CS 170 Algorithms Fall 2014 David Wagner HW 3

Due Sept. 19, 6:00pm

Instructions. This homework is due Friday, September 19, at 6:00pm electronically. Same rules as for prior homeworks.

1. (10 pts.) Practice with recurrence relations

Solve the following recurrence relations. Express your answer using $\Theta(\cdot)$ notation. (For instance, if you were given the recurrence relation $T(n) = T(n-1)+3$, the solution would be $T(n) = \Theta(n)$. $T(n) = O(n^2)$ would receive no credit, even though it is a valid upper bound, as it is not the best upper bound.) You don't need to show your work or justify your answer for this problem.

(a) $F(n) = F(n-4)+1$.

(b) $G(n) = G(n/2)+3$.

(c) $H(n) = H(n/4)+1$.

(d) $I(n) = I(n/2)+n$.

(e) $J(n) = 2J(n/4)+6$.

(f) $K(n) = 2K(n/4)+n$.

(g) $L(n) = 2L(3n/4)+1$.

(h) $M(n) = 2M(n-1)+1$.

2. (15 pts.) Procedural Terrain Generation

Recursive algorithms can be useful for generating objects with fractal structure, like realistic rocky terrain. We can store the shape of the ground as an $(n+1) \times (n+1)$ array of height values, so that $H[x][y]$ is the height at point (x,y). Say the heights at the corners $(0,0)$, $(0,n)$, $(n,0)$, and (n,n) have been given, and we need to fill in the rest of the grid.

Consider the following algorithm, more complex versions of which are the basis of the terrain generation procedures in many video games. The function Rand() returns a random number and takes constant time.

Algorithm MidpointDisplacement(n):
1. Call FillIn$(0,n,0,n)$.

Algorithm FillIn(ℓ,r,t,b):
1. If $r-\ell < 2$ or $b-t < 2$, return.
2. Let $x := \lfloor (\ell+r)/2 \rfloor$.
3. Let $y := \lfloor (t+b)/2 \rfloor$.
4. Set $H[x][y] := (H[\ell][t]+H[\ell][b]+H[r][t]+H[r][b])/4 + \text{Rand}()$.
5. Set $H[\ell][y] := (H[\ell][t]+H[\ell][b])/2$.
6. Set $H[x][t] := (H[\ell][t]+H[r][t])/2$.
7. Set $H[r][y] := (H[r][t]+H[r][b])/2$.

8. Set $H[x][b] := (H[\ell][b] + H[r][b])/2$.
9. Call FillIn(ℓ, x, t, y).
10. Call FillIn(ℓ, x, y, b).
11. Call FillIn(x, r, t, y).
12. Call FillIn(x, r, y, b).

(a) What is the asymptotic runtime of MidpointDisplacement(n), as a function of n?

(b) Is there an algorithm which computes the same values for H as MidpointDisplacement() but which is asymptotically faster? Why or why not?

3. (20 pts.) Dominated?

You are given a collection of n points (x_i, y_i) in two dimensions. Design an $O(n \lg n)$ time algorithm to check whether there exists a pair of points $(x_i, y_i), (x_j, y_j)$ such that $x_i \leq x_j$ and $y_i \leq y_j$ and $i \neq j$.

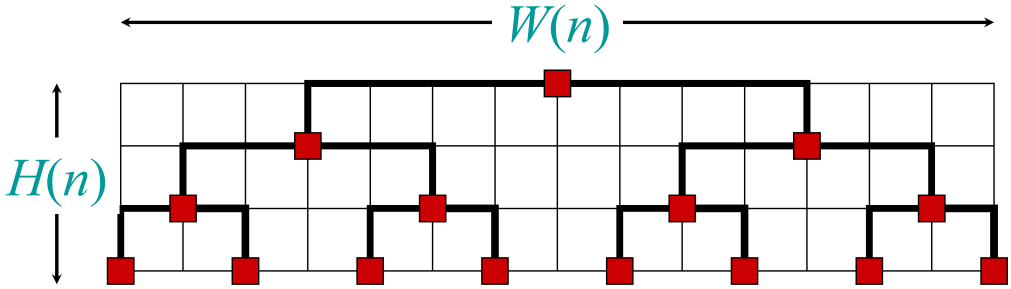
Reminder: As always, when we ask you to design an algorithm, include an explanation, algorithm pseudocode, running time analysis, and proof of correctness in your answer.

4. (25 pts.) Chip design

Charlene the chip designer approaches you with the following problem. Charlene wants to lay out a complete binary tree with n leaves on the chip, where n is a power of two. (Imagine that the leaves contain n inputs, and the goal is to compute the logical AND of these signals, using two-input AND gates.) The nodes and wires are required to follow a rectangular grid. She wants to minimize the total area required for the tree.

In the following parts, show your work and justify your answers.

(a) Charlene's first thought is to lay out the wires like this:



Let $W(n)$ denote the width of this arrangement, in number of grid-squares, and $H(n)$ the height of this arrangement. For instance, in the above picture we have $n = 8$, $H(8) = 3$, and $W(8) = 14$.

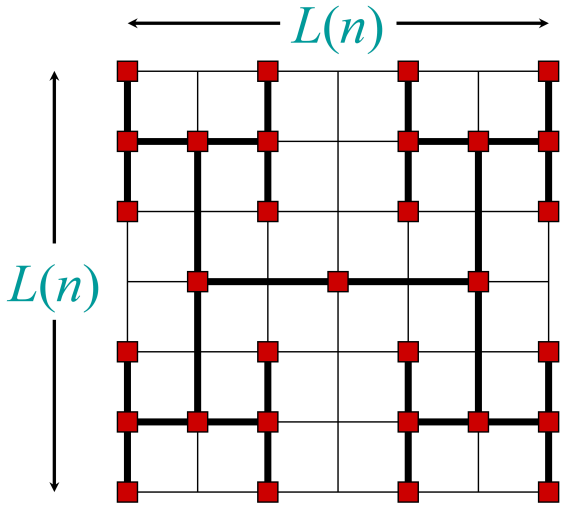
Find an explicit formula for $H(n)$.

(b) Find a recurrence relation for $W(n)$.

(c) Solve the recurrence relation for $W(n)$, to obtain an asymptotic expression for $W(n)$. In other words, find a function $f(n)$ such that $W(n) \in \Theta(f(n))$.

(d) The total area of her scheme is given by $A(n) = H(n) \times W(n)$. Find an asymptotic expression for $A(n)$. In other words, find a function $f(n)$ such that $A(n) \in \Theta(f(n))$.

(e) A flash of inspiration strikes you in the shower, and you have another idea for a possible chip layout, like this:



Assume that n is an even power of 2 (e.g., 1, 4, 16, 64, etc.). Then this arrangement will be a square, so let $L(n)$ denote the length of the side of the square, as a function of n. For instance, in the picture above we have $n = 16$ and $L(n) = 6$.

Find a recurrence relation for $L(n)$.

(f) Solve the recurrence relation for $L(n)$, to obtain an asymptotic expression for $L(n)$ (up to a multiplicative factor). In other words, find a function $f(n)$ such that $L(n) \in \Theta(f(n))$.

(g) The total area of your scheme is given by $A'(n) = L(n)^2$. Find an asymptotic expression for $A'(n)$. In other words, find a function $f(n)$ such that $A'(n) \in \Theta(f(n))$.

(h) Which is better, for large n? Charlene's layout depicted in part (a), or your layout from part (e)?

5. (20 pts.) Pattern matching, with tolerance for noise

We are given binary strings s,t; s is m bits long, and t is n bits long, and $m < n$. We are also given an integer k. We want to find whether s occurs as a substring of t, but with $\leq k$ errors, and if so, find all such matches. In other words, we want to determine whether there exists an index i such that $s_0, s_1, \ldots, s_{m-1}$ agrees with $t_i, t_{i+1}, t_{i+2}, \ldots, t_{i+m-1}$ in all but k bits; and if yes, find all such indices i.

(a) Describe an $O(mn)$ time algorithm for this string matching problem. Just show the pseudocode; you don't need to give a proof of correctness or show the running time.

(b) Let's work towards a faster algorithm. Suggest a way to choose polynomials $p(x), q(x)$ of degree $m-1, n-1$, respectively, with the following property: the coefficient of x^{m-1+i} in $p(x)q(x)$ is $m - 2d(i)$, where $d(i)$ is the number of bits that differ between $s_0, s_1, \ldots, s_{m-1}$ and $t_i, t_{i+1}, t_{i+2}, \ldots, t_{i+m-1}$.

Hint: use coefficients $+1$ and -1.

(c) Describe an $O(n \lg n)$ time algorithm for this string matching problem, taking advantage of the polynomials $p(x), q(x)$ from part (b).

(d) Now imagine that s,t are not binary strings, but DNA sequences: each position is either A, C, G, or T (rather than 0 or 1). As before, we want to check whether s matches any substring of t with $\leq k$ errors (i.e., $s_0, s_1, \ldots, s_{m-1}$ agrees with $t_i, t_{i+1}, t_{i+2}, \ldots, t_{i+m-1}$ in all but k letters), and if so, output the location of all such matches. Describe an $O(n \lg n)$ time algorithm for this problem.

Hint: encode each letter into 4 bits.

6. (10 pts.) Triple sum

Design an efficient algorithm for the following problem. We are given an array $A[0..n-1]$ with n elements, where each element of A is an integer in the range $0 \leq A[i] \leq 10n$. We are also given an integer t. The algorithm must answer the following yes-or-no question: does there exist indices i, j, k such that $A[i] + A[j] + A[k] = t$?

Design an $O(n \lg n)$ time algorithm for this problem.

Hint: define a polynomial of degree $O(n)$ based upon A, then use FFT for fast polynomial multiplication.

Reminder: don't forget to include explanation, pseudocode, running time analysis, and proof of correctness.

7. (5 pts.) Optional bonus problem: More medians

(This is an *optional* bonus challenge problem. Only solve it if you want an extra challenge.)

We saw in class a randomized algorithm for computing the median, where the expected running time was $O(n)$. Design a algorithm for computing the median where the *worst-case* running time is $O(n)$.

Hint: It's all in finding a good pivot. If you divide the array into small groups of c elements, can you use that to help you a good pivot?

Additional opportunities: For still more opportunities for additional challenges and extra enrichment, and a little bit of extra credit, keep an eye on Piazza or the course web page (under the grades section).