

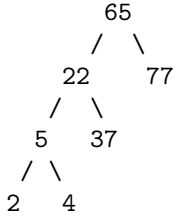
University of Toronto
ECE-345: Algorithms and Data Structures
Solutions to Midterm Examination (Fall 2008)

1. (a) False. $2^{3^{n+1}} = 2^{3 \cdot 3^n} = (2^3)^{3^n} = 8^{3^n}$, which is clearly not $O(2^{3^n})$ (compare the limits as $n \rightarrow \infty$).
- (b) True. Take the limit of $n^2 / \frac{n^3}{\lg n}$ as $n \rightarrow \infty$. We get $\frac{\lg n}{n}$, which tends to 0. Therefore, $n^2 = O(\frac{n^3}{\lg n})$.
- (c) Base case: $n = 1$ holds easily.

Induction Hypothesis: Assume statement holds for n .

Induction Step: Consider statement for $n+1$. The LHS of the statement becomes equal to $(-1)^{n+1} \binom{n(n+1)}{2} + (-1)^{n+2} (n+1)^2$ by Induction Hypothesis. This is equal to $(-1)^{n+2} (-\frac{k(k+1)}{2} + (k+1)^2) = (-1)^{n+2} (\frac{(k+1)(k+2)}{2})$, which is the LHS of the statement for $n+1$. Thus, the statement holds for $n+1$.

(d)



- (e) Master theorem doesn't apply here. Draw recursion tree. At each level, do $\Theta(n)$ work. Number of levels is $\log_{5/4} n = \Theta(\lg n)$, so guess $T(n) = \Theta(n \lg n)$ and use the substitution method to verify guess.
- (f) Apply Master theorem. We have $f(n) = n^{3/2} \lg n$ and $n^{\log_b a} = n^2$. Since $n^{3/2} \lg n = O(n^{2-\epsilon})$ for $\epsilon = 1/4$, by case 1 of the master theorem, we have $T(n) = \Theta(n^2)$.
2. (a) This is easy enough. The difference between the min and max is the largest difference. Finding both takes $O(n)$ time.
- (b) Sort the array using some $O(n \lg n)$ sorting algorithm, such as HeapSort or MergeSort. Then do a linear search on the sorted list, keeping track of the minimum distance between adjacent elements (and the pair of adjacent elements that give this distance). The search takes $O(n)$ additional time, so the entire algorithm uses $O(n \lg n)$ time.
3. (a) Not AVL tree because the left and right subtrees of B are of height 2 and 0. To make AVL tree, $\text{right-rotate}(A)$, then $\text{left-rotate}(A)$. Any two rotations that make the tree an AVL tree get full mark.
- (b) For $n > 2$, an AVL tree with root of height h contains the root node, and a child of height $h-1$. The other child must have at least height $h-2$ so that the balance condition is met. Therefore the $\text{min}(h) = 1 + \text{min}(h-1) + \text{min}(h-2)$.
4. Assume that the points $\{x_1, x_2, \dots, x_n\}$ are in sorted order, otherwise we start our algorithm by sorting them. We place the starting point of the first unit interval U_1 above x_1 and let x_i be the rightmost point covered by U_1 . Let U_2 's starting point be over x_{i+1} and continue in this fashion until all points are covered. The above process is clearly a greedy based strategy. It takes $O(n)$ time, and the whole algorithm runs in $O(n)$ time, if points are given in sorted order, or $O(n \log n)$ if they're unsorted.

To prove that the greedy strategy gives an optimal solution, let S be an optimal solution. Let the first (i.e. leftmost) unit interval of S to have x_j as a leftpoint. If $j = 1$ it agrees with our greedy method. If not, it

must be the case that $x_j < x_i$ and x_j is not one of the given n points of our set. If this is the case, then we simply "shift" the first interval of S to the right until its starting point is right above x_1 . Let $S' = S - (\text{first interval of } S) + (\text{shifted interval as described above})$. S' is still optimal and it agrees with our greedy choice in the first interval. Remove all points covered by this shifted interval, we are left with a similar but smaller problem. The same technique can be used to prove that the optimal solution can be restructured in such a way that it agrees with the greedy choice and still be optimal. Therefore, a greedy choice provides optimal solution to this problem.