CPSC 413 FALL 2020 PROBLEM SET 4

Yifan Chen 30073072

November 2020

1

(a) Suppose the coin change algorithm produces the solution $S = \{c_1, c_2, ...c_n\}$, and there is an optimal solution $O = \{o_1, o_2, ...o_m\}$, which has also been sorted by coin size in descending order. Suppose S and O differed. Assume S and O are different, there must be a first element of index d, where S and O differ, such that $O = \{s_1, s_2, ..s_{d-1}, o_d, o_{d+1}..., o_m\}$ Because the coin change algorithm would always choose the largest coin possible for s_d, o_d must be a different, smaller coin. Since the sums of the two solutions must be the same, and because O has been sorted in descending order, array O cannot catch up to solution S in the same number of coins, thus $m > n$, and O has more coins. But O is an optimal solution, and m is the lowest possible number of coins, thus by contradiction, our assumption that S and O are different must be wrong, and S = O.

2

(a)

1. Precondition: An Array of n houses $h = \{h_1, h_2, ..., h_i\}$ where each element represents a houses' distance in miles from the western endpoint of the road.
2. Postcondition: An array of m towers $t = \{t_1, t_2, ...t_m\}$, where each element represents a towers' distance in miles from the western endpoint of the road, such that any individual house is not more then 4 miles away from any tower, and the number of towers is minimized.

(b)
let t be the array of towers, h be the array of houses
t[0] = h[0] + 4 //place first tower max distance, 4 miles from house
CloseTower = 0
for i = 1; i < h; i++ do
if h[i] > T[CloseTower] + 4 then

Closetower ++
T[CloseTower] = h[i] + 4
endif
endfor
return T[] of size CloseTower.
(C). The algorithm will start by placing a tower the maximum distance of 4 miles from the first house. This way, the first house is covered, and the tower will cover as many additional houses as possible. When a house is placed outside of this towers range, another tower will be placed 4 miles away from the this house, again ensuring maximum coverage. The algorithm continues until no houses remain.
(D). No, it is not the only possible solution. For example let there be only one house at a positioned 5 miles from the western endpoint. Our algorithm would then a single tower at distance 9. This is a valid and optimal solution as the house is covered by the tower, and the number of towers is minimized, however if the tower was placed at any position p in range of 5 +/- 4 the solution would be equally valid.
(E). Suppose our algorithm produces a solution $S = \{s_1, s_2, .., s_n\}$, and there is an optimal solution $O = \{o_1, o_2, ..., o_m\}$. Suppose S and O differed. Let d be the first index where S and O differ, such that $O = \{s_1, s_2, ...s_{d-1}, o_d, o_{d-1}, ..., o_m\}$ let h be a house at index $s_d - 4$. Because our algorithm places towers 4 miles after the next uncovered house, and because both solutions are the same prior to index d, neither solution has a tower to cover h. Then o_d and s_d must both cover h. Because s_d is at p + 4, we know o_d must be $< p + 4$, or it would not cover h. Then s_d would cover all houses that o_d would be in range of. We could then exchange s_d and o_d and O would still be optimal, as it would still have the same number of houses covered with the same number of towers. Since this process can be repeated for any towers in O and A, O = A, and A thus must be optimal.
(F). Runtime would be $\theta(n)$. The algorithm would loop through the for loop n-1 times for each index of array house. Any operations inside the loop takes constant time. initializing the algorithm is also O(1).

3

(A).

1. Precondition: An array of n books, $b = t_{b1}, t_{b2}, t_{3b}..., t_{bn}$, which that each element t in n represents the number of days it takes to read the book.

2. A sorted array of n books $b = t_{b1}, t_{b2}, t_{3b}..., t_{bn}$ such that t_{b1} ¡= t_{b2} ¡=...t_{bn}.

(B). t_2, t_3, t_1
(C).
1. Sort the input array by ascending time
2. output the sorted array.

(D) Consider the input of any set of books. The time taken to read those books would be t regardless of the order of the books, meaning you will be charged t days. Finishing the shortest book first, the fee is reduces as fast as possible, meaning that in that earliest in those t days, your fee is reduced, minimizing the late fee.
(E)
(F) The algorithm consists only of a sort function, which takes $\theta(nlogn)$ runtime.