ARRAY 2

Algorithm Problem Solving – Samsung Vietnam R&D Center

Compose by phuong.ndp

Agenda

- 2D Array
- Sequential Search
- Binary Search
- Selection Sort
- Problem Solving

2D Array

2D Array

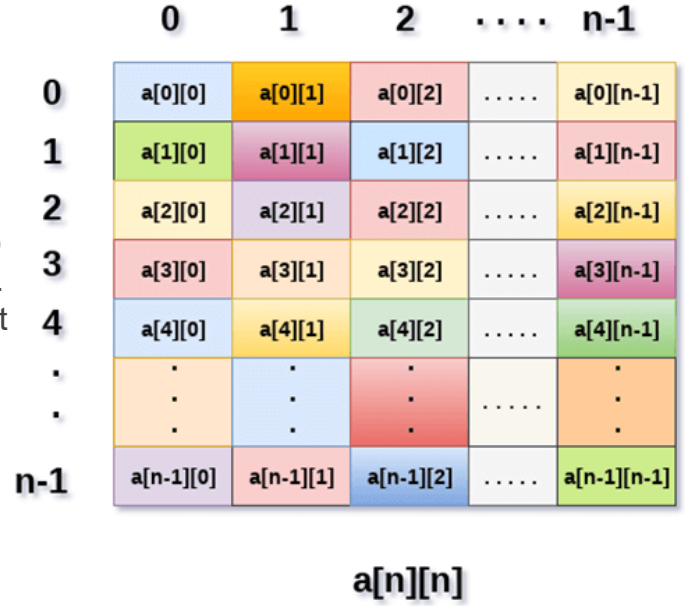
2D array can be defined as an array of arrays. The 2D array is organized as matrices which can be represented as the collection of rows and columns.

These arrays are preferably used if you want to put together data items in a table or matrix-like structure.

The syntax of declaring two dimensional array is very much similar to that of a one dimensional array, given as follo ws.

```
int arr[max_rows][max_columns];
```

In 2D arrays, all values must have the same data type. This means that yo u can't store an array of integers next to an array of strings and vice versa. For example, if one array is declared of type **int**, then its pointer can't point to the **string** type array.



2D Array

General form of declaring N-dimensional arrays:

```
data_type array_name[size1][size2]....[sizeN];
data_type: Type of data to be stored in the array. Here data_type
is valid C/C++ data type
array_name: Name of the array size1, size2,... ,sizeN: Sizes of
the dimensions
```

```
Example:
Two dimensional array: int two_d[10][20];
Three dimensional array: int three_d[10][20][30];
```

Total number of elements that can be stored in a multidimensional array can be calculated by multiplying the size of all the dimensions.

For example:

The array int x[10][20] can store total (10*20) = 200 elements.

Similarly array int x[5][10][20] can store total (5*10*20) = 1000 elements.

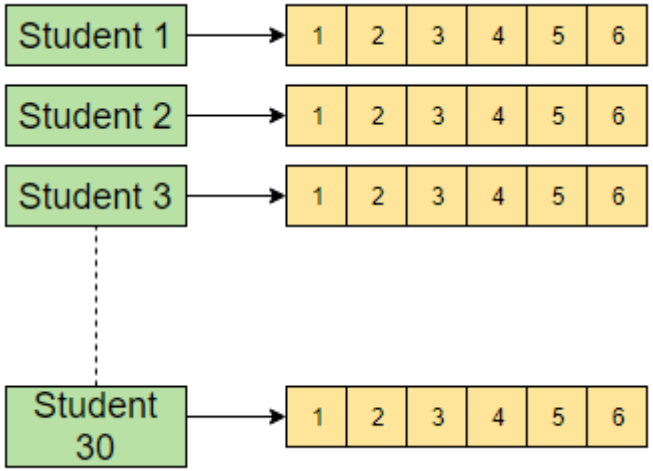
	Column 0	Column 1	Column 2
Row 0	x[0][0]	x[0][1]	x[0][2]
Row 1	x[1][0]	x[1][1]	x[1][2]
Row 2	x[2][0]	x[2][1]	x[2][2]

Example 1: Student's Records

Let's say there's a class of 30 students, and we want to store their grades for each of their courses (six courses in total). Is t here a way we can map all this information using the knowledge we have covered so far? Yes, we can!

Example 1: Student's Records

All we need to do is to keep the first column(i.e., Column 0), fixed for students, and the rest of the columns to store grades. We will have to follow a specific order to make sure the scores and students don't get mixed up. Each element of the first column will hold a reference to an array containing marks of the student. So, there will be 30 rows and 6 columns. Now we can calculate the student's percentage, class average, and the student's position.



Sequential Search

Sequential Search

The **sequential search** (sometimes called a **linear search**) is the simplest type of search, it is used when a list of integers is not in any order. It examines the first element in the list and then examines each "sequential" element in the list until a match is found. This match could be a desired work that you are searching for, or the minimum number in the list.

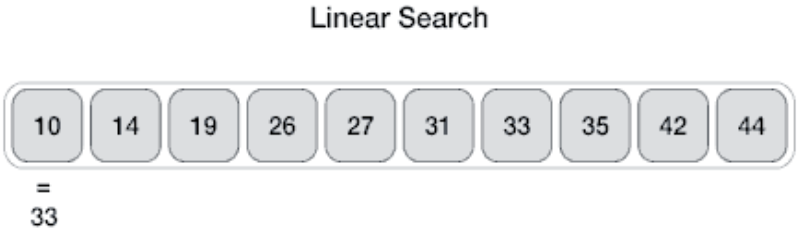
It is important to remember that the:

- maximum number of searches required is: n+1
- average number of searches required is: (n+1)/2

(n = the number of elements on the list)

A simple approach is to do linear search, i.e

- Start from the leftmost element of arr[] and one by one compare x with each element of arr[]
- If x matches with an element, return the index.
- If x doesn’t match with any of elements, return -1.



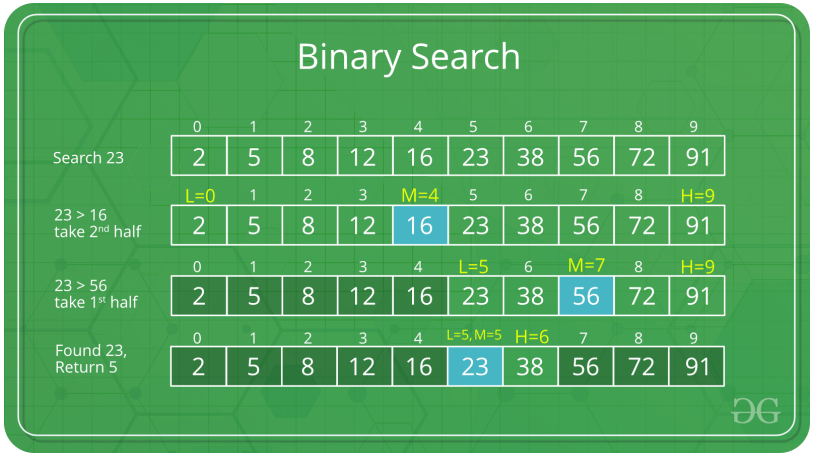
Binary Search

Binary Search

Binary search is a fast search algorithm with run-time complexity of O(log n). This search algorithm works on the principle of divide and conquer. For this algorithm to work properly, the data collection should be in the sorted form.

Binary search looks for a particular item by comparing the middle most item of the collection.

- If a match occurs, then the index of item is returned.
- If the middle item is greater than the item, then the item is searched in the sub-array to the left of the middle item.
- Otherwise, the item is searched for in the sub-array to the right of the middle item.
- This process continues on the sub-array as well until the size of the subarray reduces to zero.



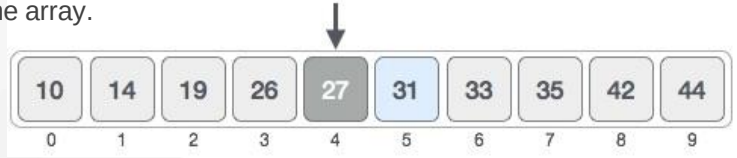
How Binary Search Works?

The following is our sorted array and let us assume that we need to search the location of value 31 using binary search.

First, we shall determine half of the array by using this formula

```
mid = low + (high - low) / 2
```

Here it is, 0 + (9 - 0) / 2 = 4 (integer value of 4.5). So, 4 is the mid of the array.



As the value is greater than 27 and we have a sorted array, so we also know that the target value must be in the upper portion of the array.

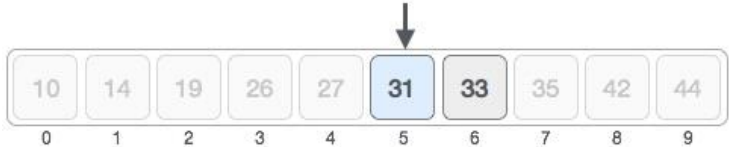
We change our low to mid + 1 and find the new mid value again.

```
low = mid + 1
mid = low + (high - low) / 2
```

Our new mid is 7 now. We compare the value stored at location 7 with our target value 31.

The value stored at location 7 is not a match, rather it is more than what we are looking for. So, the value must be in the lower part from this location.

Hence, we calculate the mid again. This time it is 5.



We compare the value stored at location 5 with our target value. We find that it is a match.

Binary Search Pseudocode

The pseudocode of binary search algorithms should look like this

```
Procedure binary_search
   A ← sorted array
   n ← size of array
   x ← value to be searched

   Set lowerBound = 1
   Set upperBound = n

   while x not found
      if upperBound < lowerBound
         EXIT: x does not exists.

      set midPoint = lowerBound + ( upperBound - lowerBound ) / 2

      if A[midPoint] < x
         set lowerBound = midPoint + 1

      if A[midPoint] > x
         set upperBound = midPoint - 1

      if A[midPoint] = x
         EXIT: x found at location midPoint
   end while

end procedure
```

Problem Solving

Given a sorted array **arr**[] of **N** integers and a number **K** is given. The task is to check if the element K is present in the array or not using binary search.

Input: First line of input contains number of testcases T. For each testcase, first line of input contains number of elements in th e array and the number K seperated by space. Next line contains N elements.

Output: For each testcase, if the element is present in the array print "1" (without quotes), else print "-1" (without quotes).
Constraints:
1 <= T <= 100
1 <= N <= 10^6
1 <= K <= 10^6
1 <= arr[i] <= 10^6

Input:
2
5 6
1 2 3 4 6
5 2
1 3 4 5 6

Output:
1
-1

Selection Sort

Selection Sort

- This sorting algorithm is an in-place comparison-based algorithm in which the list is divided into two parts, the sorted p art at the left end and the unsorted part at the right end. Initially, the sorted part is empty and the unsorted part is the e ntire list.

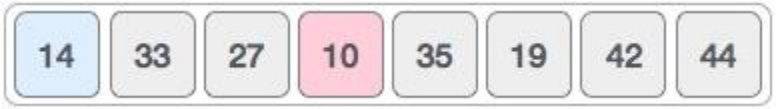
- The smallest element is selected from the unsorted array and swapped with the leftmost element, and that element be comes a part of the sorted array. This process continues moving unsorted array boundary by one element to the right.

- This algorithm is not suitable for large data sets as its average and worst case complexities are of $O(n^2)$, where **n** is the number of items.

How Selection Sort Works?

Consider the following depicted array as an example.

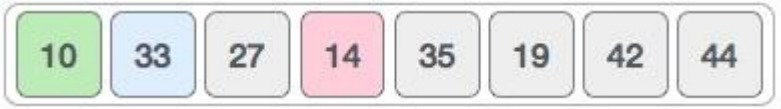
For the first position in the sorted list, the whole list is scanned se quentially. The first position where 14 is stored presently, we sear ch the whole list and find that 10 is the lowest value.



So we replace 14 with 10. After one iteration 10, which happens t o be the minimum value in the list, appears in the first position of t he sorted list.

For the second position, where 33 is residing, we start scanning t he rest of the list in a linear manner.

We find that 14 is the second lowest value in the list and it should appear at the second place. We swap these values.



After two iterations, two least values are positioned at the beginni ng in a sorted manner.

How Selection Sort Works?

The same process is applied to the rest of the items in the array.
Following is a pictorial depiction of the entire sorting process

Selection Search Pseudocode

The pseudocode of selection search algorithms should look like this

```
procedure selection sort
   list  : array of items
   n     : size of list

   for i = 1 to n - 1
   /* set current element as minimum*/
      min = i

      /* check the element to be minimum */

      for j = i+1 to n
         if list[j] < list[min] then
            min = j;
         end if
      end for

      /* swap the minimum element with the current element*/
      if indexMin != i  then
         swap list[min] and list[i]
      end if
   end for

end procedure
```

Problem Solving

The task is to complete select() function which is used to implement Selection Sort.
Input: First line of the input denotes number of test cases 'T'. First line of the test case is the size of array and second line consists of array elements.
Output: Sorted array in increasing order is displayed to the user.
Your Task : Complete the function *select()* that helps to sort the array.

Expected Time Complexity : O(n) **Expected Auxilliary Space** : O(1)
Constraints:
1 <=T<= 50
1 <=N<= 1000
1 <=arr[i]<= 1000

Input:
2
5
4 1 3 9 7
10
10 9 8 7 6 5 4 3 2 1

Output:
1 3 4 7 9
1 2 3 4 5 6 7 8 9 10

Sort Algorithm Comparison

Time and Space Complexity Comparison Table :

SORTING ALGORITHM	TIME COMPLEXITY			SPACE COMPLEXITY
	BEST CASE	AVERAGE CASE	WORST CASE	WORST CASE
Bubble Sort	Ω(N)	Θ(N^2)	O(N^2)	O(1)
Selection Sort	Ω(N^2)	Θ(N^2)	O(N^2)	O(1)
Insertion Sort	Ω(N)	Θ(N^2)	O(N^2)	O(1)
Merge Sort	Ω(N log N)	Θ(N log N)	O(N log N)	O(N)
Heap Sort	Ω(N log N)	Θ(N log N)	O(N log N)	O(1)
Quick Sort	Ω(N log N)	Θ(N log N)	O(N^2)	O(N log N)
Radix Sort	Ω(N k)	Θ(N k)	O(N k)	O(N + k)
Count Sort	Ω(N + k)	Θ(N + k)	O(N + k)	O(k)
Bucket Sort	Ω(N + k)	Θ(N + k)	O(N^2)	O(N)

Thank you!

Algorithm Problem Solving – Samsung Vietnam R&D Center

Created by phuong.ndp

Source

https://www.geeksforgeeks.org/multidimensional-arrays-c-cpp/
https://www.educative.io/courses/data-structures-coding-interviews-java/B8Q9XGrlLmJ
http://pkirs.utep.edu/cis3355/Tutorials/chapter9/9.00/seq_search.htm
https://www.tutorialspoint.com/data_structures_algorithms/binary_search_algorithm.htm
https://practice.geeksforgeeks.org/problems/who-will-win/0
https://www.tutorialspoint.com/data_structures_algorithms/selection_sort_algorithm.htm
https://www.geeksforgeeks.org/analysis-of-different-sorting-techniques/
https://practice.geeksforgeeks.org/problems/selection-sort/1