Kaleb Kim
12/1/24

CSDS 310 Assignment 4 v2

Note: Arrays are zero-indexed.

Problem 1

a) ✓

b) **Counterexample**: Consider activities $a_1 = [1, 4], a_2 = [4, 7]$, and $a_3 = [4, 5]$. The greedy solution is $\{a_3\}$, since a_3 has the shortest duration. However, the optimal solution is $\{a_1, a_2\}$.

c) **Counterexample**: Consider activities $a_1 = [1, 3], a_2 = [3, 5], a_3 = [5, 7], a_4 = a_5 = [2, 4], a_6 = a_7 = [4, 6]$. Here, the greedy solution does the following:

- a_1 or a_3 is selected since each have the fewest overlaps of 2.
- If a_1 is selected, then a_6 or a_7 is selected since each have the fewest overlaps of 3. If a_3 is selected, then a_4 or a_5 is selected by the same logic.
- No other activities can be run. The algorithm terminates, which means that the greedy solution only generates sets of length two. These sets are $\{a_1, a_6\}, \{a_1, a_7\}, \{a_3, a_4\}$, and $\{a_3, a_5\}$.

However, the optimal solution is $\{a_1, a_2, a_3\}$.

Problem 2

Pseudocode

```
procedure MAX_PROFIT(A, B, n):
    r-quicksort A
    r-quicksort B
    profit ← 1
    for $0 \le i < n$:
        profit ← profit × $A[i]^{B[i]}$
    return profit
```

Proof

Let $A = \{a_1, ..., a_n\}$ and $B = \{b_1, ..., b_n\}$. Let p be the payout generated by the greedy algorithm, such that $p = a_0^{b_0} \cdot a_1^{b_1} \cdot ... \cdot a_{n-1}^{b_{n-1}} \cdot a_n^{b_n}$.

In the greedy algorithm, we have A and B sorted in ascending order. For an exchange argument, suppose A and B are not sorted in ascending order. Let $i, j \in \mathbb{Z}$ such that $i \neq j, 0 \le i \le n$ and $0 \le j \le n$. Our supposition means that there exists product pairs of general solution $p' = a_i^{b_j} \cdot a_j^{b_i}$. Since A and B are sorted in our greedy solution, resulting in matching pairs, then we have $p = a_i^{b_i} \cdot a_j^{b_j}$.

To prove that the algorithm yields a product that is at least as optimal as the general solution, we must prove that the ratio $\frac{p}{p'} \ge 1$. We have:

$$\frac{p}{p'} = \frac{a_i^{b_i} \cdot a_j^{b_j}}{a_i^{b_j} \cdot a_j^{b_i}} = a_i^{b_i - b_j} a_j^{b_j - b_i} = \left(\frac{a_i}{a_j}\right)^{b_i - b_j}$$

Since A and B are sorted in ascending order, we have $a_i \ge a_j$ and $b_i \ge b_j$. Then, $\frac{a_i}{a_j} \ge 1$ and $b_i - b_j \ge 0$. This means that $\frac{p}{p'} \ge 1$, proving that $a_i^{b_i} \cdot a_j^{b_j} \ge a_i^{b_j} \cdot a_j^{b_i}$. Thus, by swapping for any pairs, swapping to get a pair product for $a_i^{b_i} \cdot a_j^{b_j}$ increases the total payout or it remains the same as the general solution.

Runtime

Randomized quick sort takes $O(n \log n)$ time in the worst case. It also sorts in place, using $O(1)$ extra space. The for loop runs for $\Theta(n)$ time. Thus, we have:

$$\text{Time complexity: } O(n \log n)$$
$$\text{Space complexity: } O(1)$$

Problem 3

Pseudocode

```
procedure MINIMIZE_MAX_DELAY(t, d, n):
    I ← indices array from randomized quicksort based on d
    t ← reconstruct t in order of I
    d ← reconstruct d in order of I
    start ← 0
    delays ← 0 array of n elements
    for $0 \le i < n$:
        fin ← start + t[i]
        delay ← max(0, fin − d[i])
        delays[i] = delay
        start ← fin
    return max(delays)
```

Proof

Let $t = \{t_1, ..., t_n\}$ and $d = \{d_1, ..., d_n\}$ represent the times and deadlines of activities respectively. Let D be the total delay generated by the greedy algorithm, where activities are scheduled in non-decreasing order of their deadlines.

In the greedy algorithm, we sort t and d by d in ascending order. For the sake of an exchange argument, suppose t and d are not scheduled in ascending order of deadlines. Let $i, j \in \mathbb{Z}$ such that $i \le j, 0 \le i \le n,$ and $0 \le j \le n$. Our supposition means that there exist scheduled activities with delays $D' = \max(\Delta_{i'}, \Delta_{j'})$, such that $\Delta_{i'} = f_i - d_j, \Delta_{j'} = f_j - d_i$. In the greedy solution, the delays are $\Delta_i = f_i - d_i$ and $\Delta_j = f_j - d_j$.

To prove that the algorithm yields a delay that is at most as large as the general solution, we must prove that swapping t_i and t_j decreases the maximum delay or keeps it the same. After swapping, the delays become:

$$\Delta_{i'} = f_i - d_j, \Delta_{j'} = f_j - d_i.$$

Then, we have:

$$\Delta_{j'} - \Delta_j = (f_j - d_i) - (f_j - d_j) = d_j - d_i.$$

Since $d_i > d_j$ in the unsorted solution, we have $d_j - d_i < 0$, reducing the delay for activity j.

Similarly, for activity i:

$$\Delta_{i'} - \Delta_i = (f_i - d_j) - (f_i - d_i) = d_i - d_j.$$

Since $d_i > d_j$, the delay for activity i increases.

However, because $d_i > d_j$, swapping ensures that the activity with the stricter deadline (smaller d) is scheduled earlier, minimizing the overall maximum delay. Repeating this argument for all misordered pairs ensures that sorting by deadlines produces the smallest possible maximum delay.

Thus, by swapping to schedule activities in ascending order of deadlines, our greedy algorithm minimizes the maximum delay.

Problem 4b

This problem is similar to binary representation of base 10.

We must prove that if the coin denominations are powers of 2, then this greedy choice leads to the optimal solution. Let n be the amount to make, and let the coin denominations be $C = \{1, 2, ..., 2^{k-1}\}$, with $k \leq \lceil \log_2(n) \rceil$. Let G be greedy solution which selects coins $\{g_1, g_2, ..., g_m\}$, where each g_i is the largest coin such that $g_i \leq n_i$.

Suppose G is not optimal for contradiction, in which there exists an optimal solution O with $t < m$. Let the first mismatch between G and O occur at position i, where $g_i \neq o_i$. Since g_i is the largest coin $\leq n_i$, we have $g_i > o_i$. Because the denominations are powers of 2, g_i must be a multiple of o_i:

$$g_i = 2^k, o_i = 2^{k-1} \therefore g_i = 2o_i.$$

To compensate for o_i, O must include at least two coins of value o_i to replace g_i, so we have $2o_i = g_i$. This implies O uses more coins than G, contradicting the assumption that O is optimal. Thus, G is the most optimal.