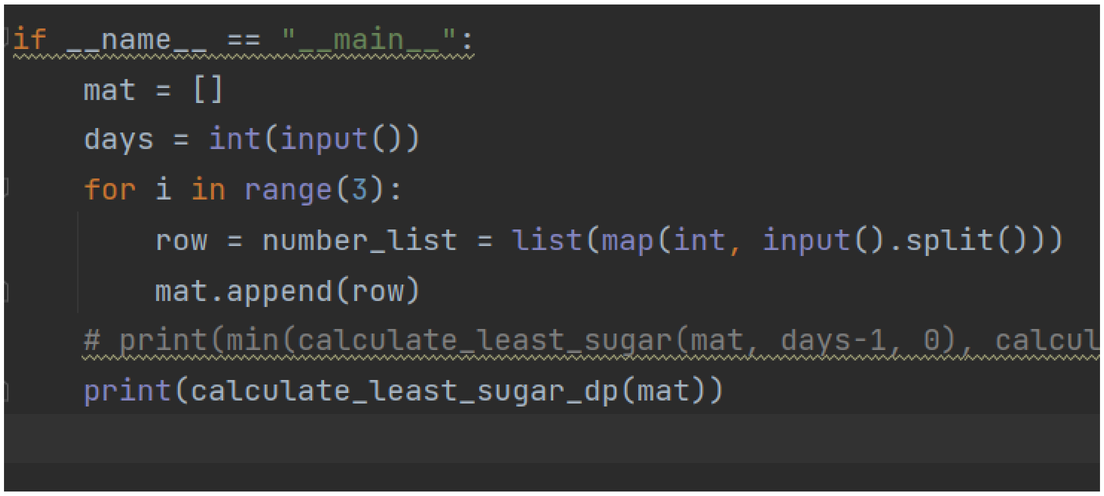
Final Project Report

Dynamic Programming

Main:

we have the input, which is the number of days. Then each how many calories each type of sweets provides.

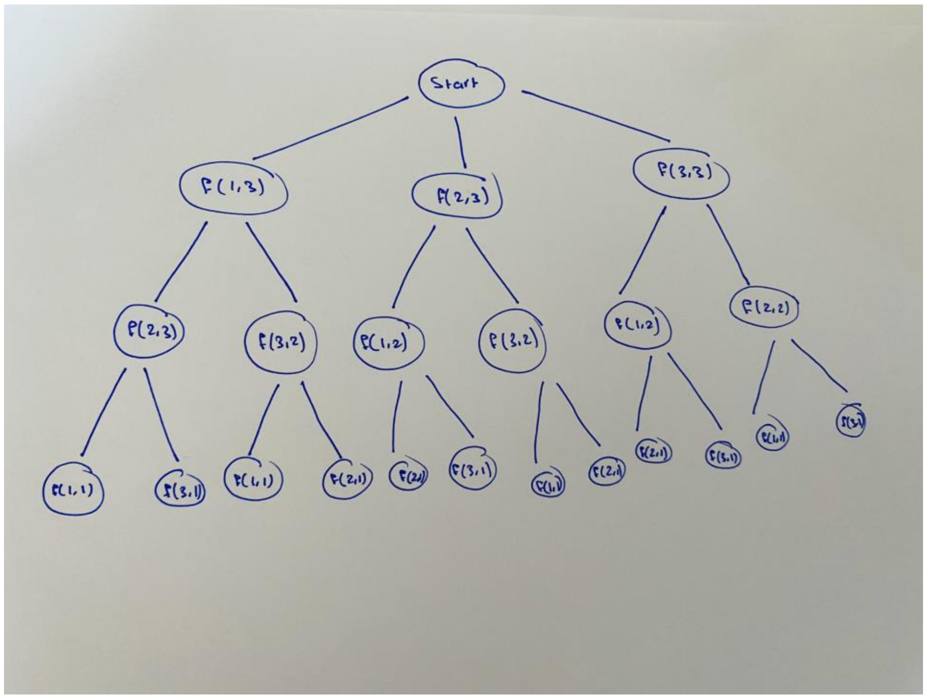
In the main we created a matrix, array of arrays in python. At first we read the days, then for loop in range of 3 because there are 3 types of sweets. Then we read one line and split the index, each number in a string then turn it into integer, then put it in a list, it returns the effect of the first sweet among 3 days. The returned row will be stored in a matrix.



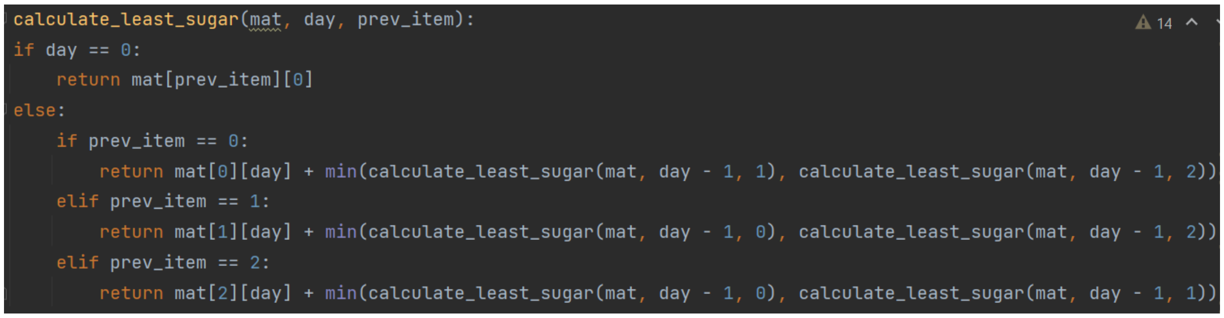
```python
if __name__ == "__main__":
    mat = []
    days = int(input())
    for i in range(3):
        row = number_list = list(map(int, input().split()))
        mat.append(row)
    # print(min(calculate_least_sugar(mat, days-1, 0), calcul
    print(calculate_least_sugar_dp(mat))
```

Divide and Conquer:

Let’s define it first by drawing the tree.



Let’s divide the method and explain each step:



```
calculate_least_sugar(mat, day, prev_item):
if day == 0:
    return mat[prev_item][0]
else:
    if prev_item == 0:
        return mat[0][day] + min(calculate_least_sugar(mat, day - 1, 1), calculate_least_sugar(mat, day - 1, 2))
    elif prev_item == 1:
        return mat[1][day] + min(calculate_least_sugar(mat, day - 1, 0), calculate_least_sugar(mat, day - 1, 2))
    elif prev_item == 2:
        return mat[2][day] + min(calculate_least_sugar(mat, day - 1, 0), calculate_least_sugar(mat, day - 1, 1))
```

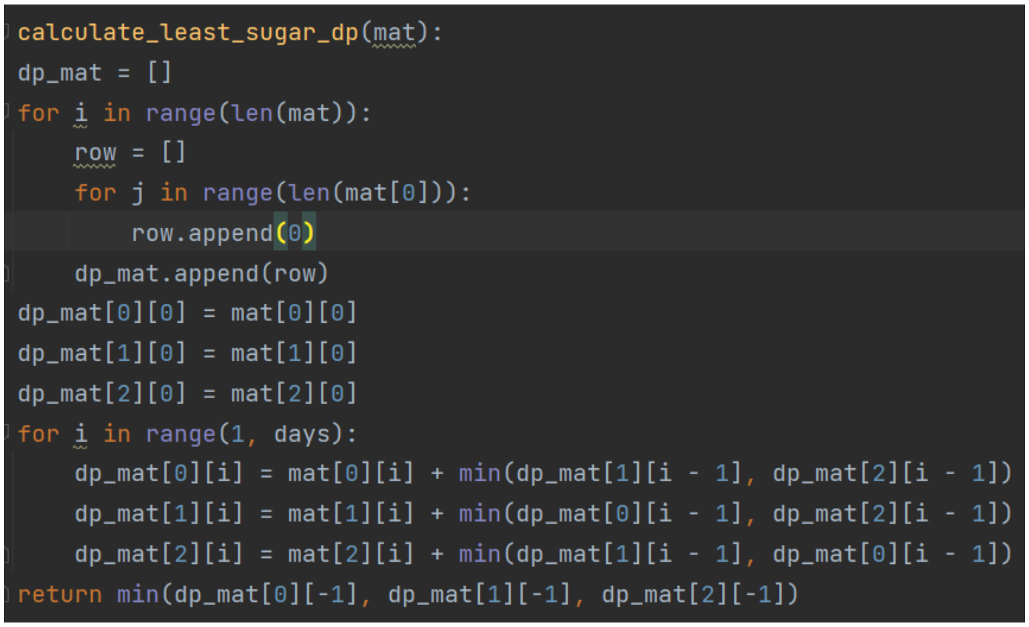
The parameters are 1) the matrix that stores the calories, the days, and the previous item since I go back through the tree.

If you are at your first day, (first point at the tree) return first day (3 or 6 or 9).

Else I am on the other nodes. I want to check the previous item. You are not allowed to choose zero (or first item) then you are allowed to choose the other items. So, I call the function twice with the parameters (the matrix, the previous day –that’s why i-1, day one). I call the function on the other item, then I choose the minimum sugar and I added to my current choice(0).

Then I repeated the same steps for the other days.

Dynamic Programming:



```
calculate_least_sugar_dp(mat):
dp_mat = []
for i in range(len(mat)):
    row = []
    for j in range(len(mat[0])):
        row.append(0)
    dp_mat.append(row)
dp_mat[0][0] = mat[0][0]
dp_mat[1][0] = mat[1][0]
dp_mat[2][0] = mat[2][0]
for i in range(1, days):
    dp_mat[0][i] = mat[0][i] + min(dp_mat[1][i - 1], dp_mat[2][i - 1])
    dp_mat[1][i] = mat[1][i] + min(dp_mat[0][i - 1], dp_mat[2][i - 1])
    dp_mat[2][i] = mat[2][i] + min(dp_mat[1][i - 1], dp_mat[0][i - 1])
return min(dp_mat[0][-1], dp_mat[1][-1], dp_mat[2][-1])
```

To solve the code from divide and conquer to dynamic programming, we need DP Matrix. We defined dp matrix, the number of rows is the number of days. I filled it with zero, there are 3 arrays in the matrix because we have 3 days. So the parameter of the method is the matrix.

First we will set the initial values.

0	0	0
0	0	0
0	0	0

If I am on 0 or first item I will take current item with minimum between the following 2 values.

3	6+6=12	9+5=14
6	3+3=6	5+5=10
8	2+3=5	6+6=12

$\longrightarrow$

And the return value is the minimum between(14,10,12)=10. The correct output.

Note: we moved through the matrix from left to right.

NOTE: python can return more than one thing so I returned the min value and dp matrix to use it in define_steps.

Define_steps:

We made array called choices, I added the initial values on a hash map, that contains the last value of each element. Least value is 10, from what elemnt we got 10? From the first choice (1).

Make a loop that go back, if the last element I chose is 0 so I check the sourse of 0 and I check the minimum, then cont. so, the output is 0 , 2 , 1. (3+2+5 = 10).

Dima Eid