ASSIGNMENT 5

DUE: Wednesday October 18, 7 PM. DO NOT COPY. ACKNOWLEDGE YOUR SOURCES.

Please read `http://www.student.cs.uwaterloo.ca/~cs341` for general instructions and policies.

1. [10 marks] **Recurrences.**

   Suppose we are able to solve a problem recursively by removing some constant from the input size, obtaining a recurrence relation for the worst case running time, $T(n)$, of the form $T(n) = bT(n-a) + T(a) + f(n)$ for some constants $a, b \in \mathbf{N}$. Assume that $T(n) \leq d$, for $n \leq a$, where d is a constant.

   (a) Show that if $b = 1$ and $f(n) = kn$, k a constant, then $T(n)$ is $O(n^2)$.

   (b) Show that if $b = 2$ and $f(n) = k$ then $T(n)$ grows exponentially.
   "Grows exponentially" means $\Omega(g(n))$ where $g(n)$ is an exponential function. Examples of exponential functions are 2^n, $n!$, $2^{n/2}$, etc.

2. [10 marks] **Dynamic Programming for Two Knapsacks.** Consider a variation of the Knapsack problem. There are two knapsacks that have capacity $W_1 > 0$ and $W_2 > 0$, respectively. There are n items $1, 2, \ldots, n$. Item i has weight $w(i) > 0$ and two values $v_1(i) > 0$ and $v_2(i) > 0$. Here $v_k(i)$ is the value one gains by putting item i into knapsack k ($k = 1, 2$). The "Two Knapsacks Problem" is to find two *disjoint* subsets of items S_1 and S_2, such that

   1. $\sum_{i \in S_1} w(i) \leq W_1$,
   2. $\sum_{i \in S_2} w(i) \leq W_2$, and
   3. $V = \sum_{i \in S_1} v_1(i) + \sum_{i \in S_2} v_2(i)$ is maximized.

   Give a dynamic programming algorithm to find the maximum value V. Your algorithm does not need to find the sets S_1 and S_2. Clearly indicate what your subproblems are, and the order in which you solve them. Justify correctness of your algorithm, and analyze its running time. Is your algorithm a polynomial-time algorithm? Why or why not?

   The second programming assignment asks you to implement your algorithm.