CS 577: Introduction to Algorithms — Homework 4

Out: 03/20/12 — **Due: 03/27/12**

Ground Rules

Same as for homework 1.

Problems

1. **(2+3 points)** One of your friends who has not taken an algorithms class is working at a cafe downtown and is often confronted with the problem of making change. Your friend would like a simple proceedure for doing this that uses the minimum number of coins.

   (a) Give a greedy algorithm that is optimal when the available coins are pennies, nickles, dimes, and quarters. Prove that your algorithm is optimal.

   (b) Another friend overheared you decribing your algorithm (and because it was so simple) she remembered it when on her study abroad program on Mars. When she returns she complains that she tried to use the algorithm but it did not work. Mars, remember, has a different currency that uses different denominations than the United States. Show that this is possible by giving a set of denominations that breaks the greedy algorithm.

2. **(2+4 points.)** A *vertex cover* of a graph $G = (V, E)$ is a subset of V, V', such that *every* edge in E is incident on some vertex in V'. The first three figures on the next page show graphs with vertex covers—each vertex in the cover is circled. You should verify that each example gives a valid vertex cover.

   In the minimum vertex cover problem, our goal is to find a vertex cover of a given graph with the fewest number of nodes. Figures (i) and (ii) on the next page display two different vertex covers for the same graph; the one on the left (i) has more nodes than the one on the right (ii); the latter is the minimum vertex cover for the graph.

   (a) Find the minimum vertex cover for the trees in figures (iv), (v), (vi) and (vii) on the next page. (Feel free to mark your solution on that page and submit the page with the rest of your homework.)

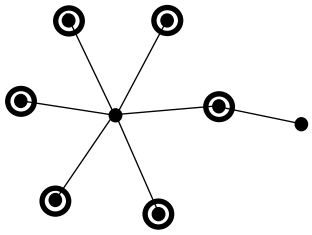
   (b) Give a linear-time algorithm that, given a tree, finds the minimum vertex cover in that tree. Prove the correctness of your algorithm. A slightly worse running time, like $O(n \log n)$ or $O(n^2)$ will receive partial credit.

   (Note: it is possible to solve this problem in linear time using a greedy strategy as well as through a dynamic program. You may use either approach.)

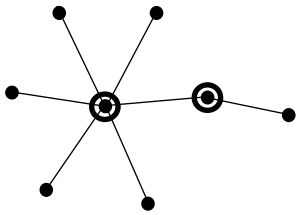
3. **(4 points)** In class, we developed an $O(nC)$ time algorithm for the Knapsack problem, where n is the number of items and C is the capacity of the knapsack. Give an algorithm for the Knapsack problem that runs in time $O(nV)$ where V is the total value of all the items (i.e. $V = \sum_{i \leq n} v_i$). The running time of your algorithm should be independent of C.

4. **(Extra credit.)** Alice and Bob play the following number search game. N is fixed to be some large number. Alice picks an integer y in the range $[1 \cdots N]$. Bob's goal is to guess this number as quickly as possible through a series of questions. At step t Bob poses a question x_t, also an integer in the range $[1 \cdots N]$. For $t \geq 2$, Alice responds by "hotter" or "colder" depending on whether x_t is closer or farther from her number y compared to the previous question x_{t-1}. That is, if $|x_t - y| < |x_{t-1} - y|$, Alice responds "hotter"; if $|x_t - y| > |x_{t-1} - y|$, Alice responds "colder"; and if the two are equally far, Alice responds "equal". The game ends as soon as Bob correctly guesses y.

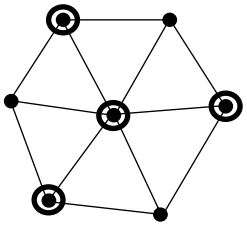
   Give a strategy for Bob that is guaranteed to find the number y in at most $\log_2 N + O(1)$ steps. (Note: it is easy to achieve this in $O(\log_2 N)$ steps; here you are asked to do it in exactly $\log_2 N$ plus a few extra steps.)



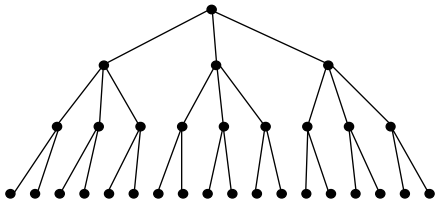
(i)



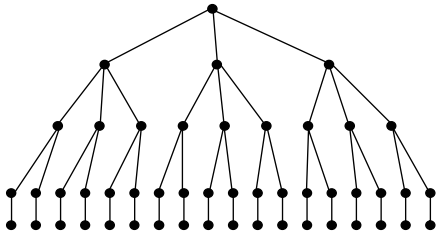
(ii)



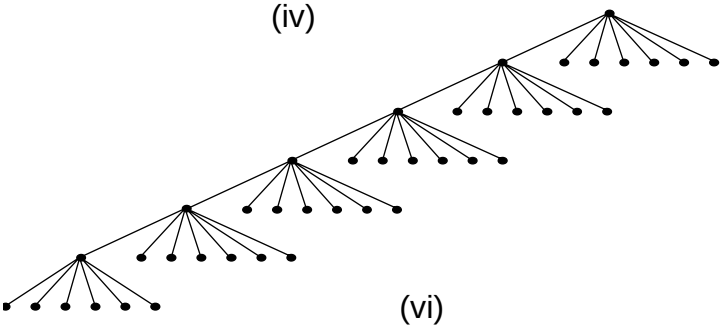
(iii)



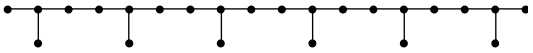
(iv)



(v)



(vi)



(vii)