COSC 6111 Advanced Design and Analysis of Algorithms

Jeff Edmonds

Assignment: Dynamic Programming

First Person:	Second Person:
Family Name:	Family Name:
Given Name:	Given Name:
Student #:	Student #:
Email:	Email:

Problem Name	If Done Old Mark	Check if to be Marked	New Mark
The Stock Buying Problem			
1 Bird-Friend Method ($\mathcal{O}(T \cdot T|S|)$ time)			
2 Shortest Path Method ($\mathcal{O}(T \cdot T|S|)$ time)			
3 Shortest Path Method ($\mathcal{O}(T|S| \cdot |S|)$ time)			
4 Shortest Path Method ($\mathcal{O}(T \cdot |S|)$ time)			

COSC 6111 Advanced Design and Analysis of Algorithms
Jeff Edmonds
Assignment: Dynamic Programming

1. Stock Market Prices You are very lucky to have a time machine bring you the value each day of a set of stocks. The input instance to your problem consists of $I = \langle T, S, Price\rangle$, where T is an integer indicating your last day to be in the market, S is the set of $|S|$ stocks that you consider, and $Price$ is a table such that $Price(t, s)$ gives the price of buying one share of stock s on day t. Buying stocks costs an overhead of 3%. Hence, if you buy p dollars worth of stock s on day t, then you can sell them on day t' for $p \cdot (1 - 0.03) \cdot \frac{Price(t',s)}{Price(t,s)}$. You have one dollar on day 1, can buy the same stock many times, and must sell all your stock on day T. You will need to determine how you should buy and sell to maximize your profits.

   Because you know exactly what the stocks will do, there is no advantage in owning more than one stock at a time. To make the problem easier, assume that at each point time there is at least one stock not going down and hence at each point in time you alway own exactly one stock. A solution will be viewed a list of what you buy and when. More formally, a solution is a sequence of pairs $\langle t_i, s_i\rangle$, meaning that stock s_i is bought on day t_i and sold on day t_{i+1}. (Here $i \geq 1$, $t_1 = 1$ and $t_{last+1} = T$.) For example, the solution $\langle\langle 1, 4\rangle, \langle 10, 8\rangle, \langle 19, 2\rangle\rangle$ means that on day 1 you put your one dollar into the 4^{th} stock, on day 10 you sell all of this stock and buy the 8^{th} stock, on day 19 you sell and buy the 2^{nd} stock, and finally on day T you sell this last stock. The value of this solution is $\Pi_i \left[(1 - 0.03) \cdot \frac{Price(t_{i+1},s_i)}{Price(t_i,s_i)}\right] = 1 \cdot (1 - 0.03) \cdot \frac{Price(10,4)}{Price(1,4)} \cdot (1 - 0.03) \cdot \frac{Price(19,8)}{Price(10,8)} \cdot (1 - 0.03) \cdot \frac{Price(T,2)}{Price(19,2)}$. (Note that the symbol Π_i works the same as $\sum_i$ except for product.)

   Design for a $\mathcal{O}(T \cdot T|S|)$ time dynamic programming algorithm for this stock buying problem. Hint: Ask the bird for the last "object" in the solution.

2. Read *Designing a Dynamic Programming Algorithm via the "Where are you" Abstraction and/or "a Reduction to Shortest Paths" Abstraction.* Consider the $\mathcal{O}(T \cdot T|S|)$ time dynamic programming algorithm hopefully developed using the Bird-Friend method in Question 1. Describe the graph G_I that corresponds to this algorithm. What are the nodes, edges, and weights in the graph G_I? Then describe how these nodes correspond to states that you might be in during your constructing of a solution of when and what to buy. Hint: it is ok for the weight of a path through G_I to be defined as the product of the weights of the edges instead of the sum - or you can take the logarithm of all the weights so that this product becomes a sum.

3. Question 1 developed a $\mathcal{O}(T \cdot T|S|)$ time dynamic programming algorithm for the stock buying problem and Question 2 viewed this algorithm as a leveled graph G_I. The goal of this question is to develop a $\mathcal{O}(T|S| \cdot |S|)$ time dynamic programming algorithm for the same problem. Which is faster depends on whether there are more stocks $|S|$ or more time steps T. To help develop this algorithm, think both of the bird-friend approach and of the states/nodes in a graph G_I approach. The document *Designing a Dynamic Programming Algorithm via the "Where are you" Abstraction and/or "a Reduction to Shortest Paths" Abstraction* gives you four types of question and two types of nodes. Some combination of these works. Give the bird-friend algorithm, the graph G_I, and the code for this $\mathcal{O}(T|S| \cdot |S|)$ time dynamic programming algorithm for the stock buying problem.

4. I was very excited to come up with an optimally faster dynamic program for the stock buying problem. One interesting thing about this algorithm is that it uses two intertwining recurrence relations. The running time is $\mathcal{O}(T \cdot |S|)$ which is better than the time $\mathcal{O}(T \cdot T|S|)$ from Questions 1 and the time $\mathcal{O}(T|S| \cdot |S|)$ from Questions 3. No algorithm solving the stocks problem could be faster than $\mathcal{O}(T \cdot |S|)$, because this is the time needed to simply look at the input instance *price*. This algorithm is a funny merging of the previous to algorithms. Again, the algorithm is some combination of the four types of question and two types of nodes. Give the bird-friend algorithm, the graph G_I, and the code for this $\mathcal{O}(T|S|)$ time dynamic programming algorithm for the stock buying problem.