CSE 331 - Summer 2015

HOMEWORK 4

Due Tuesday, August 4, 2015 @ 2:10pm

IMPORTANT: The stated deadline on assignments will be strictly enforced. I will go over solutions at the deadline and will not accept submissions after the solutions have been presented.

Homework can be submitted at any time before the deadline as either a hard copy or electronically. If submitting electronically, use a recognizable filename (ex. "`homework4.pdf`") and submit with `submit_cse331` or as an email attachment sent to me.

---

1. (40 points) Chapter 6, Exercise 1:

   Let $G = (V, E)$ be an undirected graph with n nodes. A subset of the nodes in a graph is called an *independent set* if no two of them are joined by an edge. Finding large independent sets is difficult in general, but here we'll see that it can be done efficiently if the graph is "simple" enough.

   Call a graph $G = (V, E)$ a *path* if its nodes can be written as $v_1, v_2, \ldots, v_n$, with an edge between v_i and v_j if and only if the numbers i and j differ by exactly 1. With each node v_i, we associate a positive integer weight w_i.

   The goal in this question is to solve the following problem:

   Find an independent set in a path G whose total weight is as large as possible.

   (a) Give an example to show that the following algorithm *does not* always find an independent set of maximum total weight.

The "heaviest-first" greedy algorithm

Start with S equal to the empty set
while some node remains in G **do**
  Pick a node v_i of maximum weight
  Add v_i to S
  Delete v_i and its neighbors from G
end while
Return S

   (b) Give an example to show that the following algorithm also *does not* always find an independent set of maximum total weight.

Let S_1 be the set of all v_i where i is an odd number
Let S_2 be the set of all v_i where i is an even number
(Note that S_1 and S_2 are both independent sets)
Between S_1 and S_2, return the one with larger weight

(c) Give an algorithm that takes an n-node path G with weights and returns an independent set of maximum total weight. The running time should be polynomial in n, independent of the values of the weights.

part a: 10 points

part b: 10 points

part c: 20 points

2. (45 points) Chapter 6, Exercise 2:

Suppose you're managing a consulting team of expert computer hackers, and each week you have to choose a job for them to undertake. Now, as you can well imagine, the set of possible jobs is divided into those that are *low-stress* and those that are *high-stress*. The basic question, each week, is whether to take on a low-stress job or a high-stress job.

If you select a low-stress job for your team in week i, then you get a revenue of $l_i > 0$ dollars; if you select a high-stress job, you get a revenue of $h_i > 0$ dollars. The catch, however, is that in order for your team to take on a high-stress job in week i, it's required that they do no job (of either type) in week $i - 1$; they need a full week of prep time to get ready for the crushing stress level. On the other hand, it's okay for them to take a low-stress job in week i even if they have done a job (of either type) in week $i - 1$.

So, given a sequence of n weeks, a *plan* is specified by a choice of "low-stress," "high-stress," or "none" for each of the n weeks, with the property that it "high-stress" is chosen for week $i > 1$, then "none" has to be chosen for week $i - 1$. (It's okay to choose a high-stress job in week 1.) The *value* of the plan is determined in the natural way: for each i, you add l_i to the value if you choose "low-stress" in week i, and you add h_i to the value if you choose "high-stress" in week i. (You add 0 if you choose "none" in week i.)

The problem. Given sets of values $l_1, l_2, \ldots, l_n$ and $h_1, h_2, \ldots, h_n$, find a plan of maximum value.

(a) Show that the following algorithm does not correctly solve this problem, by giving an instance on which it does not return return the correct answer.

---

for iterations $i = 1$ to n **do**
  if $h_{i+1} > l_i + l_{i+1}$ **then**
    Output "Choose no job in week i"
    Output "Choose a high-stress job in week $i + 1$"
    Continue with iteration $i + 2$
  else
    Output "Choose a low-stress job in week i"
    Continue with iteration $i + 1$
  end if
end for

To avoid problems with overflowing array bounds, we define $h_i = l_i = 0$ when $i > n$

In your example, say what the correct answer is and also what the above algorithm finds.

(b) Give an efficient algorithm that takes values for $l_1, l_2, \ldots, l_n$ and $h_1, h_2, \ldots, h_n$ and finds a plan of maximum value.

part a: 15 points

part b: 30 points

3. (15 points) **Interview Question**

Given a number n and 3 different operations that you can perform:

- $n = n/2$
- $n = n/3$
- $n = n - 1$

Give an algorithm that reaches $n == 1$ in the minimum number of operations for any given starting n.