UNIVERSITY INSTITUTE OF COMPUTING

MASTER OF COMPUTER APPLICATIONS

DESIGN AND ANALYSIS OF ALGORITHMS

24CAT611

UNIT-2

DISCOVER . LEARN . EMPOWER

Outline

- **Divide and Conquer: General method**
- Binary Search
- Advantages and disadvantages of divide and conquer

Divide & Conquer

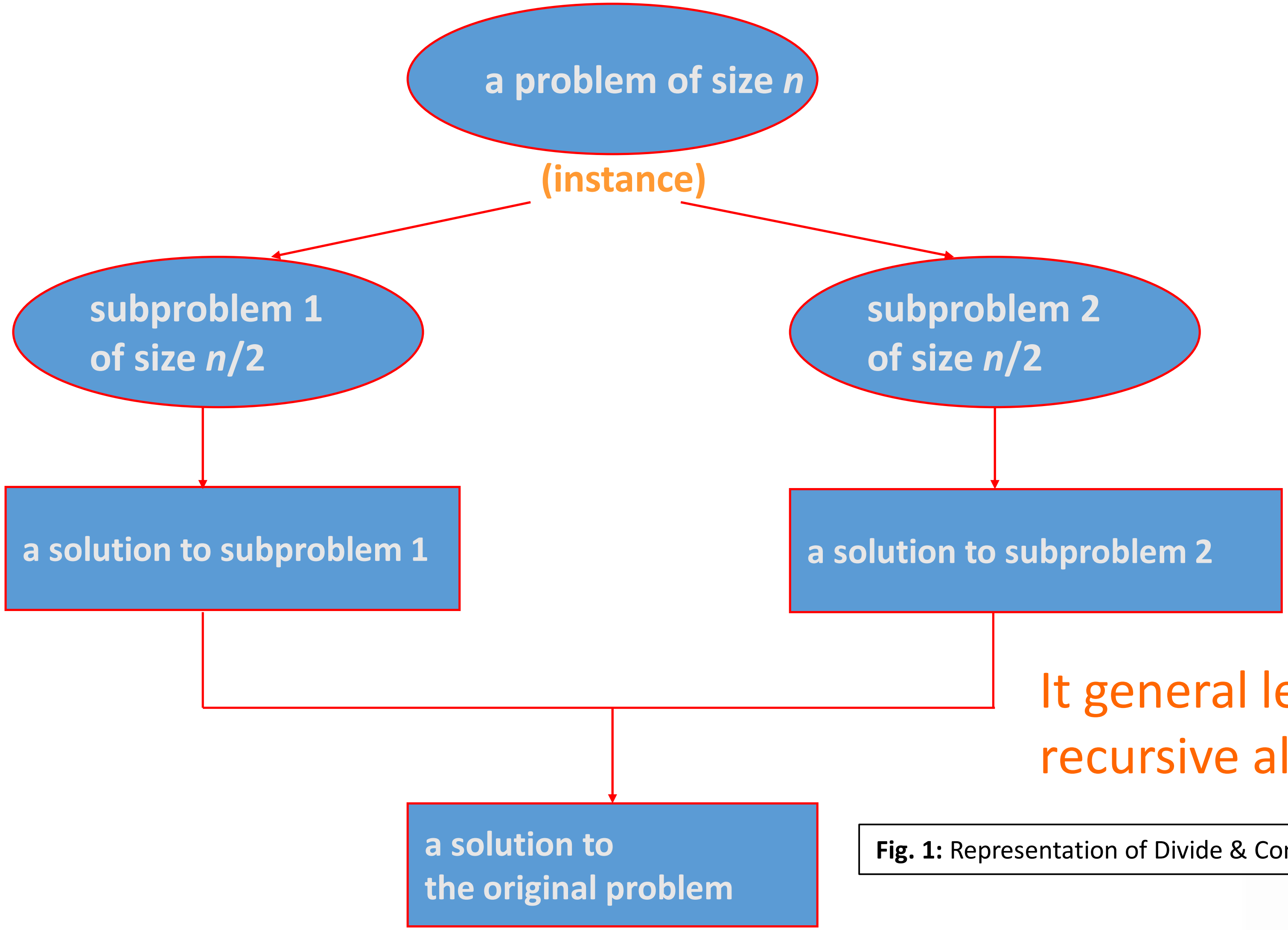
Divide and Conquer is an algorithm design paradigm that involves breaking up a larger problem into *non-overlapping* sub-problems, solving each of these sub-problems, and combining the results to solve the original problems. A problem has non-overlapping sub-problems if you can find its solution by solving each sub-problem once.

Divide & Conquer (contd.)

The three main steps in the divide and conquer paradigm are:

- **divide**: involves breaking the problem into smaller, non-overlapping chunks.
- **conquer**: involves solving the sub-problems recursively.
- **combine**: involves combining the solutions of the smaller sub-problems to solve the original problem.

Divide-and-Conquer Technique (cont.)



It general leads to a recursive algorithm!

Fig. 1: Representation of Divide & Conquer

Divide-and-Conquer Examples

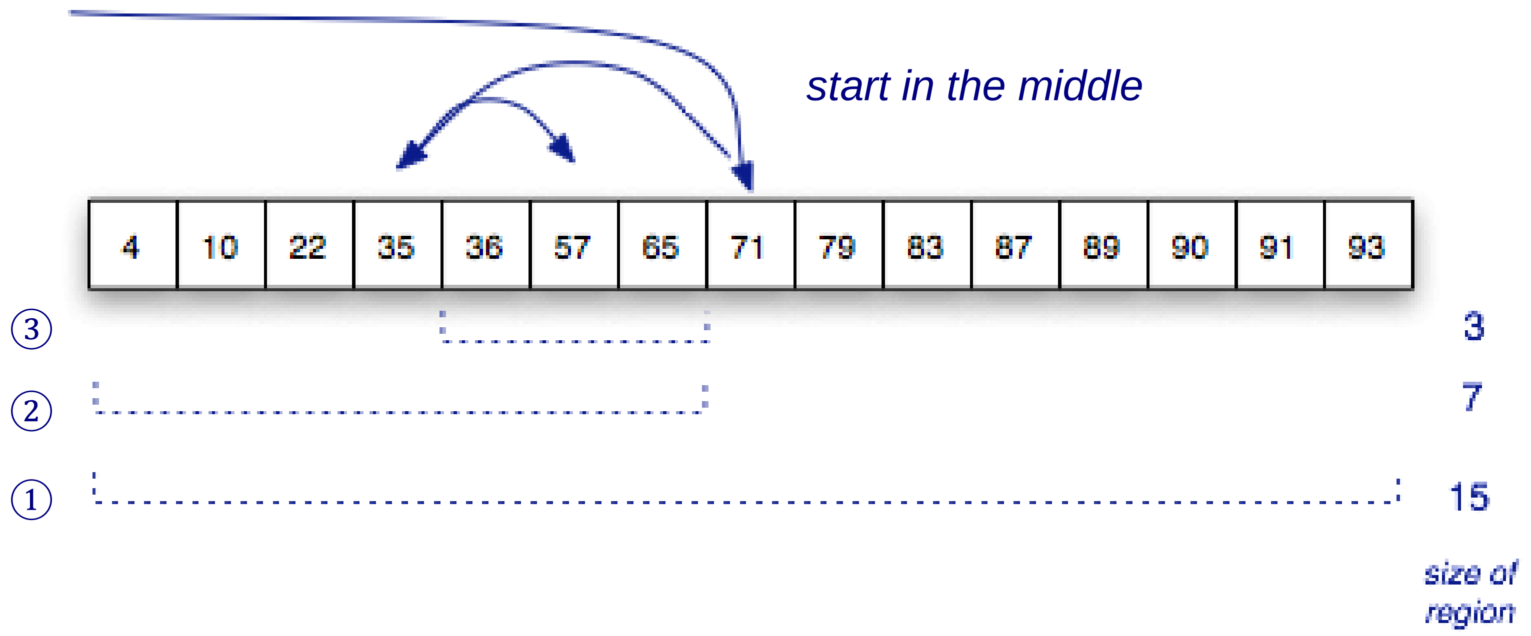
- Sorting: mergesort and quicksort
- Binary tree traversals
- Binary search (?)

Binary Search

- The binary search algorithm uses the divide-and-conquer strategy to search through an array
- The array ***must be sorted***
  - the “zeroing in” strategy for looking up a word in the dictionary won’t work it the words are not in alphabetical order
  - binary search will not work unless the array is sorted

Binary Search (contd.)

- To search a list of *n* items, first look at the item in location *n*/2
  - then search either the region from 0 to *n*/2-1 or the region from *n*/2+1 to *n*-1
- **Example:** searching for 57 in a sorted list of 15 numbers



Binary Search (contd.)

- The algorithm uses two variables to keep track of the boundaries of the region to search

  lower the index ***one below*** the leftmost item in the region

  upper the index ***one above*** the rightmost region

4	10	22	35	36	57	65	71	79	83	87	89	90	91	93

initial values when searching an array of n items:

lower = -1

upper = n

Binary Search (contd.)

- The algorithm is based on an iteration ("loop") that keeps making the region smaller and smaller
  - the initial region is the complete array
  - the next one is either the upper half or lower half
  - the one after that is one quarter, then one eighth, then...

4	10	22	35	36	57	65	71	79	83	87	89	90	91	93

initial values when searching an array of n items:

lower = -1

upper = n

Binary Search (contd.)

- The heart of the algorithm contains these operations:

mid = (lower + upper) / 2

return mid if k == a[mid]

upper = mid if k < a[mid]

lower = mid if k > a[mid]

- The first iteration when searching for 57 in a list of size 15:

4	10	22	35	36	57	65	71	79	83	87	89	90	91	93
							*							

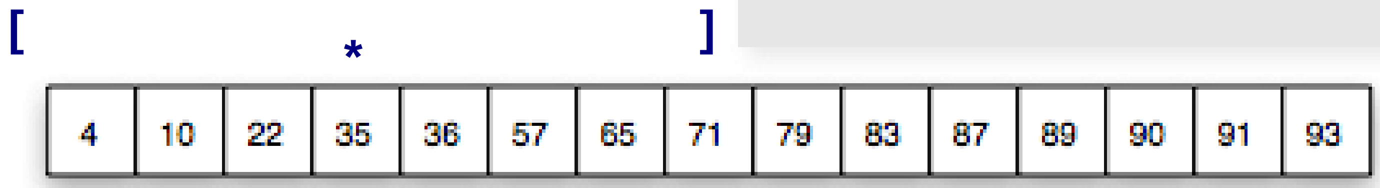
```
lower = -1      mid = 14 / 2 = 7      upper for next
upper = 15                            iteration: 7
```

Binary Search (contd.)

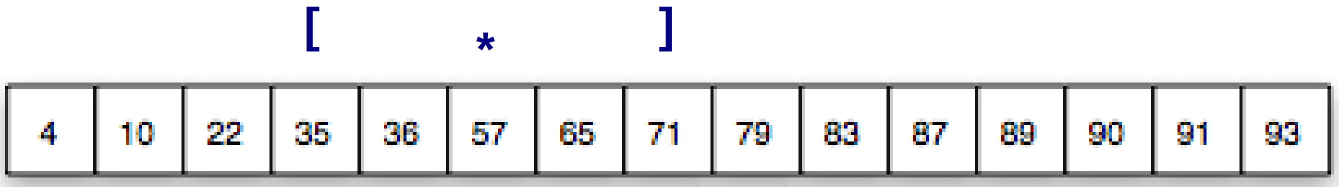
- The remaining iterations when searching for 57:

```
mid = (lower + upper) /
2return mid if k ==
a[mid]upper = mid if k <
a[mid]lower = mid if k >
a[mid]
```

lower = -1
upper = 7
mid = 3
lower = 3



lower = 3
upper = 7
mid = 5
found it!

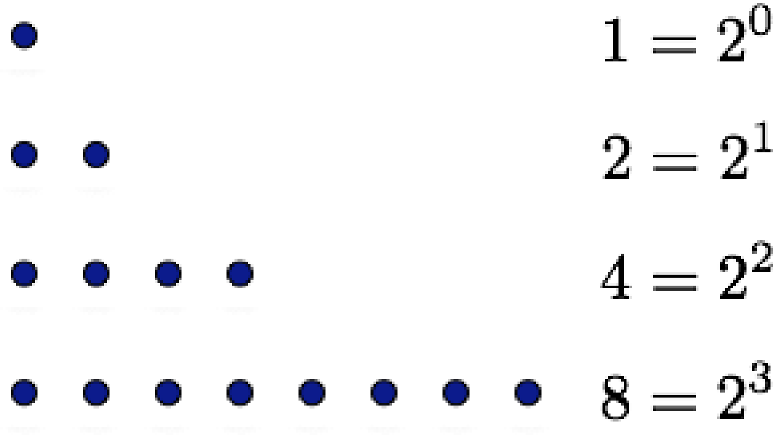


This search required only 3 comparisons:

`a[7],a[3],a[5]`

Binary Search (contd.)

- The number of iterations made by this algorithm when it searches an array of *n* items is roughly
- To see why, consider the question from the other direction
  - suppose we have an array that starts out with 1 item
  - suppose each step of an iteration doubles the size of the array
  - after *n* steps we will have 2^n items in the array



The complexity of Binary search algorithm is O (log n)

Divide-and-Conquer - Advantages

- Complexity can be reduced using the concepts of divide and conquer.
- Increase in productivity by allowing multiple programmers to work on different parts of the project independently at the same time.
- Modules can be re-used many times, thus it saves time, reduces complexity and increase reliability.
- Easier to update/fix the program by replacing individual modules rather than larger amount of code.
- Ability to either eliminate or at least reduce the necessity of employing GOTO statement
- Solves difficult problems with less time complexity than its brute-force counterpart.
- Since the sub-problems are independent, they can be computed in parallel

Divide and Conquer - Disadvantages

- Problem decomposition may be very complex and therefore not actually suitable to divide and conquer.
- Recursive nature of the solution may end up duplicating sub-problems, dynamic solutions may be better in some of these cases, like Fibonacci.
- Recursion into small/tiny base cases may lead to huge recursive stacks, and efficiency can be lost by not applying solutions earlier for larger base cases.

Complexity

- The complexity of the divide and conquer algorithm is calculated using the master theorem.

$T(n) = aT(n/b) + f(n)$,
where,
n = size of input
a = number of subproblems in the recursion
n/b = size of each subproblem. All subproblems are assumed to have the same size.
$f(n)$ = cost of the work done outside the recursive call, which includes the cost of dividing the problem and cost of merging the solutions

Frequently Asked Questions

- What do you understand about binary search?
- Define complexity of binary search.

References

1) https://www.tutorialspoint.com/data_structures_algorithms/divide_and_conquer.htm

2) **Data Structures and Algorithms made easy** By Narasimha Karumanchi.

3) The Algorithm Design Manual, 2nd Edition by Steven S Skiena

4) **Fundamentals of Computer Algorithms - Horowitz and Sahani**

THANK YOU