CS207 Design and Analysis of Algorithms

Sajith Gopalan

Indian Institute of Technology Guwahati

sajith@iitg.ac.in

January 17, 2022

Divide and Conquer

Divide and Conquer

- solve a problem recursively
  - Divide the problem into a number of smaller instances of the same problem
  - For each of these subproblems, if its size is sufficiently large, then Conquer it recursively, else Conquer it directly
  - Combine the solutions to the subproblems into the solution for the original problem

Maximum subarray problem

- Given is an array $A[1, \dots, n]$ of integers
- Required to find (i, j) such that $i < j$ and $A[i \dots j]$ is a contiguous subarray that has the maximum sum
- Example:
- Consider $A[1 \dots 5]$ that contains -2, 1, 3, -7, 5 in locations 1 to 5 in that order
- Pairs $(1, 2), (1, 3), (1, 4), (1, 5), (2, 3), (2, 4), (2, 5), (3, 4), (3, 5), (4, 5)$ correspond to sums $-1, 2, -5, 0, 4, -3, 2, -4, 1, -2$ respectively
- The maximum subarray of A is given by $(2, 3)$ and corresponds to sum 4.

A brute force solution

- Examine every subarray as in the above example
- There are $\binom{n}{2}$ subarrays
- This algorithm has a time complexity of $\Theta(n^2)$
- Can we do faster?

A Divide and Conquer solution

- Divide the array into two equal halves
- Find the maximum subarray on either side recursively
- The smaller of the two is rejected
- The remaining one or the maximum subarray that spans *mid* is the answer
- Here $mid = \lfloor (1 + n)/2 \rfloor$

Divide and Conquer solution Continued

- How do we find the maximum subarray that spans *mid*?
- Find the maximum right justified subarray to the left of *mid*
- Find the maximum left justified subarray to the right of *mid*
- The concatenation of the two will be the maximum subarray that spans *mid*
- No *mid*-spanning subarray can sum to a larger value

Divide and Conquer solution Continued

Algorithm 1 To find the maximum left justified subarray to the right of *mid*

```
rsum = −∞; sum = 0;
for i = mid + 1 to n do
    sum = sum + A[i];
    if sum > rsum then
        rsum = sum; r = i;
    end if
end for
```

This algorithm runs in $O(n)$ time
The maximum right justified subarray to the left of *mid* can be found symmetrically.

Analysis

- The maximum subarray that spans *mid*, therefore, can be found in $O(n)$ time
- So, the time complexity of the recursive algorithm is given by $T(n) = 2T(n/2) + cn$
- The solution is $T(n) = O(n \log n)$