Lab02-Divide & Conquer and Greedy Approach

Algorithm and Complexity, Xiaofeng Gao, Spring 2022.

$*$ If there is any problem, please contact TA Wanghua Shi.

$*$ Name:_________ Student ID:_________ Email: ____________

1. Can Master Theorem apply to the recursive formula $T(n) = 2T(\frac{n}{5}) + O(\log n)$? What is the time complexity of $T(n)$ thereby?

2. Given an array of positive integers, we will implement *floating point division* between adjacent integers. For instance, given an array [66, 22, 15, 78], we will execute $66/22/15/78 \approx 0.003$. However, you can add some parentheses at any position to change the priority of arithmetic and get a maximum quotient. Given an input array, design an algorithm to output an arithmetic with the maximum quotient, be sure to avoid redundant parentheses. For example, given the above input "[66, 22, 15, 78]", your algorithm should output "66/(22/15/78)", because it is the maximum quotient (illustrated as follows):

   - $66/22/15/78 \approx 0.003$;
   - $66/(22/15/78) = 3510$;
   - $66/(22/15)/78 \approx 0.58$;
   - $66/22/(15/78) = 15.6$;
   - $66/(22/(15/78)) \approx 0.58$.

3. Given an array $A = [a_1, \cdots, a_n]$, we define "k-reverse" operation $(1 \leq k \leq n)$ as reversing the sub-array $[a_1, a_2, \cdots, a_k]$, *i.e.*, changing $A = [a_1, a_2, \cdots, a_k, a_{k+1}, \cdots, a_n]$ to $A = [a_k, a_{k-1}, \cdots, a_1, a_{k+1}, \cdots, a_n]$. For instance, if we perform a "3-reverse" operation on array $A = [72, -16, -38, 9]$, we can get the result $A = [-38, -16, 72, 9]$. Please design an algorithm to sort A in ascending order only by reverse operations. Output the list of k values per step and analyze its time complexity. For instance, given an array $A = [3, 2, 4, 1]$, your output should be as follows:

   Round 1: $k = 4$, $A = [1, 4, 2, 3]$;

   Round 2: $k = 2$, $A = [4, 1, 2, 3]$;

   Round 3: $k = 4$, $A = [3, 2, 1, 4]$;

   Round 4: $k = 3$, $A = [1, 2, 3, 4]$.

4. A *perfect array* A with n numbers satisfies: (1) it is a permutation of integers in the range of $[1, n]$; and (2) there is no index k with $1 \leq i < k < j \leq n$ where $2 \cdot A[k] = A[i] + A[j]$. For any positive integer n, design an algorithm to generate a perfect array A of length n (any perfect array is acceptable).

Remark: You need to include your .pdf and .tex files in your uploaded .rar or .zip file.