Lab 4

Arrays

Overview

- An array is a collection of items of same data type stored at contiguous memory locations. This makes it easier to calculate the position of each element by simply adding an offset to a base value, i.e., the memory location of the first element of the array (generally denoted by the name of the array)
- Types of Array:
  - Fixed sized
  - Dynamic sized
- Allocated in two types of memory :
  - Stack
  - Heap

Exercise 1: Process the Queries

- Problem Statement : You are given two arrays of Integers a and b, and list of queries, the elements of which are queries you are required to process. Every queries[i] can have one of the following forms:
  - [0,i,x] . In this case you need to assign a[i] the value of the x (a[i] = x)
  - [1,x]. In this case, you need to find the total number of pairs of indices i and j such that a[i] + b[i] = x

    Perform the given queries and return an array containing the results of the queries of the type [1,x].
- Input/Output Example
  - Input : a = {3,4} b = {1,2,3} queries = {{1,5}, {0,0,1}, {1,5}}
  - Output : results = {2,1}
  - Input : a = {2,3} b = {1,2,2} queries = {{1,4}, {0,0,3}, {1,5}}
  - Output: results = {3,4}

Algorithm

- Input : array a, array b and list of queries
- Output : list of results

1) start traversing the list of queries :
    A) if the length of the query is 2 :
        Check if the sum of a's element and b's element is target
        Increment the count for each pair
        Add the count in to the results list
    B) else:
        Update the value at the ith location of a array
2) return results

Exercise 2 : Find Max and Second Max from the array

- Problem Statement : Given an array arr[] of positive integers of size N that may contain duplicates. The goal is to discover the array's maximum and second maximum values, both of which must be distinct; if no second max exists, the second max will be -1.
- Require Input : int arr[] and int N
- Output : vector<int>
- Input and Expected Output Example :
  - **Input:**
  - N = 3
  - arr[] = {2,1,2}
  - **Output:** 2 1
  - N = 5
  - arr[] = {1,2,3,4,5}
  - **Output:** 5 4

Algorithm

1) Initialize the first as value of arr[0] and second as calue of -1
2) Start traversing the array from array[1],
   a) If the current element in array say arr[i] is greater
      than first. Then update first and second as,
      second = first
      first = arr[i]
   b) If the current element is in between first and second,
      then update second to store the value of current variable as
      second = arr[i]
3) Return the value stored in first and second.