CS 5592: Design and Analysis of Algorithms
Homework 2

Author: Hayden McParlane

(1) There are n houses located on a west-to-east street. H[i] (meters), $1 \leq i \leq n$, is the distance from the west end of the street to the ith house. You may assume that H[1] < H[2] < H[3] < ... < H[n]. There is no post office on the street. We plan to build several post offices on the street such that any house can reach a post office within 100 meters. Please design an $\mathcal{O}(n)$ algorithm to compute the locations for the post offices, P[j] (meters), $1 \leq j \leq m$, where P[j] is the distance from the west end of the street to the jth post office. Make sure that the number of post offices, m, is minimized.

Algorithm 1 Algorithm used to place the minimal number of post offices such that each house in H will have a post office at most 100 meters away.

```
function PlacePostOffices(H, n)
    $P \leftarrow \emptyset$

    Append($P, H[1] + 100$)
    $j \leftarrow 1$
    for i from 2 to n do
        if |H[i] - P[j]| > 100 then
            Append($P, H[i] + 100$)
            $j \leftarrow j + 1$
        end if
    end for

    return P
end function
```

(2) Suppose we drive a pickup truck from city A to city B. Along the highway we will pass through n apple markets labeled with 1, 2, 3, ..., n, where you can buy or sell apples. City A and city B also have an apple market each. From a customer point of view, the buying price B[i] and selling price S[i] (dollar per pound) at market i are known. Now, we will stop at one of the stationsto buy apples and then stop and another station to sell apples. Please design an $\mathcal{O}(n)$ greedy algorithm to find market i to buy apples and market j$\geq$ i to sell apples such that the profit will be maximized. We assume that it would be too costly and forbidden to drive backward. You need to do exactly one trade even if the profit is negative.

Algorithm 2 Greedy algorithm used maximize profit for the apple buy/sell problem above. Note that in certain instances this algorithm may be suboptimal, a quality common to greedy algorithms.

```
function MaxProfit(B, S, n)
    buy ← 0
    sell ← 0
    for i from 1 to n + 1 do
        if B[buy] > B[i] then
            buy ← i
            sell ← i
        else
            if S[sell] < S[i] then
                sell ← i
            end if
        end if
    end for

    return (buy, sell)
end function
```

③ Re-consider the following problem in quiz 1: Given a sequence of n real numbers stored in an array, A[1], A[2], A[3], ..., A[n], we wish to find two numbers A[i] and A[j], where i < j, such that A[i] $\leq$ A[j] and their sum is the largest. If no such two numbers exist, report -∞. This time, please design an $\mathcal{O}(n)$ greedy algorithm to solve this problem.

Algorithm 3 Greedy algorithm solving the problem above. It's important to note that a quality of greedy algorithms is possible suboptimal performance. This solution may perform suboptimally whereas the divide-and-conquer algorithm presented previously would not. However, this algorithm performs better. It is a trade-off.

```
function LargestSum(A, n)
    $max \leftarrow -\infty$
    $submax \leftarrow -\infty$

    for k from 1 to n do
        if max $\leq$ A[k] then
            $subMax \leftarrow max$
            $max \leftarrow A[k]$
        end if
    end for

    if max equals -$\infty$ or submax equals -$\infty$ then
        return -$\infty$
    else
        return $max$ and $submax$
    end if
end function
```