Design and Analysis of Algorithm

Assignment 5

Group 9 :

- Aditya Aggarwal - IIT2019210
- Divy Agrawal - IIT2019211
- Aman Rubey - IIT2019212

Problem Statement

Check whether element X is present or not using Divide and conquer.
Show advantage against linear uni-processor search.

Divide and Conquer

It decomposes a given problem into small sub-problems recursively until it is relatively easier to solve those small sub-problems. Then the solution to the original problem is computed by combining the solutions to the sub-problems.

Basically , a divide and conquer algorithm consists of following three steps.

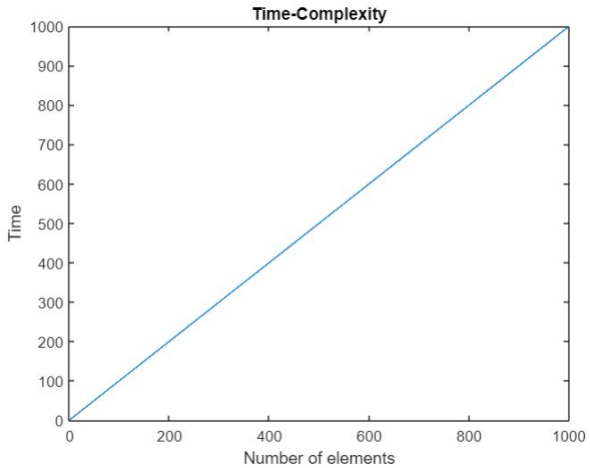
- **Divide:** Dividing the bigger problem into a set of smaller problems.
- **Conquer**: Solving every smaller problem individually, recursively.
- **Combine**: Merging the solutions of the smaller problems to get the solution to the whole bigger problem.

Algorithm

- Declare a vector v and store n integers into it. Now, we have to search an integer x in this vector.
- Now, call the solve function by passing starting and ending indexes of vector.
- We have to divide the bigger problem into a smaller problem, now assuming this small problem as a big problem we have to divide it until we achieve an endpoint.
- Now the endpoint in this scenario will be that when their is only 1 element in the vector, for that we will apply search in O(1) by just checking v[index]==x or not.
- Division will divide the vector in around 2 equal parts (left and right). For every part we will divide it again to left and right until both left and right becomes the same.
- Now, for our endpoint we know the result, we just have to conquer them to get the solution of the bigger problem that is, if we get our answer=True in any of the smaller problem, then the answer of the bigger problem will also be True otherwise answer will be False.

Time Complexity

The time complexity of this problem is O(N).



Suppose if the time complexity of the problem consisting of N elements is T(N) then the time complexity of the subproblems of this problem will be T(N/2) approximately and let time complexity for executing rest statements be C_1. Then,

$$\mathbf{T(N) = 2T(N/2)+C_1}$$

Which is equivalent to,

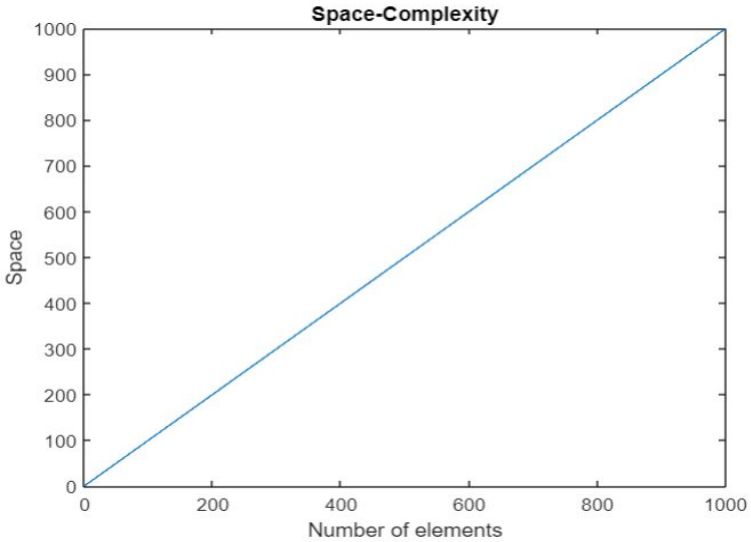
$$\mathbf{T(N) = 2T(N/2) + \Theta(1)}$$

Therefore upon applying master's theorem we get,

$$\mathbf{T(N) = O(N)}$$

Auxiliary Space Complexity

The Auxiliary space complexity is O(N).



Let N denote the number of elements in the vector in which we have to check whether the required element is present or not.

The function solve(low,high) is being called approximately 2N-1 times that is O(N) times and each time when it is called 3 variables of size 4 bytes is created. Therefore, Auxiliary Space complexity will be,

T(N) = O(N)

TEST CASE

INPUT 1:

4 999

34 4 56 999

INPUT 2:

5 780

123 364 273 782 781

OUTPUT 1:

999 is present

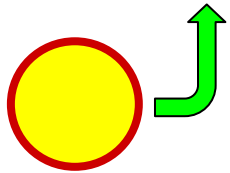
OUTPUT 2:

780 is not present

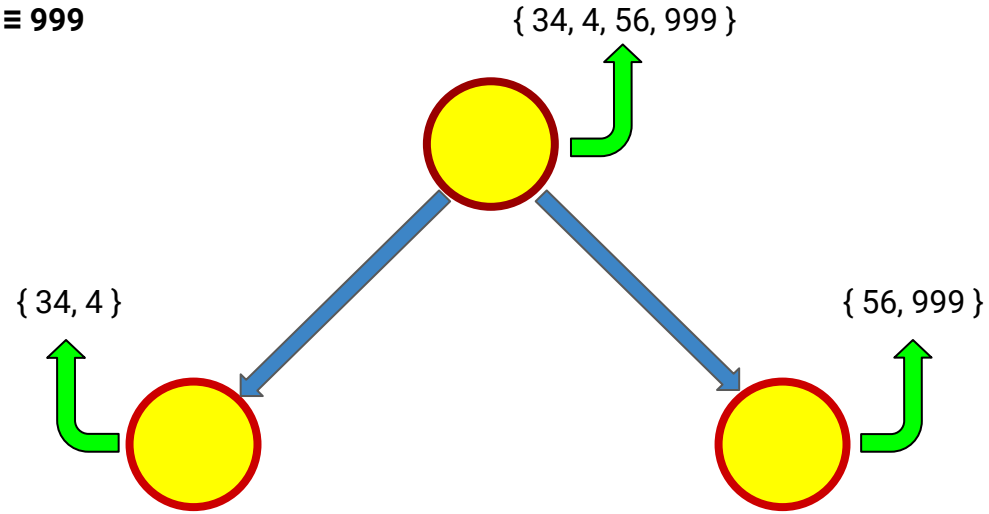
Example

Element to be searched ≡ 999

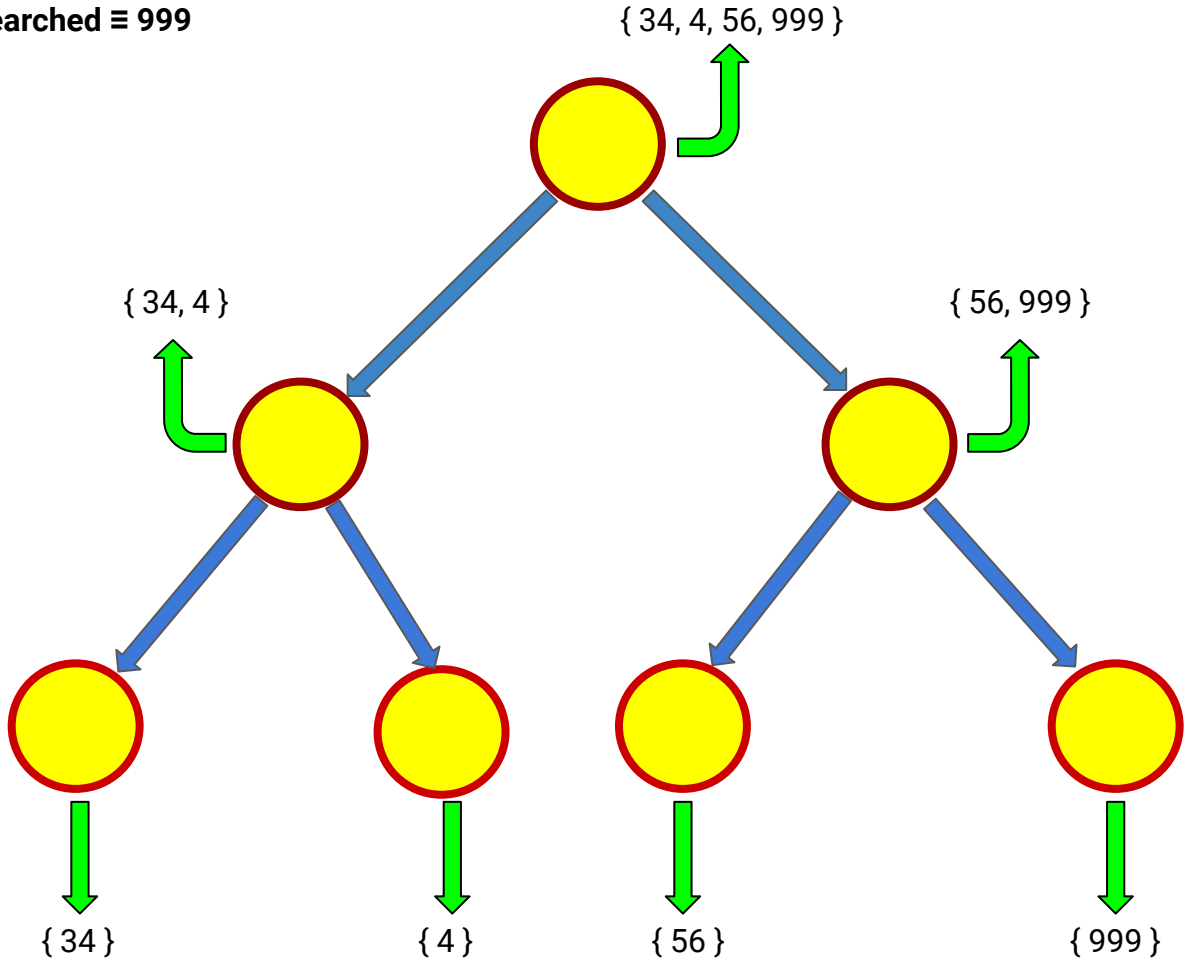
{ 34, 4, 56, 999 }



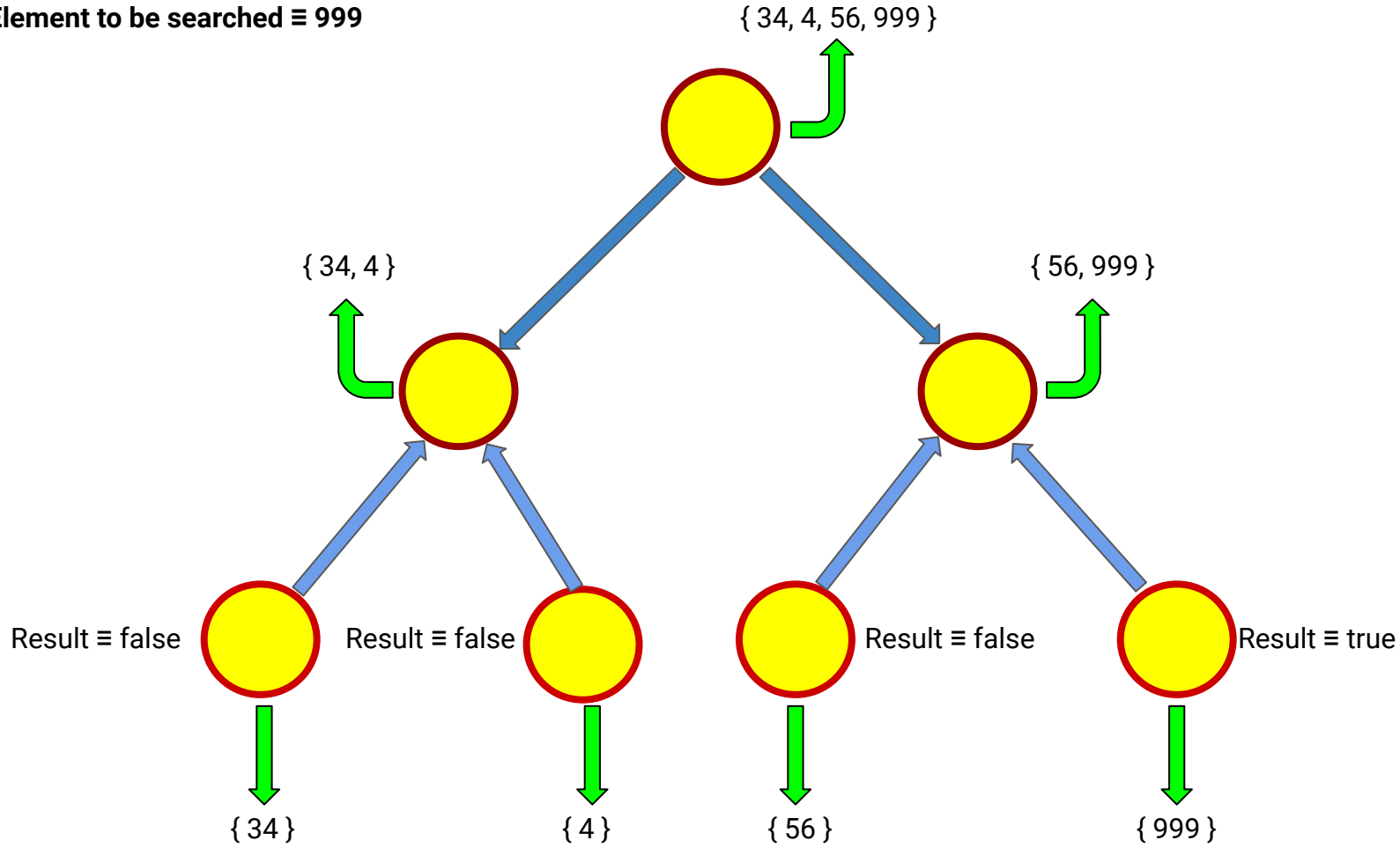
Element to be searched ≡ 999



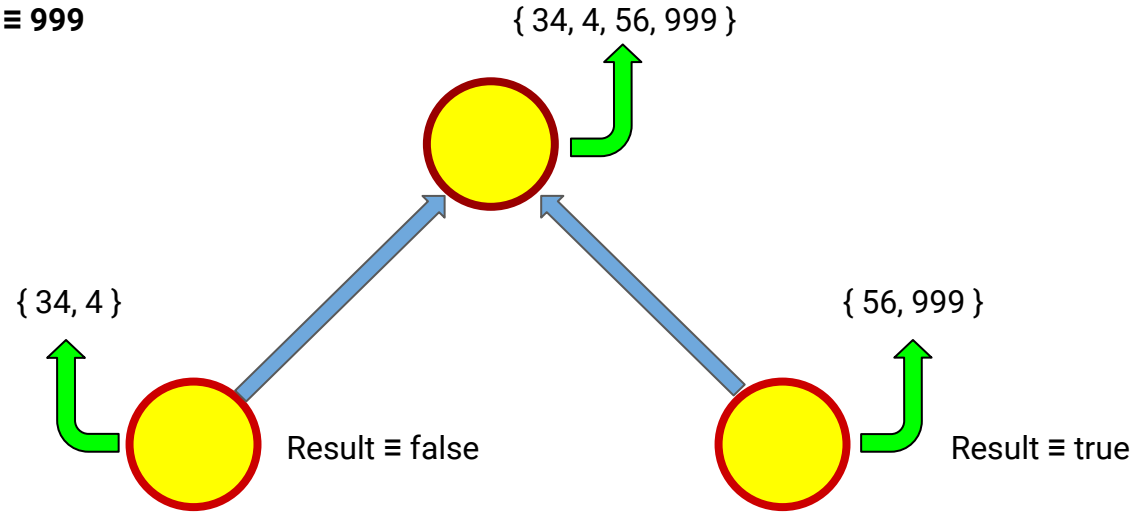
Element to be searched ≡ 999



Element to be searched ≡ 999



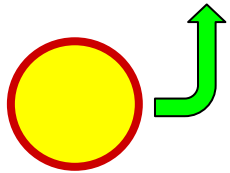
Element to be searched ≡ 999



Element to be searched ≡ 999

{ 34, 4, 56, 999 }

Result ≡ true



Advantage against Linear Uni-Processor Search

1. When the number of inputs becomes very large Divide and Conquer Algorithm executes faster than Linear Uni-processor Search despite of the fact that both the algorithms have same time complexity due to the implementation of multithreading by the OS at software level without kernel knowing about it. But, when the number of inputs are low the time of execution is same for both.

2. Another advantage of Divide and Conquer Algorithm is the use of memory caches efficiently as when the sub problems becomes very simple, they can be solved within the cache, without having to access the slower main memory(Cache is second fastest accessible memory after Registers), which saves a lot of time and makes the algorithm more efficient.

```
┌──(aditya㉿kali)-[~/…/DAA_PR/git/daa_assignment/Assignment-05]
└─$ g++ search_element_x.cpp -o search_element_x

┌──(aditya㉿kali)-[~/…/DAA_PR/git/daa_assignment/Assignment-05]
└─$ ./search_element_x
4 999
34 4 56 999
999 is present

┌──(aditya㉿kali)-[~/…/DAA_PR/git/daa_assignment/Assignment-05]
└─$ ./search_element_x
10 78023
479 98475 8475 9479 945797 945 9847 5947 554 80374
78023 is not present

┌──(aditya㉿kali)-[~/…/DAA_PR/git/daa_assignment/Assignment-05]
└─$ ./search_element_x
5
6
1 2 3 4 5
6 is not present

┌──(aditya㉿kali)-[~/…/DAA_PR/git/daa_assignment/Assignment-05]
└─$ _
```

THANKS