CS29002 Algorithms Laboratory

Assignment No: 10

Last date of submission: 19–October–2016

Mr. Intermediario buys items (of a single type) from a manufacturer and sells those to a distributor. Each item has a buying rate and a selling rate (assumed to be positive integers) on a day. These rates are available to Mr. Intermediario for n consecutive days in two arrays $B[\,]$ and $S[\,]$. Only a whole number of items can be bought or sold. Mr. Intermediario has an initial capital C (again a positive integer). If the buying rate is b on a day, he can buy (at most) $\lfloor C/b \rfloor$ items. He cannot sell items before buying them. Buying and selling on the same day are allowed. *All* the items bought on Day i must be sold on Day j for some j satisfying $0 \leqslant i \leqslant j \leqslant n-1$. Write a program to help Mr. Intermediario achieve the maximum possible profit.

Part 1: Single transaction

Suppose that Mr. Intermediario makes exactly one buying and one selling. He has to choose the days i and j such that the quantity $\left\lfloor \frac{C}{B[i]} \right\rfloor \Big(S[j] - B[i]\Big)$ is maximized. An exhaustive search over all pairs (i, j) satisfying $0 \leqslant i \leqslant j \leqslant n-1$ takes $\Theta(n^2)$ running time.

We can do much better using a divide-and-conquer strategy. Break the set of days into two equal-sized halves. Recursively compute the optimal profits LOPT and ROPT for the left and right halves. It is also allowed that items are bought in the first half, and sold in the second half. The optimal way to do this is to find Day i in the first half on which the buying rate is minimum, and to find Day j in the second half on which the selling rate is maximum. Let LROPT denote the profit for these choices of i and j. Then, the maximum profit of Mr. Intermediario is max(LOPT, ROPT, LROPT). This algorithm has a running time given by $T(n) = 2T(n/2) + \Theta(n)$, where the $\Theta(n)$ effort is associated with locating the minimum buying rate and the maximum selling rate in the two halves. By the master theorem, $T(n) = \Theta(n \log n)$. The space requirement is $\Theta(\log n)$ since the depth of recursion is $\log_2 n$.

- Write a function *singletrans1*(B, S, n, C) to implement this divide-and-conquer algorithm.
- Write a function *singletrans2*(B, S, n, C) to implement a modification of the above divide-and-conquer algorithm, achieving a running time of $\Theta(n)$. Since the recursion depth would not change, the space requirement will remain $\Theta(\log n)$.

Part 2: Multiple transactions

Now, assume that Mr. Intermediario can make buying and selling multiple times. Let t be the number of transactions made ($t = 0$ is also allowed). The task of Mr. Intermediario is to choose buying days $i_1, i_2, \ldots, i_t$ and selling days $j_1, j_2, \ldots, j_t$ satisfying $0 \leqslant i_1 \leqslant j_1 < i_2 \leqslant j_2 < i_3 \leqslant j_3 < \cdots < i_t \leqslant j_t \leqslant n-1$. The condition $j_k < i_{k+1}$ indicates that a new transaction cannot be started on a day when some items are sold. However, transactions with $i_k = j_k$ are allowed. With transactions, the available capital of Mr. Intermediario changes. He uses his available capital to buy the maximum possible number of items (integral numbers only). Finally, *all* the items bought on Day i_k must be sold on Day j_k.

- Write an $O(n^2)$-time function *multitrans*(B, S, n, C) to maximize the final profit of Mr. Intermediario in the case of multiple transactions. Your function may use $O(n)$ additional space. Take a dynamic-programming approach. For $i = 0, 1, 2, \ldots, n-1$ (in that sequence), iteratively compute the maximum capital that Mr. Intermediario can have at the end of Day i.

The *main*() function

- Read n, the arrays $B[\,]$ and $S[\,]$, and the initial capital C from the user.
- Call *singletrans1* to print an optimal single transaction.
- Call *singletrans2* to print an optimal single transaction.
- Call *multitrans* to print a sequence of transactions maximizing the total profit.

  Present the outputs in the following format.

Sample output

```
+++ n = 10
+++ Buying prices  : 10 15 16  9 25  6  5 18  5 21
+++ Selling prices : 24 13  8 12  9 21  7 21  6 14
+++ C = 1000

+++ Single transaction: O(n log n) time

    Buying  date =  6, Buying  rate =  5
    Selling date =  7, Selling rate = 21

    Maximum profit = 3200

+++ Single transaction: O(n) time

    Buying  date =  6, Buying  rate =  5
    Selling date =  7, Selling rate = 21

    Maximum profit = 3200

+++ Multiple transactions

    Initial capital = 1000

    Buying  date =  0, Buying  rate = 10
    Selling date =  0, Selling rate = 24
    Current capital = 2400

    Buying  date =  3, Buying  rate =  9
    Selling date =  3, Selling rate = 12
    Current capital = 3198

    Buying  date =  5, Buying  rate =  6
    Selling date =  5, Selling rate = 21
    Current capital = 11193

    Buying  date =  6, Buying  rate =  5
    Selling date =  7, Selling rate = 21
    Current capital = 47001

    Buying  date =  8, Buying  rate =  5
    Selling date =  9, Selling rate = 14
    Current capital = 131601

    Maximum profit = 130601
```

Submit a single C/C++ source file. Do not use global/static variables.