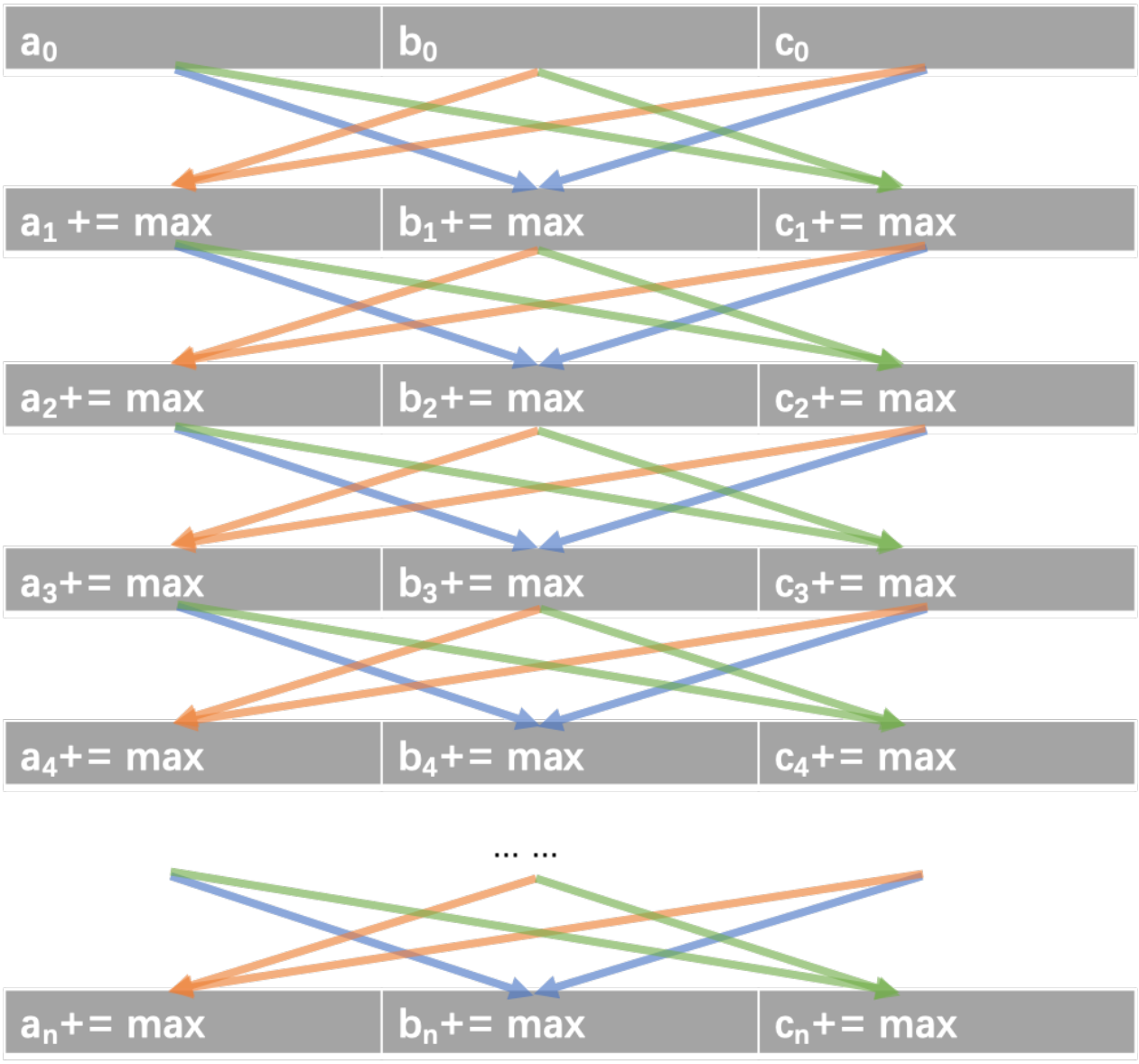
Yuan Gao z5239220 Q3

You are on vacation for N days at a resort that has three possible activities. For each day, for each activity, you've determined how much enjoyment you will get out of that activity. However, you are not allowed to do the same activity two days in a row. What is the maximum total enjoyment possible?

1. Formulated

> We can assume and create an activity `array[n][3]`:
> `n` present days and every day has 3 possible activities.
> For example:
>
> ```
> activities[4][3] = {
>    {1, 2, 4},
>    {1, 3, 2},
>    {2, 5, 1},
>    {3, 1, 2}
> };
> ```
>
> There are 4 days vacation.
> The element present how much enjoyment when people choice this activity.

2. Dynamic programming array



> As the picture shows, in i^{th}:
> When we pick a_i, `activities[i][0] += max(activities[i - 1][1], activities[i - 1][2]);`
> When we pick b_i, `activities[i][1] += max(activities[i - 1][0], activities[i - 1][2]);`
> When we pick c_i, `activities[i][2] += max(activities[i - 1][1], activities[i - 1][0]);`
> The result is : $max(a_n, b_n, c_n)$

3. Code

```
int letsHappy(vector<vector<int>> activities){
    if(activities.size() == 0) return 0;
    if(activities.size() == 1) return max(activities[0][0], max(activities[0]
[1], activities[0][2]));

    int n = activities.size();
    for(int i = 1; i < n; ++i){
        activities[i][0] += max(activities[i - 1][1], activities[i - 1][2]);
        activities[i][1] += max(activities[i - 1][0], activities[i - 1][2]);
        activities[i][2] += max(activities[i - 1][1], activities[i - 1][0]);
    }

    return max(activities[n - 1][0], max(activities[n - 1][1], activities[2]
[0]));
}
```

4. Time complexity

> Assume the vacation is n days
>
> > Traverse a activities array

> Hence, total time complexity is $O(n)$