CMSC 28100-1 Spring 2017
Homework 2

April 6, 2017

1. *Knapsack Problem* is defined as follows. Input: a set S of $|S| = n$ items each with a positive integer weight w_i and a nonnegative value v_i ($i \in S$), knapsack capacity W, and the target total value V. Output: 'Yes' if there is a subset $S' \subseteq S$ of items such that $\sum_{i \in S'} w_i \leqslant W$ and $\sum_{i \in S'} v_i \geqslant V$. 'No' otherwise.

(a) Give an algorithm for *Knapsack Problem.* If you give a dynamic programing algorithm, state the precise 'English' definitions of your subproblems, state base cases, and the recurrence. Try to make it as efficient as you can. Analyze its running time. Does it run in polynomial time?

(b) Define *εW-Knapsack Problem* for a fixed constant ε exactly as *Knapsack Problem* above except that you have the guarantee that, in the input, $\varepsilon W \leqslant w_i$ for each $i \in S$ (i.e. no arbitrarily small weights). Can you come up with a polynomial time algorithm for this problem? Or is your algorithm for part a already polynomial time for this problem? Why? Analyze. (expected answer is only a couple of lines) Note: ε is not a part of the input.

(c) Define *εV-Knapsack Problem* for a fixed constant ε exactly as *Knapsack Problem* above except that you have the guarantee that, in the input, $\varepsilon V \leqslant v_i$ for each $i \in S$ (i.e. no arbitrarily small values). Can you come up with a polynomial time algorithm for this problem? Analyze your algorithm's running time. (algorithm is not complicated, just two lines) Note: ε is not a part of the input.

2. We are given an array $A[1..n]$ of n real numbers, $n \geqslant 3$, with the special property that $A[1] \geqslant A[2]$ and $A[n-1] \leqslant A[n]$. We say that an element $A[x]$ is a local minimum if $A[x-1] \geqslant A[x]$ and $A[x] \leqslant A[x+1]$. First, note that, given $A[1] \geqslant A[2]$ and $A[n-1] \leqslant A[n]$, array must have at least one local minimum, and it is trivial to find it in $O(n)$ time. Give an algorithm to find and return a local minimum of A in $O(\log n)$ time. Analyze running time of your algorithm. Prove the correctness of your algorithm.

3. Given an undirected simple graph $G = (V, E)$, give a $O(|V|)$ time algorithm to find out if it has a cycle. A cycle is a path that starts and ends with the same vertex. Note that your algorithm should not depend on the number of edges in the graph. So even if the graph is very dense with $\Omega(|V|^2)$ edges, your algorithm should still run in $O(|V|)$ time. Analyze the run time of your algorithm. Algorithm is simple, but the argument to justify $O(|V|)$ running time is may not be immediate to see.