CS 577: Introduction to Algorithms	Homework 5
Out: 02/24/16	Due: 03/02/16

Ground Rules

- See HW1.
- Both problems will be graded.

Problems

1. Imagine you are a summer intern at a large financial services company. To test your acumen, your boss gives you the following job. The market offers n investments; the i-th one can be bought for c_i and sold (30 days later) for p_i. We assume that $p_i \geq c_i > 0$ for all i. You must buy exactly r of these, and your bonus for this exercise will not reflect the net profit, but rather the return ratio

$$\frac{\sum_{i \in S} p_i}{\sum_{i \in S} c_i}.$$

Your goal is to find an optimal set S of size r that maximizes this ratio.

(a) Show that a greedy algorithm that selects the r investments with the largest p_i/c_i ratio does not always produce the optimal solution. (Give a counterexample.)

(b) Your colleague claims that you can use dynamic programming to solve this problem based on the following principle of optimality. Let OPT(i, r) denote an optimal subset of options $\{i, \cdots, n\}$ of size r.

Claimed principle of optimality: OPT$(1, r)$ is either OPT$(2, r)$ or $\{1\} \cup$ OPT$(2, r-1)$

Give a counterexample showing that this property is not true.

(c) Suppose that you are told that the optimal solution achieves a ratio y. Give a greedy algorithm for finding the set of size r that achieves that ratio. If y is not the correct ratio, can your algorithm determine whether this guess is too large or too small?

(d) Use your solution to part (c) to develop a polynomial time algorithm for solving this problem. Give a brief argument of correctness, and analyze the running time of your algorithm.

2. A complex linear structure is to be assembled out of n smaller pieces. We will think of each piece as an interval $[a, b]$. The joining operation takes $[a, b]$ and $[b, c]$ and produces $[a, c]$. After joining, each subpart must be tested. Assume that the cost to test $[u, v]$ is given by $f(u, v) > 0$.

Different assembly orders potentially have different total testing cost. For example, suppose that we have three pieces corresponding to intervals $[1, 2], [2, 3]$, and $[3, 4]$, and the cost of testing is given by: $f(1, 3) = 3$, $f(2, 4) = 1$, and $f(1, 4) = 5$. Then assembling the first and second pieces first and then joining them with the third has a total testing cost of $f(1, 3) + f(1, 4) = 8$, whereas assembling the second and third pieces first and then joining them with the first has a total testing cost of $f(2, 4) + f(1, 4) = 6$. Therefore, the second assembly order is preferable.

Design an $O(n^2)$ algorithm to find an optimal (least total testing cost) assembly order. Give a brief argument of correctness, and analyze the running time.

Hint: Use dynamic programming. What should the principle of optimality say in this case?