Homework 3 Written Problems

Jessica Saviano

Question 1:

For this problem, a divide and conquer with recursion, is the way to go. I would first start off by making a new array that is the same length as the secret array, where the elements of this new array are/correspond to the index of the secret array. We would then split the new array in half and call the function to determine which half of the array the two is in. If there is an odd number of indices, the function can return any of the three values (0,1,-1). If it returns 1 or -1 we chose the next/correct half of the array to recurse on. If the function returns 0, I would check the cardinality of the two list and the one with the smaller cardinality has to have the two in it. If there are an even number of indices, the function will only return 1 or -1, and we can then pick from there which side to recurse on until there are an odd number of indices and we get a return call of 0. Therefore, this function will first be called on the full list/array of indices split in half to determine where the two is, and then it will have 2 sub lists of indices as the parameter and the function will recrusively be called and will continue until a sub list with a cardinality of one/ an odd small caridanlity is reached. This sublist will have the index of where the two is location. So, based on this algorithm the recurrence relation will be $T(n) = T(n/2) + fr$ where fr is the runtime of the function. This is the case because we call the function the first time and then recursively go down the half of indices that contains the two. So, the first call is n amount of non-recrusie work and then from there we are running the next calls on only half of the indices of the previous calls. The runtime of the reccurence relation is determined with case three of master's theorem because n grows faster than $n^k = n^0 = 1$, so $f(n)$ dominates, and the runtime answer is $\Theta(n)$.

Question 2:

To solve $T(n) = T(n-1) + n$, we first have to start by unrolling the reccurrrence in an attempt to see a pattern:
$T(n) = [T(n-2) + (n-1)] + n$
$= T(n-2) + 2n - 1$
unroll again!
$= [T(n-3) + (n-2)] + 2n - 1$
$= T(n-3) + 3n - 3$
and again..
$= T(n-4) + 4n - 6$
We can now see this is a common/worldwide famous pattern related to the handshake problem, so the pattern in: $T(n-d) + dn - (d(d-1)/2)$ where d is the depth of recursion. Now we have to see where we hit T(1):
$1 = n - d, d = n - 1$
now we sub back in for d:
$T(1) + (n-1)(n) - ((n-1)((n-1)-1)/2)$

$= T(1) + (n-1)(n) - ((n-1)((n-2)/2)$
So now, we pick the most significant term, which is $(n-1)(n-2)$ because this is equal to n^2. Therefore, the answer is $\Theta(n^2)$

Question 3:

For this problem, our guess"is that this recurrence relation is $\Theta(n^{log_3 4})$. To solve, we use $T(n) = c * n^{log_3 4}$. When we try to plug in n/3 for n we end up with an answer that does not prove the strict form of our inductive hypotheses. We are off by n, which is not allowed. So, our inductive hypothesis is$T(n) = c * n^{log_3 4} - dn$ because we have to subtract off a lower level of n for this to work. Now we have to solve for T(n).
$T(n) <= 4 * (c(n/3)^{log_3 4} - d(n/3)) + n$
$= 4((n)^{log_3 4}/4 - d(n/3)) + n$
$= c * n^{log_3 4} - (4/3)dn + n$
$T(n) <= c * n^{log_3 4} - dn$

this form exactly matches our inductive hypothesis and holds for all d >= 3. We have therefore proved by induction that the recurrence relations is and element of $O(n^{log_3 4})$

Question 4:

4) $2T(n/4) + 1$: $A = 2, B = 4, f(n) = 1, k = log2/log4 = 0.5$, so we have to compare n^k which is $n^{0.5}$ to $f(n)$. This is case 1 of masters theorem, since $\Theta(1)$ grows slower, n^k dominates, and the answer is $\Theta(n^{0.5})$
5)$2T(n/4) + 1$: $A = 2, B = 4, f(n) = n^{0.5}, k = log2/log4 = 0.5$, so we have to compare n^k which is $n^{0.5}$ to $f(n)$. This is case 2 of masters theorem, since they both grow the same, the runtime answer is $\Theta(n^{0.5} * logn)$
7)$2T(n/4) + 1$: $A = 2, B = 4, f(n) = n, k = log2/log4 = 0.5$, so we have to compare n^k which is $n^{0.5}$ to $f(n)$. This is case 3 of masters theorem, since n grows faster than n^k , $f(n)$ dominates, and the runtime answer is $\Theta(n)$
7)$2T(n/4) + 1$: $A = 2, B = 4, f(n) = n^2, k = log2/log4 = 0.5$, so we have to compare n^k which is $n^{0.5}$ to $f(n)$. This is case 3 of masters theorem, since n^2 grows faster than n^k , $f(n)$ dominates, and the runtime answer is $\Theta(n^2)$