Data Structures

BS (CS) _Fall_2024

Lab_04 Manual

Learning Objectives:

1. Sorting Algorithms
2. Searching Algorithms

Sorting Algorithms

What is the purpose of sorting?

What exactly is the purpose of a sorting algorithm? It is the job of a sorting algorithm to take an unordered list of numbers and, well, order them.

```
#pre-sorted   numbers = [5, 1, 4, 2, 3]
#post-sorted  numbers = [1, 2, 3, 4, 5]
```

Sorting an unordered list of numbers greatly increases the power you have over those numbers. Sorted lists allow you to better pick apart the information hiding inside the numbers.

There are a number of sorting algorithms. However, we will only cover two sorting algorithms.

1. Selection Sort
2. Bubble Sort

Selection Sort

In this, we divides the input array into a sorted part at the beginning and an unsorted part at the end. The algorithm repeatedly selects the smallest (or largest, depending on the order) element from the unsorted part and swaps it with the first unsorted element, gradually growing the sorted portion until the entire list is sorted.

Time Complexity: $O(n^2)$
Space Complexity: O(1)

Bubble Sort

In this, we start from the begnining of the array, swap the first two elements, if the first one is bigger than the second one. This process is continuously repeted until we reach the end of the array. In this way, the array elements are put in a sorted order as smaller elements bubble upto the beginning of the array.

Time Complexity: $O(n^2)$
Space Complexity: O(1)

Searching Algorithms

Operation of locating a specific data item in an array

- Successful: If location of the searched data is found
- Unsuccessful: Otherwise

Two algorithms for searching in arrays

- Linear search (or sequential search)
- Binary search

Linear Search

Algorithm works as follows:

- Starts from the first element of the array
- Uses a loop to sequentially step through an array
- Compares each element with the data item being searched
- Stops when data item is found or end of array is reached

Binary Search

Algorithm works as follows:

- Starts searching from the middle element of an array
- If value of data item is less than the value of middle element
    - Algorithm starts over searching the first half of the array
- If value of data item is greater than the value of middle element
    - Algorithm starts over searching the second half of the array
- Algorithm continues halving the array until data item is found

Lab Task

Question 1

Imagine you are tasked with sorting a list of student records in C++. Each student record consists of the following information:

- Student ID (an integer)
- Name (a string)
- GPA (a float)

You are required to use the insertion sort algorithm to sort these student records based on their GPAs in non-decreasing order. Write a C++ function that takes an array of student records and its size as input and sorts the records using insertion sort.

Write a C++ function selectionSortStudents that sorts an array of student records based on their GPAs using the insertion sort algorithm. The student records should be sorted in non-decreasing order of GPA. Each student record is represented as a struct with the following definition:

```
struct StudentRecord {
    int studentID;
    std::string name;
    float GPA;
};
```

The function should have the following signature:

```
void selectionSortStudents(StudentRecord arr[], int size)
```

Provide the implementation of the selectionSortStudents function and any additional helper functions you may need.

Question 2

Imagine you are managing a database of student grades, and you need to perform two key operations: sorting the grades using the bubble sort algorithm and searching for a specific student's grade using binary search. Each student is uniquely identified by their student ID, and their corresponding grade is represented as an integer.

Write a C++ program that first performs the following tasks:

Implement a function bubbleSort to sort an array of student grades in non-decreasing order using the bubble sort algorithm. The function should have the following signature:

void bubbleSort(int studentIDs[], int grades[], int numStudents)

Next, implement a function binarySearch to search for a specific student's grade based on their student ID. The function should perform a binary search on the sorted array of grades and return the grade if the student is found or -1 if the student ID is not in the database. The function should have the following signature:

int binarySearch(int studentIDs[], int grades[], int numStudents, int studentID)

Question 3

Eric loves puzzle making. One night he went to sleep after completing his puzzle only to wake up and find his puzzle had been messed up. Help Eric in **sorting** the puzzle pieces back to correct form.

- The first row of puzzle has to be sorted using **selection sort**
- The second row of puzzle has to be sorted using **bubble sort**

Prototype:

Unsorted Puzzle

32	30	39	34	37	35	33					
50	48	54	59	47	49	52	51	55	53	57	58

Sorted Puzzle

30	32	33	34	35	37	39					
47	48	49	50	51	52	53	54	55	57	58	59

Question 4

Write a C++ function to perform a linear search for a target string in a list of strings. The function should return the index of the target string if found, otherwise return -1.

Prototype: int linearSearchString(string arr[], int n, string target);