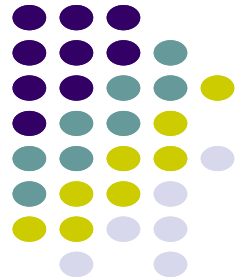
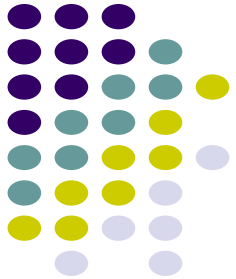
Searching and Sorting Algorithms

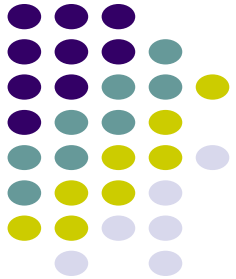


Searching and Sorting Algorithms



- Searching and sorting algorithms are fundamental tools in computer science and data processing.
- They enable efficient retrieval and organization of data, allowing for faster and more effective data manipulation

Searching Algorithms

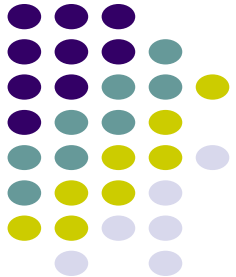


- An important feature of the system is the ability to efficiently search for data in complex data structures.
- It is possible to search for the required data in each data point, although this will not be the most effective approach.
- However, as the size of the data increases, we will require more complex algorithms to search the data.

Linear Search

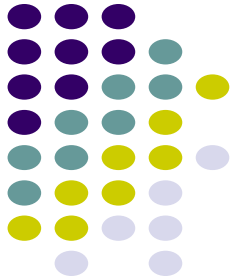
- Linear search is a sequential searching algorithm where we start from one end and check every element of the list until the desired element is found.
- A simple strategy for searching data involves looping through each element in search of the desired element.
- When a match is found between each data point, the results are returned, and the algorithm exits the loop.
- Otherwise, the algorithm will continue to search until it reaches the end of the data set.
- Due to the inherent exhaustive nature of linear search, it is obvious that linear search has a disadvantage of being very slow.

Binary Search



- Sorted data is a prerequisite for the binary search algorithm.
- A list is iteratively divided into two parts and the algorithm keeps track of the lowest and highest indices until it finds the value it requires:

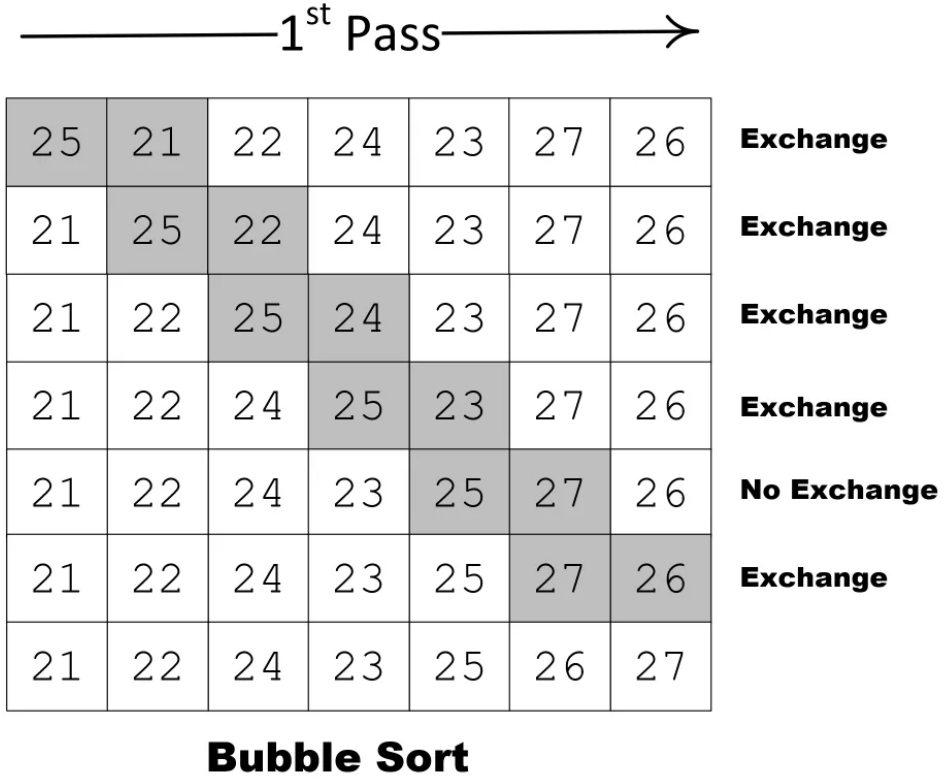
Sorting Algorithms



- Many modern algorithms require the ability to efficiently sort and search items in complex data structures.
- In order to achieve an efficient solution to a real-world problem, the right sorting and searching algorithm will be required.

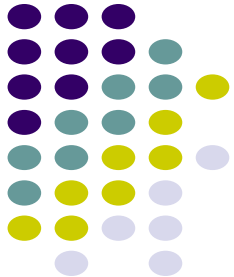
Bubble Sort Algorithm

- Bubble sort is a sorting algorithm that compares two adjacent elements and swaps them until they are in the intended order.
- Just like the movement of air bubbles in the water that rise up to the surface, each element of the array move to the end in each iteration. Therefore, it is called a bubble sort.
- In bubble sorting, adjacent neighbor values are compared. Values at higher positions are exchanged if their values are higher than those at lower positions.
- Iterations continue until the list reaches the end.



Bubble Sort

Selection Sort Algorithm

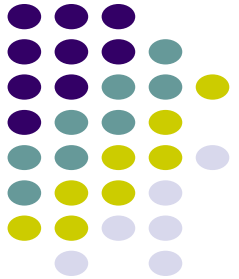


- Selection sort is a sorting algorithm that selects the smallest element from an unsorted list in each iteration and places that element at the beginning of the unsorted list.

Working of Selection Sort

- Set the first element as minimum
- Compare minimum with the second element. If the second element is smaller than minimum, assign the second element as minimum.
- Compare minimum with the third element. Again, if the third element is smaller, then assign minimum to the third element otherwise do nothing. The process goes on until the last element.
- After each iteration, minimum is placed in the front of the unsorted list.
- For each iteration, indexing starts from the first unsorted element. Step 1 to 3 are repeated until all the elements are placed at their correct positions.

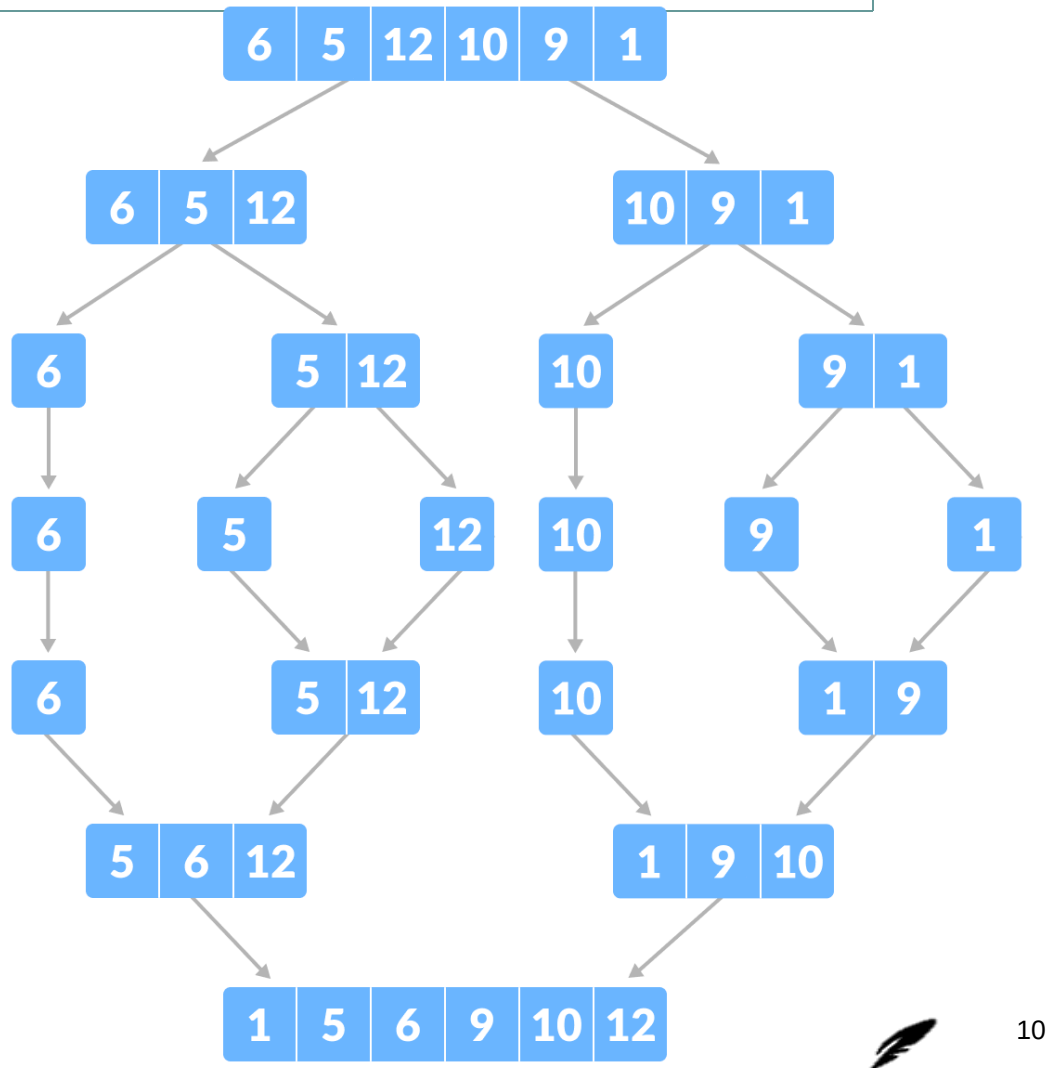
Insertion Sort Algorithm



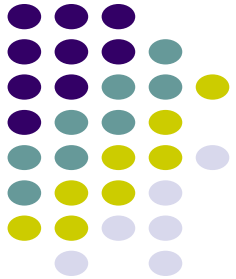
- Insertion sort is a sorting algorithm that places an unsorted element at its suitable place in each iteration.
- Insertion sort works similarly as we sort cards in our hand in a card game.
- We assume that the first card is already sorted then, we select an unsorted card.
- If the unsorted card is greater than the card in hand, it is placed on the right otherwise, to the left. In the same way, other unsorted cards are taken and put in their right place.

Merge Sort Algorithm

- Merge Sort is one of the most popular sorting algorithms that is based on the principle of Divide and Conquer Algorithm.
- Here, a problem is divided into multiple sub-problems.
- Each sub-problem is solved individually.
- Finally, sub-problems are combined to form the final solution.



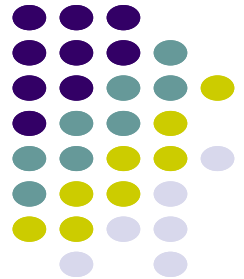
Quicksort Algorithm



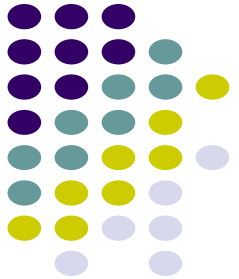
Quicksort is a sorting algorithm based on the divide and conquer approach where

- An array is divided into subarrays by selecting a pivot element (element selected from the array).
- While dividing the array, the pivot element should be positioned in such a way that elements less than pivot are kept on the left side and elements greater than pivot are on the right side of the pivot.
- The left and right subarrays are also divided using the same approach. This process continues until each subarray contains a single element.
- At this point, elements are already sorted. Finally, elements are combined to form a sorted array.

Greedy Algorithm

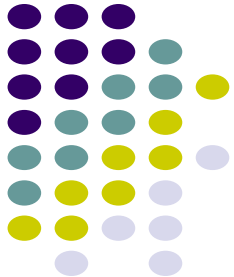


Greedy Algorithm



- A greedy algorithm is an approach for solving a problem by selecting the best option available at the moment.
- It doesn't worry whether the current best result will bring the overall optimal result.
- The algorithm never reverses the earlier decision even if the choice is wrong. It works in a top-down approach.
- This algorithm may not produce the best result for all the problems.
- It's because it always goes for the local best choice to produce the global best result.

Greedy Algorithm



We can determine if the algorithm can be used with any problem if the problem has the following properties:

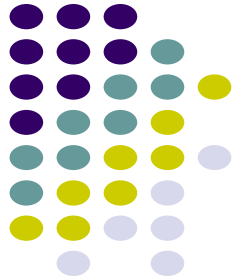
- **Greedy Choice Property**

If an optimal solution to the problem can be found by choosing the best choice at each step without reconsidering the previous steps once chosen, the problem can be solved using a greedy approach. This property is called greedy choice property.

- **Optimal Substructure**

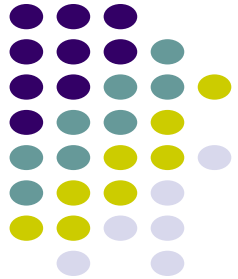
If the optimal overall solution to the problem corresponds to the optimal solution to its subproblems, then the problem can be solved using a greedy approach. This property is called optimal substructure.

Advantages of Greedy Approach



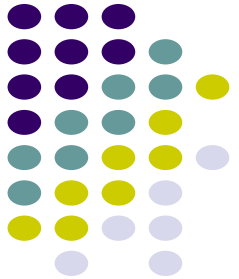
- The algorithm is easier to describe.
- This algorithm can perform better than other algorithms (but, not in all cases).

Drawback of Greedy Approach



- As mentioned earlier, the greedy algorithm doesn't always produce the optimal solution. This is the major disadvantage of the algorithm

Greedy Algorithm Examples



- Fractional Knapsack: Optimizes the value of items that can be fractionally included in a knapsack with limited capacity.
- Dijkstra’s algorithm: Finds the shortest path from a source vertex to all other vertices in a weighted graph.
- Kruskal’s algorithm: Finds the minimum spanning tree of a weighted graph.
- Huffman coding: Compresses data by assigning shorter codes to more frequent symbols.