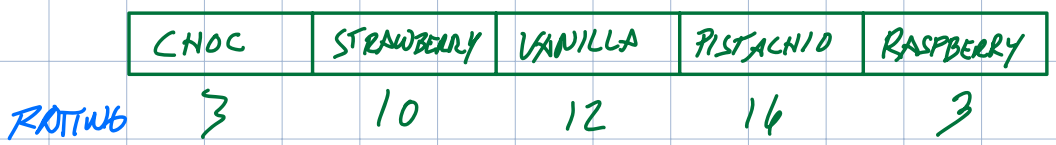
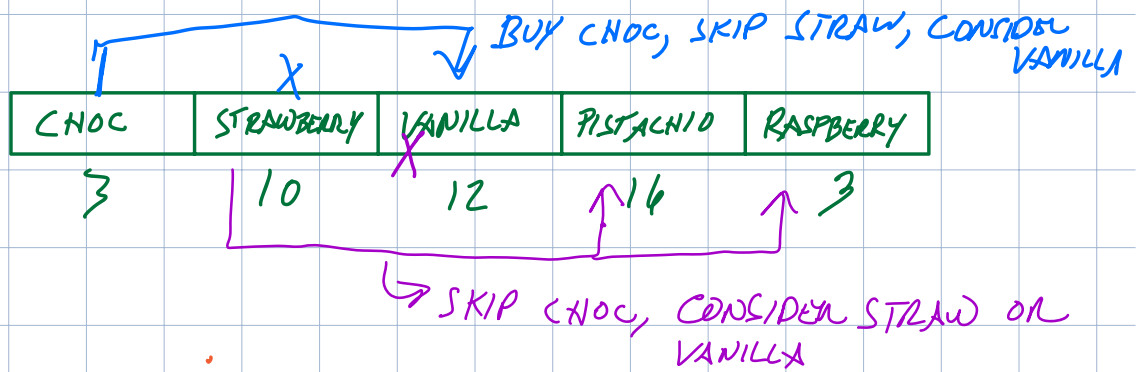
Goal: buy set of candies with the max total rating, but with a constraint:
- you can't get two candies that are adjacent to each other





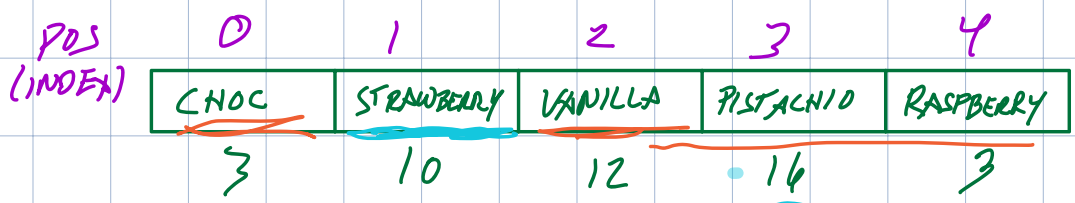
HOW TO FIND SET W/ MAX RATING?

OPTIONS...

CHOC + VANILLA + RASP => 3 + 12 + 3 = 18 ✓

STRAW + PIST => 10 + 16 = 26

CHOC + PIST => 3 + 16 = 19



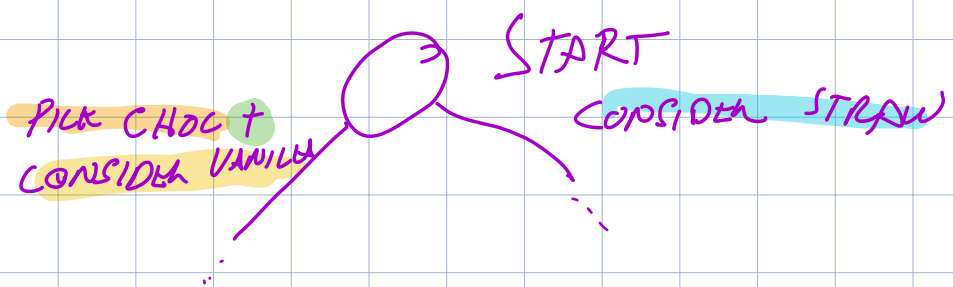
How would we do this generally? Can recursively build up a solution starting from the end

```
max_sweets(sweets_rating, from_pos):
    if from_pos is end of list
        return rating(last) # Rating of last item in list

    else if from_pos is one before the end


        return max(rating(last), rating(last - 1))

    else
        return max((rating(from_pos) + max_sweets(from_pos + 2)),
                   max_sweets(from_pos + 1)

        // Compute both, keep the larger one
```

PICK CHOC

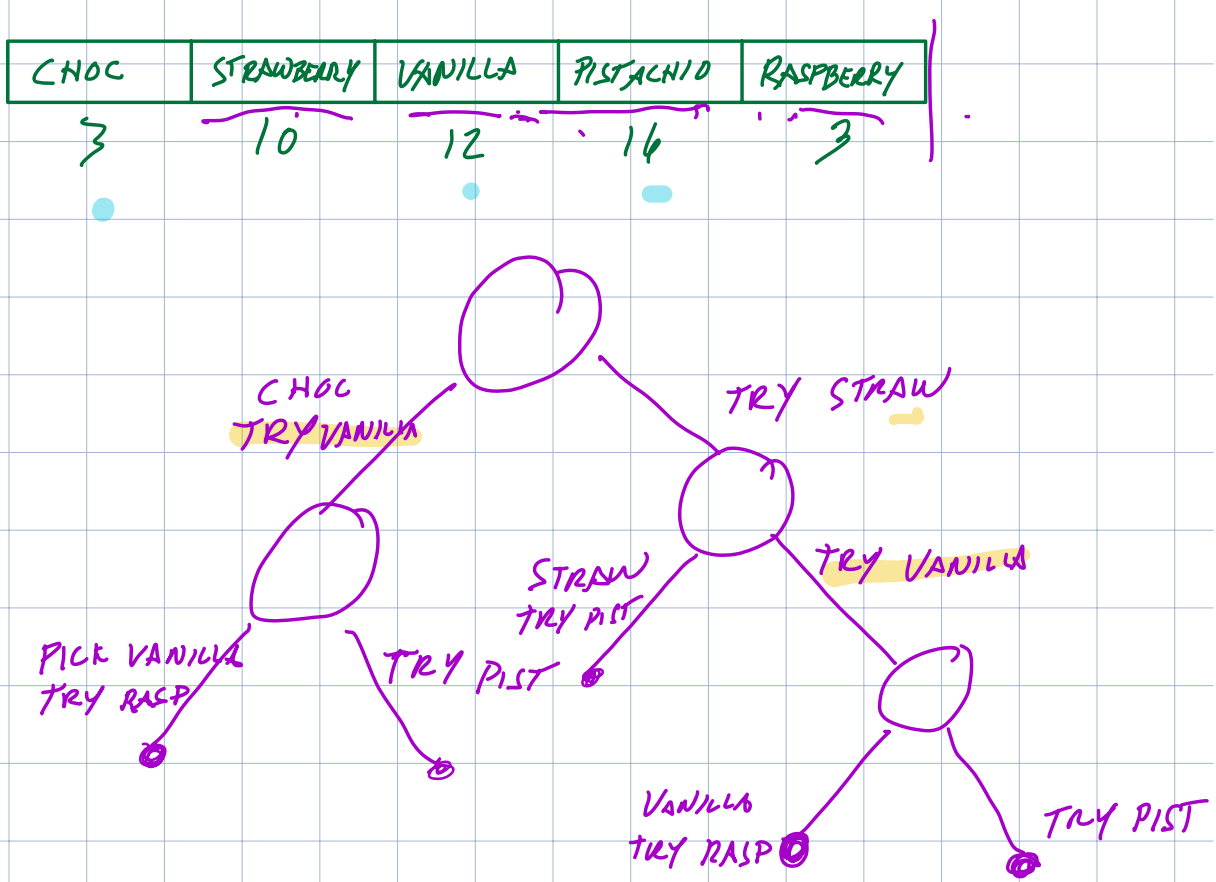
CONSIDER VANILLA

STRAWBERRY

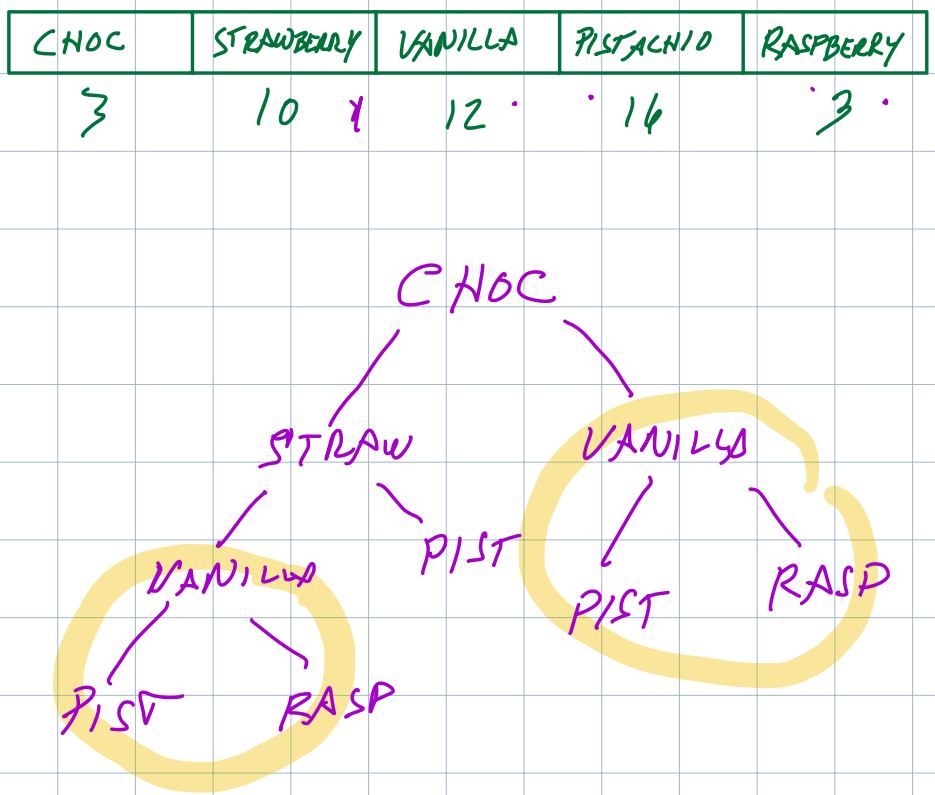


=> Can look at this like sort of a tree structure, where each edge is a call to the recursive function over a certain subset of the data

Here's what the full tree could look like for this example:



Try it: take a look at the edges. What do you notice?



We have some repeats! Multiple instances of the same tree structure => calling the recursive function multiple times with the same arguments!

=> We are doing the **same computation multiple times,** which is inefficient

=> on a big list, this could get expensive!

For this problem, the runtime actually turns out to be *exponential* ie, O(2^n), which is much worse than other algorithms we've seen so far. For more details on this, see the typed notes.