CSEN 703/707 - Analysis and Design of Algorithms

Lecture 3 - Divide and Conquer I

Nourhan Ehab
nourhan.ehab@guc.edu.eg

Department of Computer Science and Engineering
Faculty of Media Engineering and Technology

Problem of the Day

Example

You are playing a guessing game. The game organizer chooses a secret number n between 0 and 100. Your task is to guess what n is. You can guess any number x and ask the organizer if x is n. The organizer will reply saying that either x is smaller, larger, or equal to n. You need to identify n by asking the organizer the minimum number of questions or you will be eliminated from the game. How can you do this?

Outline

GUC German University in Cairo

Divide and Conquer

Philosophy: Philip II of Macedon

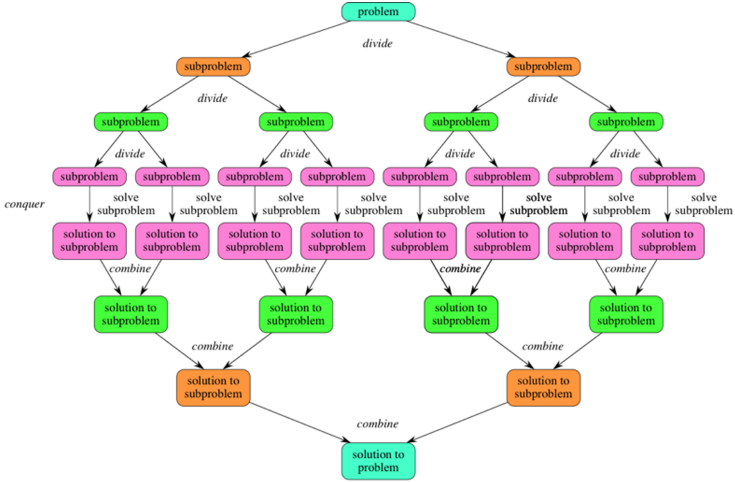
You can't beat them when they are together. Split them first and handle each faction separately.

Steps of Designing a D&C Algorithm

GUC German University in Cairo

1. Divide the overall problem into subproblems. A subproblem here will be a smaller instance of the same type of problem.
2. Conquer the sub-problems by solving them recursively. If the subproblem sizes are small enough, however, just solve the subproblems in a straightforward manner.
3. Combine the solutions to the subproblems to construct the solution of the bigger original problem.

Steps of Designing a D&C Algorithm



Binary Search

Binary Search

```
BinarySearch(A, key, i, j)
if j >= i then
    mid = ⌊(i+j)/2⌋ ;
    if A[mid] == key then
        return mid ;
    end
    if A[mid] > key then
        BinarySearch(A, key, i, mid − 1);
        else
            BinarySearch(A, key, mid + 1, j);
        end
    end
end
return -1;
```

Recurrences

- A recurrence is an equation or inequality that describes a function in terms of its value on smaller inputs.
- We use recurrences to express the running time of recursive algorithms.
- General format:

$$T(n) = \begin{cases} \Theta(1) & \text{if } n = 1 \\ aT(b) + D(n) + C(n) & \text{otherwise} \end{cases}$$

where a is the number of subproblems, b is the size of each subproblem in terms of n, $D(n)$ is the divide time, $C(n)$ is the combine time.

Recurrences

Example (Binary Search)

Express the running time of binary search as a recurrence.

$$T(n) = \begin{cases} \Theta(1) & \text{if } n = 1 \\ T(\frac{n}{2}) + \Theta(1) + 0 & \text{otherwise} \end{cases}$$

Remember that binary search only works when the array is sorted!

Outline

Merge Sort as a D&C Algorithm

1. Divide the array into two halfs.
2. Conquer the problem by recursively sorting the subarrays in each of the two subproblems created by the divide step.
3. Combine by merging the two sorted subarrays back into a single sorted subarray.

Merge Sort Example

Example

Trace Merge sort on $A = [99, 6, 86, 15, 58, 35, 86, 0]$.
(https://opendsa-server.cs.vt.edu/embed/mergesortAV)

Merge Sort

```
MergeSort(A)
if Length(A)==1 then
    return A ;
    else
        mid = ⌊Length(A)/2⌋ ;
    end
end
L = A[1, ..., mid] ;
R = A[mid + 1, ..., n] ;
MergeSort(L) ;
MergeSort(R) ;
Merge(L, R, A);
```

Merge Procedure

```
Merge(L, R, A)
nL = Length(L) ; nR = Length(R) ; i = j = k = 1 ;
while i ≤ nL && j ≤ nR do
    if L[i] <= R[j] then
        A[k] = L[i] ; i++ ;
        else
            A[k] = R[j]; j++ ;
        end
    end
    k++ ;
end
while i ≤ nL do
    A[k] = L[i]; i++; k++;
end
while i ≤ nR do
    A[k] = R[j]; j++; k++;
end
```

Merge Sort Recurrence

Example

Express the running time of merge sort as a recurrence.

$$T(n) = \begin{cases} \Theta(1) & \text{if } n = 1 \\ 2T(\frac{n}{2}) + \Theta(1) + \Theta(n) & \text{otherwise} \end{cases}$$

Outline

1. Divide and Conquer Algorithms
2. Merge Sort
3. Solving Recurrences
4. Largest Subrange
5. Recap

Solving Recurrences

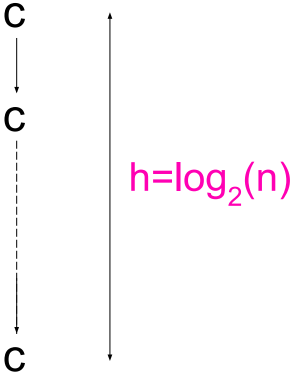
- By solving a recurrence, we mean obtaining an upper bound on its running time.
- Two methods:
  1. Recursion tree method: converts the recurrence into a tree whose nodes represent the costs incurred at various levels of the recursion. We use techniques for bounding summations to solve the recurrence.
  2. Master Theorem $\Rightarrow$ Next Lecture!

Solving Recurrences

Example

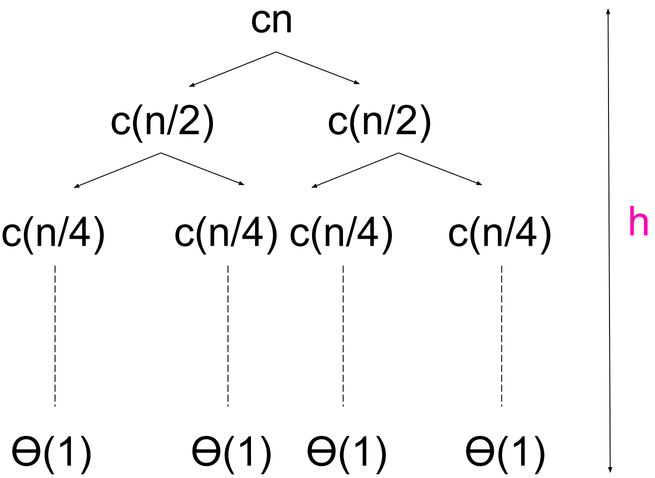
Using the recursion tree method obtain a bound on the running time of binary search and merge sort using the Θ notation.

Recursion Tree Method - Binary Search



$$T(n) = \Sigma_{i=0}^{log(n)} c = c(log(n) + 1) = c\ log(n) = \Theta(log(n))$$

Recursion Tree Method - Merge Sort



Recursion Tree Method - Merge Sort

$\frac{n}{2^h} = 1 \Rightarrow h = log(n)$

Cost of leaves $= c\ 2^h = c^{2^{log(n)}} = c\ n^{log(2)} = cn$

Cost of the rest of the tree $= \Sigma_{i=0}^{log(n)-1} cn = cn(log(n))$

$T(n) = cn\ log(n) + cn = \Theta(n\ log(n))$

Is merge sort always faster than insertion sort?

Recursion Tree Method - One More Example

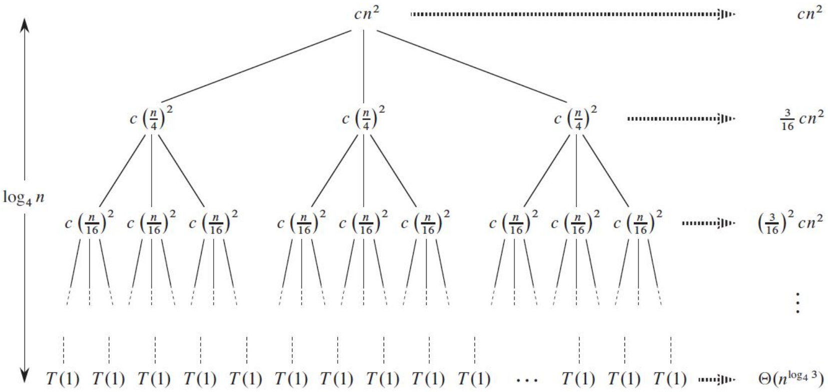
GUC German University in Cairo

Example

Use the recursion tree method to solve the recurrence

$$T(n) = 3T(\frac{n}{4}) + \Theta(n^2)$$

Recursion Tree Method - One More Example



Recursion Tree Method - One More Example

Recall the geometric series $\Sigma_{i=0}^{n} r^i = \frac{1-r^{n+1}}{1-r}$.

$\frac{n}{4^h} = 1 \Rightarrow h = log_4(n)$

Cost of leaves $= c\ 3^h = c\ 3^{log_4(n)} = c\ n^{log_4(3)}$

Cost of the rest of the tree $= \Sigma_{i=0}^{log_4(n)-1} (\frac{3}{16})^i cn^2$

$$= cn^2\ \Sigma_{i=0}^{log_4(n)-1} (\frac{3}{16})^i = cn^2\ (\frac{1-(\frac{3}{16})^{log_4(n)}}{1-\frac{3}{16}}) = \frac{16}{13} cn^2\ (1 - \frac{3^{log_4(n)}}{16^{log_4(n)}})$$

$$= \frac{16}{13} cn^2\ (1 - \frac{n^{log_4(3)}}{n^2}) = \Theta(n^2)$$

Recursion Tree Method - Six Steps

1. Draw the recursion tree.
2. Figure out the height of the tree h.
3. Cost of the leaves $=$ number of leaves $\times$ c.
4. Figure out a formula representing the cost of each level (possibly in terms of the level number).
5. Cost of the rest of the tree $= \Sigma_{i=0}^{h-1}$ cost of each level.
6. Total running time $=$ cost of leaves $+$ cost of the rest of the tree.

Outline

Largest Subrange Problem

Example

Suppose you are tasked with writing the advertising copy for a hedge fund whose monthly performance this year was

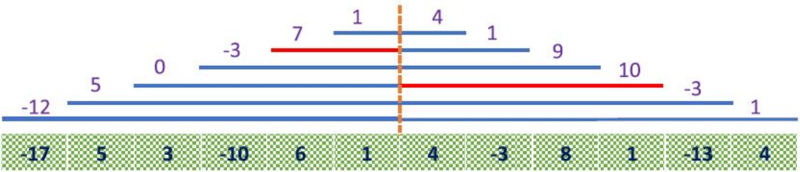
$$[-17, 5, 3, -10, 6, 1, 4, -3, 8, 1, -13, 4]$$

You lost money for the year, but from May through October you had your greatest gains over any period, a net total of 17 units of gains. This gives you something to brag about! Write an efficient algorithm to determine the period with the maximum gain given any array A representing the hedge fund's performance.
In general: The largest subrange problem takes an array A of n numbers, and asks for the value of the maximum subrange $S = \Sigma_{k=i}^{j} A[k]$.

Largest Subrange - D&C Solution Intuition

1. Divide the given array in two halves.
2. Return the maximum of following three:
   a. Maximum subarray sum in left half (Make a recursive call).
   b. Maximum subarray sum in right half (Make a recursive call).
   c. Maximum subarray sum such that the subarray crosses the midpoint.

Largest Subrange - D&C Solution Intuition



Largest Subrange D & C - Pseudo Code

LargestSubrange(A, i, j)
if $i == j$ **then**
    return $A[i]$;
end
$mid = \lfloor \frac{i+j}{2} \rfloor$;
return max(LargestSubrange(A, i, mid),
LargestSubrange$(A, mid + 1, j)$,
MaximumCrossing(A, i, j, mid))

Largest Subrange D & C - Pseudo Code

```
MaximumCrossing(A, i, j, mid)
sumLeft = sumRight = sum = 0 ;
for (k = mid; k >= i; k − −) do
    sum+ = A[k] ;
    if sum > sumLeft then
        sumLeft = sum ;
    end
end
sum = 0 ;
for (k = mid + 1; k <= j; k + +) do
    sum+ = A[k];
    if sum > sumRight then
        sumRight = sum ;
    end
end
return max(sumLeft, sumRight, sumLeft + sumRight)
```

Largest Subrange D&C - Analysis

$$T(n) = \begin{cases} \Theta(1) & \text{if } n = 1 \\ 2T(\frac{n}{2}) + \Theta(1) + \Theta(n) & \text{otherwise} \end{cases}$$

This recurrence is similar to merge sort.

Hence, $T(n) = \Theta(n\ log(n)) \Rightarrow$ much better than the naive $\Theta(n^2)$ solution!

Applications

- Searching and sorting lies at the heart of many CS problems including web applications, databases, and cryptography.
- Largest Subrange is used in Genomic sequence analysis to identify important biological segments of protein sequences. It is also used in computer vision to detect the brightest area in an image.

Outline

GUC German University in Cairo

Points to Take Home

1. The Divide and Conquer Paradigm.
2. Solving recurrences using the recursion tree method.
3. Binary search run time analysis.
4. Merge sort algorithm and analysis.
5. The largest subrange D&C solution.
6. Reading Material:
   - Introduction to Algorithms, Chapter 4: Sections 4.1 and 4.4.
   - The Algorithm Design Manual, Chapter 5: Sections 5.1, 5.3, and 5.6.

Next Lecture: The Master Theorem and Quick Sort!

Due Credits

The presented material is based on:

1. Previous editions of the course at the GUC due to Dr. Wael Aboulsaadat, Dr. Haythem Ismail, Dr. Amr Desouky, and Dr. Carmen Gervet.
2. Stony Brook University's Analysis of Algorithms Course.
3. MIT's Introduction to Algorithms Course.
4. Stanford's Design and Analysis of Algorithms Course.