above pictorial example is for iteration-1.
- Similarly, remaining iteration can be done.
- At the end of all the iterations we will get the sorted values in an array as given below:

11	12	13	14	15	16

Selection sort

- The selection sort is a simple sorting algorithm that improves on the performance of bubble sort by making only one exchange for every pass through the list.
- This algorithm will first find the smallest elements in array and swap it with the element in the first position of an array, then it will find the second smallest element and swap that element with the element in the second position, and it will continue until the entire array is sorted in respective order.

Selection sort - Example



Initial array	At the end First pass	At the end Second pass	At the end Third pass	At the end Fourth pass	At the end Fifth pass
13	11	11	11	11	11
16	16	13	13	13	13
11	13	16	14	14	14
18	18	18	18	15	15
14	14	14	16	16	16
15	15	15	15	18	18

Insertion sort

- Insertion sort is a simple sorting algorithm.
- It works by taking elements from the list one by one and inserting then in their correct position in to a new sorted list.
- This algorithm builds the final sorted array at the end.
- This algorithm uses n-1 number of passes to get the final sorted list.

Insertion sort - Example

44	16	83	07	67	21	34	45	10	Assume 44 is a sorted list of 1 item
16	44	83	07	67	21	34	45	10	inserted 16
16	44	83	07	67	21	34	45	10	inserted 83
07	16	44	83	67	21	34	45	10	inserted 07
07	16	44	67	83	21	34	45	10	inserted 67
07	16	21	44	67	83	34	45	10	inserted 21
07	16	21	34	44	67	83	45	10	inserted 34
07	16	21	34	44	45	67	83	10	inserted 45
07	10	16	21	34	44	45	67	83	inserted 10

Dynamic programming

- Dynamic programming is used when the solution to a problem can be viewed as the result of a sequence of decisions.
- Dynamic programming approach is similar to divide and conquer (i.e) the problem can be divided into smaller sub-problems.
- Results of the sub-problems can be re-used to complete the process.
- Dynamic programming approaches are used to find the solution in optimized way.

Dynamic programming

Steps to do Dynamic programming:

- **The given problem will be divided into smaller overlapping sub-problems.**
- **An optimum solution for the given problem can be achieved by using result of smaller sub-problem.**
- **Dynamic algorithms uses Memoization.**

Example: Fibonacci Iterative Algorithm with Dynamic Programming Approach

Initialize f0=0, f1 =1

step-1: Print the initial values of Fibonacci f0 and f1

step-2: Calculate fibanocci fib ← f0 + f1

step-3: Assign f0← f1, f1← fib

step-4: Print the next consecutive value of fibanocci fib

step-5: Go to step-2 and repeat until the specified number of terms generated

For example if we generate fibonacci series up to 10 digits, the algorithm will generate the series as shown below:

- The Fibonacci series is : 0 1 1 2 3 5 8 13 21 34 55

Evaluation

1. **The word comes from the name of a Persian mathematician Abu Ja'far Mohammed ibn-i Musa al Khowarizmi is called?**

   (A) Flowchart (B) Flow

   <u>(C) Algorithm</u> (D) Syntax

2. **From the following sorting algorithms which algorithm needs the minimum number of swaps?**

   (A) Bubble sort (B) Quick sort

   (C) Merge sort <u>(D) Selection sort</u>

3. **Two main measures for the efficiency of an algorithm are**

   (A) Processor and memory (B) Complexity and capacity

   <u>(C) Time and space</u> (D) Data and space

4. **The algorithm that yields expected output for a valid input in called as**

   <u>(A) Algorithmic solution</u> (B) Algorithmic outcomes

   (C) Algorithmic problem (D) Algorithmic coding

Evaluation

5. Which of the following is used to describe the worst case of an algorithm?

   (A) Big A (B) Big S (C) Big W <u>(D) Big O</u>

6. Big Ω is the reverse of

   <u>(A) Big O</u> (B) Big θ (C) Big A (D) Big S

7. Binary search is also called as

   (A) Linear search (B) Sequential search

   (C) Random search <u>(D) Half-interval search</u>

8. The Θ notation in asymptotic evaluation represents

   (A) Base case <u>(B) Average case</u>

   (C) Worst case (D) NULL case

Evaluation

9. If a problem can be broken into subproblems which are reused several times, the problem possesses which property?

   (A) Overlapping subproblems (B) Optimal substructure

   (C) Memoization (D) Greedy

10. In dynamic programming, the technique of storing the previously calculated values is called?

   (A) Saving value property (B) Storing value property

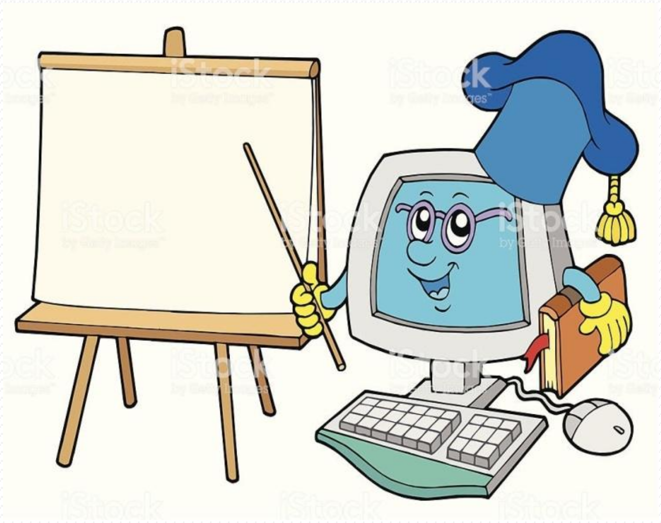
   (C) Memoization (D) Mapping

IMPORTANT QUESTIONS:

1. What is an Algorithm?
2. Write the phases of performance evaluation of an algorithm.
3. What is Insertion sort?
4. What is Sorting?
5. What is searching? Write its types.
6. List the characteristics of an algorithm.
7. Discuss about Algorithmic complexity and its types.
8. What are the factors that influence time and space complexity.
9. Write a note on Asymptotic notation.
10. What do you understand by Dynamic programming?

THANK YOU!!!

Knowledge is power. Information is liberating. Education is the premise of progress, in every society, in every family.



J. KAVITHA, B.Sc, B.Ed, M.C.A, M.Phil.,
Computer Instructor Gr - I
GHSS, S.S.KULAM
Coimbatore – 641107.