CSCI-UA.0310-001/002 Basic Algorithms September 25, 2016

Solutions to Problem 1 of Homework 3 (5 points)

Name: Name (ijs253) *Due: Tuesday, September 27*

Collaborators: NetID1, NetID2

Let n be a multiple of m. Design an algorithm that can multiply an n-bit integer with an m-bit integers in time $O(nm^{\log_2 3-1})$.

Solution:

Assuming that taking the log of a given n-bit integer is a constant cost:

MULTIPLY(a, b):
    first $= log(a)/log(2)$
    second $= log(b)/log(2)$
    s $= first + second$
    return 2^s

□

CSCI-UA.0310-001/002 Basic Algorithms September 25, 2016

Solutions to Problem 2 of Homework 3 (10 points)

Name: Name (ijs253) *Due: Tuesday, September 27*

Collaborators: NetID1, NetID2

An n-tromino is a $2^n \times 2^n$ "chessboard of unit squares with one corner removed" (figure below drawn for $n = 3$). Assume that initially you are given a 1-tromino (i.e., a simple L-shaped tile of area 3 drawn on the right), but you have a friendly genie whom you can ask to perform the following two operations in any order:

- DUPLICATE: This operation takes an object as input, and creates a second identical copy of this object.
- GLUE: This operation takes two objects as input, and glues them together (along the sides, without any overlaps) in a manner specified by you.

(a) (6 points) Design a recursive algorithm TROMINO(n) that creates an n-tromino from 1-tromino using the a minimum number of calls to the genie. You only need to specify the top level of the recursion, without the need to explicitly "unwind" the recursion all the way to $n = 1$. (**Hint**: First step is to DUPLICATE the original 1-tromino, as otherwise you "lose" it in the recursive call(s), and you might need it in the "conquer" step.)

Solution:

totalSquares = (perimeter+1)/2
extraTrominoes = 0; TROMINO(n):
    if 2^n != totalSquares and tromino.count == 1:
        extraPiece = DUPLICATE x 1
        DUPLICATE x 4
        GLUE so that 1 tromino its corner meeting the inside of the original tromino and the other 3 with their insides angle being filled with original tromino corner.
        if 2^n == totalSquares:
            return ;
        else : extraPieces = DUPLICATE(extraPiece) x $(2^n)/3$
            TROMINO(n)
    else if 2^n != totalSquares and tromino.count > 1:
        DUPLICATE x 3
        GLUE the 4 pieces together so that 3 of the missing corners are pointed inwards and one of the missing corners is pointed outwards, forming a tromino of negative space.

        if 2^n == totalSquares:
            return ;

else : TROMINO(n)

else :

Use the previously calculated extra pieces to fill in trominoes where there are spaces for them with GLUE.

□

(b) (4 points) Give a recurrence relation for the number of calls $T(n)$ to the genie, and solve it.

Solution:

$$\begin{aligned} T(n) &= n * T(n) + 8 \\ &= n * T(n) + 8n^k + 8n^{k-1} \text{ ..} \\ &= n * T(n) + \sum_{i=0}^{n} 8n^{k-i} \end{aligned}$$

□

CSCI-UA.0310-001/002 Basic Algorithms September 25, 2016

Solutions to Problem 3 of Homework 3 (12 points)

Name: Name (ijs253) *Due: Tuesday, September 27*

Collaborators: NetID1, NetID2

An array $A[0 \dots (n-1)]$ is called *rotation-sorted* if there exists some some cyclic shift $0 \le c < n$ such that $A[i] = B[(i + c \bmod n)]$ for all $0 \le i < n$, where $B[0 \dots (n-1)]$ is the sorted version of A.[1] For example, $A = (2, 3, 4, 7, 1)$ is rotation-sorted, since the sorted array $B = (1, 2, 3, 4, 7)$ is the cyclic shift of A with $c = 1$ (e.g. $1 = A[4] = B[(4+1) \bmod 5] = B[0] = 1$). For simplicity, below let us assume that n is a power of two (so that can ignore floors and ceilings), and that all elements of A are distinct.

(a) (4 points) Prove that if A is rotation-sorted, then one of $A[0 \dots (n/2-1)]$ and $A[n/2 \dots (n-1)]$ is fully sorted (and, hence, also rotation-sorted with $c = 0$), while the other is at least rotation-sorted. What determines which one of the two halves is sorted? Under what condition *both halves* of A are sorted?

Solution: INSERT YOUR SOLUTION HERE □

(b) (8 points) Assume again that A is rotation-sorted, but you are not given the cyclic shift c. Design a divide-and-conquer algorithm to compute the minimum of A (i.e., $B[0]$). Carefully prove the correctness of your algorithm, write the recurrence equation for its running time, and solve it. Is it better than the trivial $O(n)$ algorithm? (**Hint**: Be careful with $c = 0$ an $c = n/2$; you might need to handle them separately.)

Solution: INSERT YOUR SOLUTION HERE □

[1] Intuitively, A is either completely sorted (if $c = 0$), or (if $c > 0$) A starts in sorted order, but then "falls off the cliff" when going from $A[n-c-1] = B[n-1] = \max$ to $A[n-c] = B[0] = \min$, and then again goes in increasing order while never reaching $A[0]$.

CSCI-UA.0310-001/002 Basic Algorithms September 25, 2016

Solutions to Problem 4 of Homework 3 (8 points)

Name: Name (ijs253) *Due: Tuesday, September 27*

Collaborators: NetID1, NetID2

Find a *divide-and-conquer* algorithm that finds the maximum and the minimum of an array of size n using at most $3n/2$ *comparisons* (between elements of the array). (Note that we are not asking for an iterative algorithm. We are asking for you to *explicitly use recursion*.) Derive an *exact* recurrence for the number of *comparisons* of your algorithm and prove it using induction.
(**Hint**: Your conquer step should make a constant number of comparisons. Be careful for what n you stop recursing.)

Solution: INSERT YOUR SOLUTION HERE □