Assignment 2

Erik Norlin, CID: norliner

September 18, 2023

Problem 4

a)

An algorithm for this problem that runs in $O(n)$ would be an algorithm that touches each element once. Since the rooms are sorted in a descending manner we can utilize this. We start by computing the largest and smallest rooms added together, i.e., $s_i + s_j$ where $i = 0$ and $j = n$. If this is equal to s we are done. if it is too large we pick one room smaller than s_i (s_{i+1}) and keep the smallest room, add them together and check again. Instead if it is too small we pick one room larger than s_j (s_{j-1}) and keep the largest room, add them together and check again. We repeat this until we found $s_i + s_j = s$ or if $i > j$. When $i > j$ we start doing the same comparisons again which is unnecessary. When we have checked $s_i + s_j$ where $i = j$ we have done n operations and thus the algorithm runs in $O(n)$.

Pseudo code for this algorithms

```
r_1 = 0 # index of the first room to return
r_2 = 0 # index of the second room to return
i = 1
j = n
bool = false

while(!bool)
    if s_i + s_j > s
        i += 1
    else if s_i + s_j < s
        j -= 1
    else if s_i + s_j = s
        r_1 = i
        r_2 = j
        bool = true
    if i > j
        bool = true

return r_1, r_2
```

If no room size equal to s are found r_1 and r_2 return as zeros, otherwise they return indices of rooms that added together equal to the size of s.

b)

We prove that the greedy algorithm does not miss a solution by mistake by using proof by contradiction. Let R be the set of rooms sorted by size in descending order. Assume that the greedy algorithm misses a solution by mistake. This implies that there exists some solution s_i and s_j in R that added together equal to s. The greedy algorithm starts checking for compatible sizes of two rooms starting with s_i and s_j where $i = 1$ and $j = n$ (at each ends in R). If s_i and s_j together are larger than s it proceeds to check if s_{i+1} and s_j together equal to s. If together they are smaller than s it proceeds to check if s_i and s_{j-1} together equal to s. If s_i and s_j together equal to s a solution is found. This contradicts the assumption. Thus, the algorithm cannot miss a solution by mistake.