COMP4500
Assignment 2

Maxwell Bo (43926871)

October 19, 2018

Question 1

(a)

```
from typing import List

def min_order(H: int, requests: List[int]) -> List[int]:
    midpoint = (min(requests) + max(requests)) / 2

    if H <= midpoint:
        return sorted(requests)
    else:
        return sorted(requests, reverse=True)

def min_time(H: int, ordering: List[int]) -> int:
    seek_to_start = abs(ordering[0] - H)
    range = max(ordering) - min(ordering)

    return seek_to_start + range

def greedy_algo(H: int, requests: List[int]):
    minOrder = min_order(H, requests)
    minTime = min_time(H, minOrder)

    return minTime, minOrder

print(greedy_algo(200, [40, 180, 300, 10])) # (390, [300, 180, 40, 10])
```

a.

Consider some arbitrary list of track requests.

Assume $H \leqslant \min(\mathit{requests})$. The optimal ordering is such that *requests* is sorted in ascending order, moving to $\min(\mathit{requests})$, sweeping across all the intermediate tracks until it lands at $\max(\mathit{requests})$. The head had to move $(\max(\mathit{requests}) - \min(\mathit{requests})) + (\min(\mathit{requests}) - H)$ tracks.

Assume $\max(\mathit{requests}) \leqslant H$. The optimal ordering is such that *requests* is sorted in descending order, moving to $\max(\mathit{requests})$, sweeping across all the intermediate tracks until it lands at $\min(\mathit{requests})$. The head had to move $(\max(\mathit{requests}) - \min(\mathit{requests})) + (H - \max(\mathit{requests}))$ tracks.

Let the $\mathit{midpoint}(\mathit{requests}) = (\min(\mathit{requests}) + \max(\mathit{requests}))/2$. We want to minimize the distance we move to the start or end of the sorted *requests* before we start our sweep.

Assume $\min(requests) \leqslant H \leqslant midpoint$. The optimal ordering is such that $requests$ is sorted in ascending order, moving to $\min(requests)$, sweeping across all the intermediate tracks until it lands at $\max(requests)$. The head had to move $(\max(requests) - \min(requests)) + (H - \min(requests))$ tracks.

Assume $midpoint \leqslant H \leqslant \max(requests)$. The optimal ordering is such that $requests$ is sorted in descending order, moving to $\max(requests)$, sweeping across all the intermediate tracks until it lands at $\min(requests)$. The head had to move $(\max(requests) - \min(requests)) + (\max(requests) - H)$ tracks.

Thus, our optimal cost is always the range of the requests, plus the the distance travelled to get to either the minimum or the maximum of the requests, whichevers smallest.

Question 2

b.

The recursive procedure forms a tree with branching factor 3, with each branch created by either *fully rebooting*, *partially rebooting* or opting not to reboot.

At the root, we have 1 node. On the next level, we have 3, and on the next level, we have 9. This is the geometric series.

To calculate the total number of nodes in the tree after k depth, we sum the geometric series.

$$
\begin{aligned}
\sum_{i=0}^{k} 3^i &= 1 + 3 + 3^2 + \ldots + 3^k \\
&= \frac{3^{k+1} - 1}{3 - 1} \\
&= \frac{3^{k+1} - 1}{2}
\end{aligned}
$$

This is $O(3^k)$.

d.

By peeking at the implementation of the bottom-up table construction procedure, we can see:

```
for (int d = lastDay; d >= 0; d--) {
    for (int i = 0; i <= d + 1; i++) {
        // add to table
    }
}
```

which is equivalent to

$$
\begin{aligned}
\sum_{d=0}^{k-1} \sum_{i=0}^{d} 1 &= \sum_{d=1}^{k} \sum_{i=1}^{d+1} 1 \\
&= \sum_{d=1}^{k} (d + 1) \\
&= \sum_{d=1}^{k} d + \sum_{d=1}^{k} 1 \\
&= (\sum_{d=1}^{k} d) + k \\
&= \frac{k(k+1)}{2} + k \\
&= \frac{k^2 + k}{2} + k \\
&= \frac{k^2 + 3k}{2}
\end{aligned}
$$

Therefore, we can see the upper-bound is $\Omega(k^2)$.