CSE 431/531: Algorithm Analysis and Design (Spring 2020)

Divide-and-Conquer – Recitation

Lecturer: Shi Li

Department of Computer Science and Engineering
University at Buffalo

Solving Recurrences

For each of the following recurrences, use the master theorem to give the tight asymptotic upper bound.

1. $T(n) = 4T(n/3) + O(n)$. $T(n) = O(\qquad)$
2. $T(n) = 3T(n/3) + O(n)$. $T(n) = O(\qquad)$
3. $T(n) = 4T(n/2) + O(n^2\sqrt{n})$. $T(n) = O(\qquad)$
4. $T(n) = 8T(n/2) + O(n^3)$. $T(n) = O(\qquad)$

Solving Recurrences

For each of the following recurrences, use the master theorem to give the tight asymptotic upper bound.

1. $T(n) = 4T(n/3) + O(n)$. $T(n) = O(n^{\lg_3 4})$
2. $T(n) = 3T(n/3) + O(n)$. $T(n) = O(\quad)$
3. $T(n) = 4T(n/2) + O(n^2\sqrt{n})$. $T(n) = O(\quad)$
4. $T(n) = 8T(n/2) + O(n^3)$. $T(n) = O(\quad)$

Solving Recurrences

For each of the following recurrences, use the master theorem to give the tight asymptotic upper bound.

1. $T(n) = 4T(n/3) + O(n)$. $T(n) = O(n^{\lg_3 4})$
2. $T(n) = 3T(n/3) + O(n)$. $T(n) = O(n \lg n)$
3. $T(n) = 4T(n/2) + O(n^2\sqrt{n})$. $T(n) = O(\quad)$
4. $T(n) = 8T(n/2) + O(n^3)$. $T(n) = O(\quad)$

Solving Recurrences

For each of the following recurrences, use the master theorem to give the tight asymptotic upper bound.

1. $T(n) = 4T(n/3) + O(n)$. $T(n) = O(n^{\lg_3 4})$
2. $T(n) = 3T(n/3) + O(n)$. $T(n) = O(n \lg n)$
3. $T(n) = 4T(n/2) + O(n^2\sqrt{n})$. $T(n) = O(n^2\sqrt{n})$
4. $T(n) = 8T(n/2) + O(n^3)$. $T(n) = O(\qquad)$

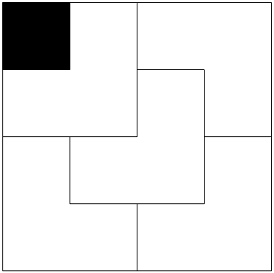
Solving Recurrences

For each of the following recurrences, use the master theorem to give the tight asymptotic upper bound.

1. $T(n) = 4T(n/3) + O(n)$. $\quad T(n) = O(n^{\lg_3 4})$
2. $T(n) = 3T(n/3) + O(n)$. $\quad T(n) = O(n \lg n)$
3. $T(n) = 4T(n/2) + O(n^2\sqrt{n})$. $\quad T(n) = O(n^2\sqrt{n})$
4. $T(n) = 8T(n/2) + O(n^3)$. $\quad T(n) = O(n^3 \lg n)$

Covering Chessboard using L-shape Tiles

Consider a $2^n \times 2^n$ chessboard with one arbitrary chosen square removed. Prove that any such chessboard can be tiled without gaps by L-shaped pieces, each composed of 3 squares. The following figure shows how to tile a 4×4 chessboard with the square on the left-top corner removed, using 5 L-shaped pieces.



Finding Local Minimum In a 1-D Array

Given an array $A[1 .. n]$ of n **distinct** numbers, we say that some index $i \in \{1, 2, 3 \cdots, n\}$ is a local minimum of A, if $A[i] < A[i-1]$ and $A[i] < A[i+1]$ (we assume that $A[0] = A[n+1] = \infty$). Suppose the array A is already stored in memory. Give an $O(\lg n)$-time algorithm to find a local minimum of A.

Finding Local Minimum In a 2-D Matrix(Hard Problem)

Given a two-dimensional array $A[1 .. n, 1 .. n]$ of n^2 **distinct** numbers, and $i, j \in \{1, 2, \cdots, n\}$, we say that (i, j) is a local minimum of A, if
$A[i, j] < A[i, j-1], A[i, j] < A[i, j+1], A[i, j] < A[i-1, j]$ and $A[i, j] < A[i+1, j]$ (we assume that $A[i, j] = \infty$ if $i \in \{0, n+1\}$ or $j \in \{0, n+1\}$).
Suppose the array A is already stored in memory. Give an $O(n)$-time algorithm to find a local minimum of A.

Integer Multiplication

Given two n-digit integers, output their product. Design a $n^{\log_2 3}$-time algorithm to solve the problem. Notice that you can not multiple two big integers directly using a single operation.

Majority and Weak Majority

Given an array of integers $A[1..n]$, we would like to decide if

1. there exists an integer x which occurs in A more than $n/2$ times. Give an algorithm which runs in time $O(n)$.
2. there exists an integer x which occurs in A more than $n/3$ times. Give an algorithm which runs in time $O(n)$.

You can assume we have the algorithm Select as a black-box, which, given an n-size array A and integer $1 \leq i \leq n$, can return the i-th smallest element in a size n-array in $O(n)$-time.

Median of Two Sorted Arrays

Given two sorted arrays A and B with total size n, you need to design and analyze an $O(\log n)$-time algorithm that outputs the median of the n numbers in A and B. You can assume n is odd and all the numbers are distinct.For example,

- Input: $A = [3, 5, 12, 18, 50]$,
- $B = [2, 7, 11, 30]$,
- Output: 11
- Explanation: the merged set is $[2, 3, 5, 7, 11, 12, 18, 30, 50]$