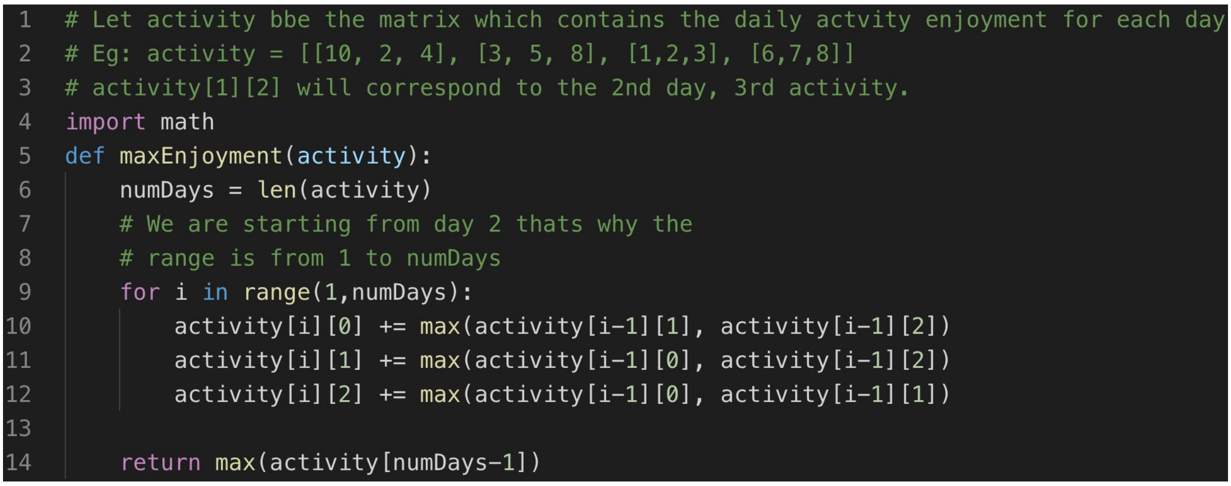
COMP3121 Homework Q3

Arth Sanskar Patel
z5228942

July 27, 2020

1 Answer

Here we can solve this question by solving the sub problem P(i, j) such that for any day $i \leq N$, we select and activity a_j such that it give us the maximum enjoyment. We can loop through the days starting from day 2 and for each day, we add the maximum enjoyment activity a_j out of the two remaining activities after removing the activity to which we are adding. For example, if we are choosing activity a_1, then we will add the maximum enjoyment out of activities a_2 and a_3 from the previous day. We are adding the other two activities to our chosen activity because we are required to not do the same activity two days in a row. In the end we can simply choose the maximum amount we are left with. Pseudo code for this is given below



```python
# Let activity bbe the matrix which contains the daily actvity enjoyment for each day
# Eg: activity = [[10, 2, 4], [3, 5, 8], [1,2,3], [6,7,8]]
# activity[1][2] will correspond to the 2nd day, 3rd activity.
import math
def maxEnjoyment(activity):
    numDays = len(activity)
    # We are starting from day 2 thats why the
    # range is from 1 to numDays
    for i in range(1,numDays):
        activity[i][0] += max(activity[i-1][1], activity[i-1][2])
        activity[i][1] += max(activity[i-1][0], activity[i-1][2])
        activity[i][2] += max(activity[i-1][0], activity[i-1][1])

    return max(activity[numDays-1])
```

We are only going through array of arrays (matrix) once, hence the time complexity for this solution is only $O(n)$. The python max function or any max function also runs in $O(n)$ time and hence the total complexity is $O(n)$. This question is similar to the leetcode question of minimising paint to paint houses so that no two houses in a row are of the same colour. It can be found here.