DAA Theory Project

latex practice

Fractional Cascading

-This is a sample subtitle -

By

March 11, 2021

Contents

1 Abstract

In this paper, we investigate the *Fractional Cascading* technique in building range trees and for fast searching of an element in multiple arrays. In this venture, we introduce and examine *Linear Search*, *Binary Search*, *Bridge Building* and *Auxiliary lists*. We also look at some of the *applications* of this technique, and suggest data structures for its efficient realisation.

2 Problem Statement

Fractional cascading: You are given an input of k ordered lists of numbers, each of size n as well as a query value x. The problem's output is to return, for each list, True if the query value appears in the list and False if it does not. For example, if the input is:

(a) List L_1: $[3, 4, 6]$

(b) List L_2: $[2, 6, 7]$

(c) List L_3: $[2, 4, 9]$

and the query value is 4, then the expected output is [*True*,*False*,*True*].
Give an algorithm to solve the fractional cascading problem.

3 Brute Force

3.1 Linear Search

Linear search is the most basic search technique, wherein we sequentially compare each array element to the target element. In the worst case of the target element not coinciding with *any* list element, the algorithm would reach the end of the list and we would report an unsuccessful search. As each element is compared at most once, the time complexity is $O(n)$, where n is the size of the list.

This algorithm forms the basis of the simplest solution to our problem: We just run linear search on each of the k lists. If we have q queries, this takes $O(q \cdot k \cdot n)$ time, which is a lot and in real world situations, if k, q, and n are even moderately large, the time taken would become astronomical.

```
Data: k arrays of size n and a query element x
Result: Boolean array regarding whether element is present in the
        indiced array or not
Function LinearSearch(Array,x):
    for $i \leftarrow 0$ to $n$ do
        if Array[i] == $x$ then
            return True;
        end
    end
return False
Function Main:
    output = [] ;
    for $i \leftarrow 0$ to $k$ do
        output.append(linearSearch(input[i],x);
    end
return 0
```

Note that, however, this approach does not take into account any relevant information given to us in the question which can speed up this algorithm. It is given that the *lists are sorted*, so we can exploit this property and employ a faster searching technique to solve this problem in a better way: *Binary Search*

4 Improved Brute Force

4.1 Binary Search

Binary search is another searching algorithm that works correctly only on sorted arrays.
It begins by comparing the target element with the element at the middle of the list.

- If they are equal, we have found the target in the list
- If the target is larger, and as the list is sorted, we must now turn our attention to the *right half* of the list
- Similarly, if the target is smaller, we must focus on the *left half* of the list.

In the worst case, Binary Search will take $O(\log n)$ comparisons, where n is the size of the list.

To improve the performance of the Brute Force subroutine, we replace the Linear Search subroutine with the aforementioned Binary Search subroutine. If we have q queries, this takes $O(q \cdot k \cdot \log n)$ time, which is certainly a lot better than the initial brute force algorithm, but can yet be improved further.

```
Data: k arrays of size n and a query element x
Result: Boolean array regarding whether element is present in the
        indiced array or not
Function BinarySearch(Array,x,left,right):
    if right ≥ left then
        mid=$\frac{(left+right)}{2}$;
        if Array[mid] == x then
            return True;
        end
        if Array[mid] > x then
            BinarySearch(Array,x,left,mid-1);
        end
        else
            BinarySearch(Array,x,mid+1,right);
        end
    end
    else
        return False;
    end
end Function
Function Main:
    output = [] ;
    for i ← 0 to k do
        output.append(BinarySearch(input[i],x);
    end
return 0
```

5

References

[1] Alcosser, Harvard. “Diamond Bar High School.”