Analysis of Algorithms - Final

K. Subramani
LCSEE,
West Virginia University,
Morgantown, WV
{ksmani@csee.wvu.edu}

1 Instructions

1. The Final is to be turned in by $1:00$ *pm.* in class.
2. Each question is worth 4 points. Attempt as many problems as you can. You will be given partial credit, as per the policy discussed in class.

2 Problems

1. **Induction and Recurrences:**
   (a) Professor Rabinowitz claims that the following property is true of all positive integers n: Either n is a power of 2, or there is some number between n and $2 \cdot n$, which is a power of 2. Prove that the Professor is correct.
   (b) Solve the following recurrence (exactly or asymptotically):

$$\begin{aligned} T(n) &= 1, \text{ if } n = 1 \\ &= 2 \cdot T(n-1) + 1, \text{ if } n \geq 2. \end{aligned}$$

2. **Counterexamples:** In class, we showed that the fractional knapsack problem can be solved efficiently using a greedy approach. Now consider the $0/1$ knapsack problem, in which the objective (maximizing profit) and constraint (respecting knapsack capacity) are identical to the fractional knapsack problem; however, an object is either completely selected or completely discarded, i.e., you cannot select a fraction of an object. Does the greedy strategy discussed in class result in an optimal solution for the $0/1$ knapsack problem? If so, provide a proof of the same; if not, provide a numeric counterexample.
3. **Graph Theory:** Let $\mathbf{G} =< \mathbf{V}, \mathbf{E} >$ denote an undirected, unweighted graph. The *square* of $\mathbf{G}$ is the graph $\mathbf{G^2} =< \mathbf{V}, \mathbf{E^2} >$ defined as follows: $\mathbf{G^2}$ has the same vertex set as $\mathbf{G}$. (v_i, v_j) is an edge in $\mathbf{G^2}$, if and only if, either (v_i, v_j) is an edge in $\mathbf{G}$ or (inclusively) there exists a vertex v_k, such that (v_i, v_k) and (v_k, v_j) are edges in $\mathbf{G}$. Design an efficient algorithm to compute $\mathbf{G^2}$, given that $\mathbf{G}$ is stored using an adjacency matrix.
4. **Sorting and Searching:** Explain briefly how Randomized Quickselect selects the k^{th} smallest element of an n-element integer array, using $O(n)$ comparisons, in the expected case.
5. **Greedy Strategy:** The problem of *Coin changing* is concerned with making change for a specified coin value using the fewest number of coins, with respect to the given coin denominations. Assume that your coin denominations are quarters (25 cents), dimes (10 cents), nickels (5 cents) and pennies (1 cent) and that you have an infinite supply of the same. For instance, given a coin value of 31 cents, the optimal change value is 3 coins, viz., 1 quarter, 1 nickel and 1 penny. (Any other breakup will result in more than 3 coins!) Design a greedy algorithm for the Coin changing problem and argue its correctness.