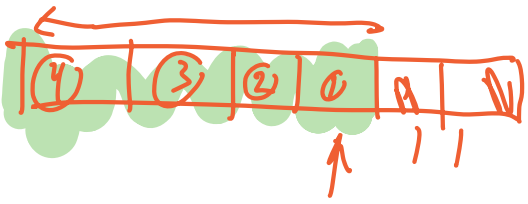
Multiple ways to do dynamic programming



Initial ChatGPT version

```
def max_sweets_chatgpt(n, rating):
    dp = [0] * n  # Make list with N 0's in it
    dp[0] = rating[0]
    dp[1] = max(rating[0], rating[1])

    for i in range(2, n):  #  iterates over 2 ... N
        dp[i] = max(dp[i - 1], dp[i - 2] + rating[i])

    return dp[n - 1]
```

This version builds the result by iterating over the data from the start to end:
- The base cases are the first two elements of the array
- The result is in the last element of the array

=> This is called a **"top-down" approach**

Version we last class

```
def max_sweets_class(ratings):
    if len(ratings) == 1:
        return ratings[0]

    best = [0] * len(ratings) # Makes an array with all zeroes in it


    best[-1] = ratings[-1]  # Base case (last item)
    best[-2] = max(ratings[-1], ratings[-2])  # Base case (second to last)

    # Fill in remaining spots, working right to left
    # Start at second from last, stop at index 0, going backwards
    for i in range(len(ratings) - 3, -1, -1):  #
        best[i] = max(ratings[i] + best[i + 2],
                      best[i + 1])

    return best[0]


# How to run each one
sweets = [3, 10, 12, 16, 4]
print(max_sweets_chatgpt(len(sweets), sweets)) # 26
print(max_sweets_class(sweets))  # 26
```

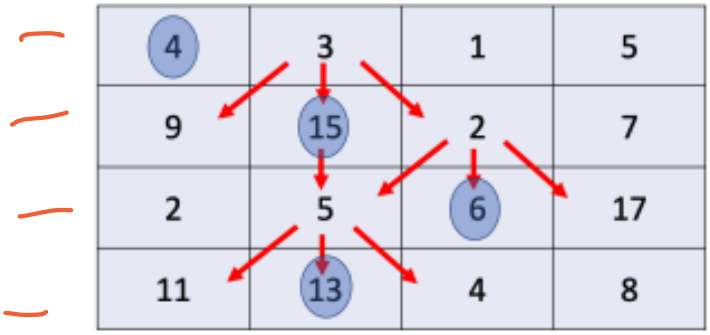
This version builds the result by iterating over the data from the end of the array back to the beginning:
- Base cases are the last two items
- Iterate over the array starting from end, going backwards
- Result is in the first element of the array

=> This is called a **"bottom-up" approach**

These are just different ways of formulating the solution to the problem. It's always possible to write both, though one form may seem more intuitive for different situations

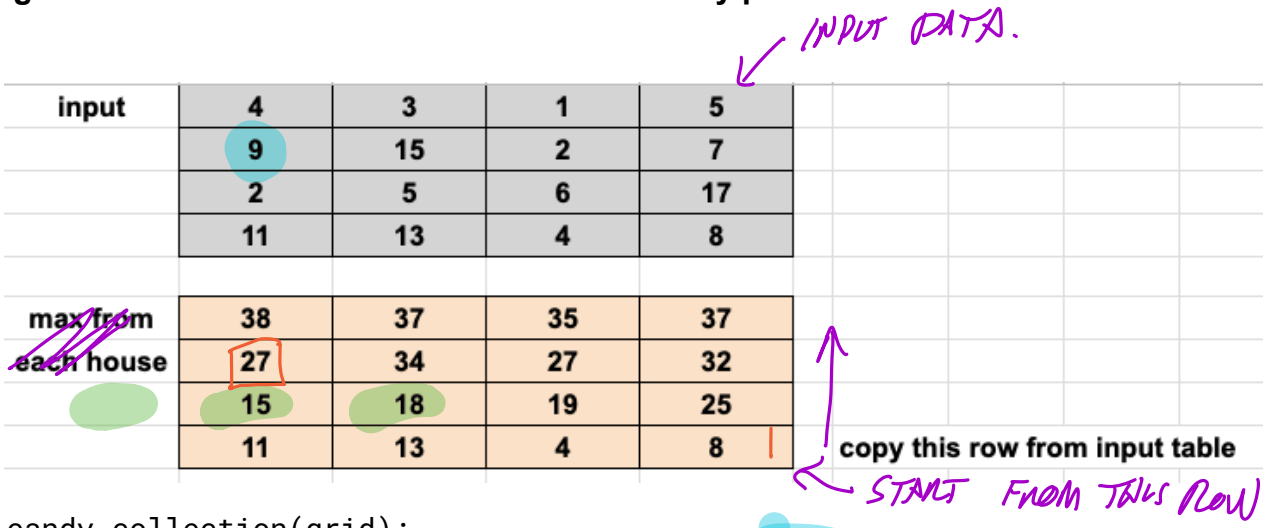
New Problem: Gathering Goodies in 2 dimensions

Imagine that you live in a neighborhood that is laid out in a grid. It's Halloween, and you get to stop at one house on each east-west street (in each row) to collect candy. The idea is that you will start from some house in the top row, then make your way down to the bottom row. From each house you visit, the next one has to be either directly below or diagonally adjacent. Your goal is to maximize the number of pieces of candy that you collect in total. Write a program to find the best path through the neighborhood.



Work either by hand in below table or in spreadsheet

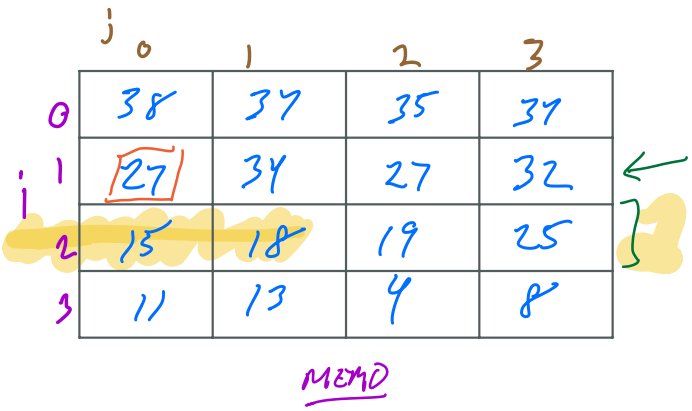
Debugging the ChatGPT solution to the Halloween candy problem



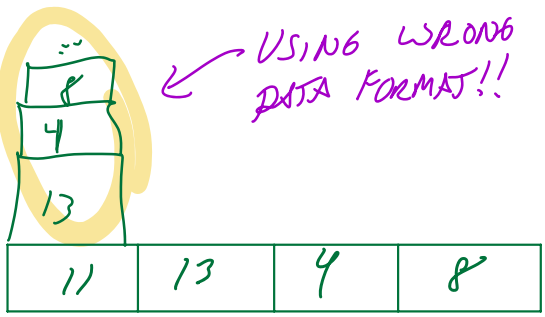

```
def max_candy_collection(grid):
    m = len(grid) # number of rows
    n = len(grid[0]) # number of columns

    # Initialize the memo table with the same values as the bottom row of the grid
    memo = [list(row) for row in grid[-1:]]
    # Fill in the memo table by working our way up from the second-to-last row
    for i in range(m-2, -1, -1):
        for j in range(n):
            # Calculate the maximum number of candy pieces that can be collected
            # from the next house by taking the maximum of the three possible
            # moves from the next row's corresponding houses, plus the value of
            # the current house.
            candy = grid[i][j] + max(
                memo[-1][j],
                memo[-1][max(j-1, 0)],
                memo[-1][min(j+1, n-1)]
            )

            # Store the maximum number of candy pieces in the table for this house
            memo.insert(0, [candy])
    # Find the max collection by checking the values in the top row of the table
    max_candy = max(memo[0])
    return max_candy
```

The rest of these notes are mainly scratch work from live code debugging done in class. See the worked out example for a text-based version of how this works. For more on these notes, see the recording.

memo (expected)



memo (ChatGPT version, so far)



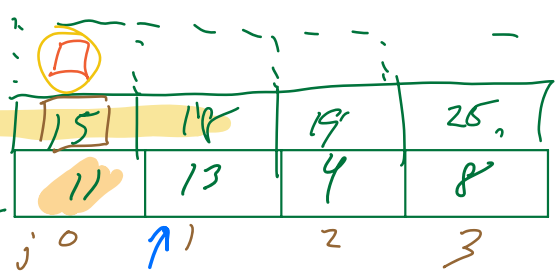
Before, we make the whole data structure at the beginning
This programs builds it as we go

4	3	1	5
9	15	2	7
2	5	6	17
11	13	4	8

How did chatGPT get 22?!?!?

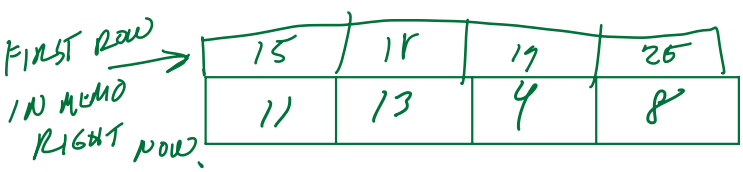
```
memo[i][j] = grid[i][j] + max(the three cells below this one)
```

```
candy = grid[i][j] + max(
    memo[-1][j],
    memo[-1][max(j-1, 0)],
    memo[-1][min(j+1, n-1)]
)
```



9 + MAX(11, 11, 13) = 9 + 13 = 22

MEMO[-1][0] => 11
MEMO[-1][0] => 11
MEMO[-1][1] => 13



A generalized debugging strategy

1. Figure out (on paper, in REPL, etc) what you expect program to be doing
2. Try to figure out how program deviates from that behavior
3. If this doesn't show you where the problem is, try to look deeper at what happens in (1) and (2), repeat

ChatGPT, and friends are based on large language models (LLMs)
Take in large quantities of human knowledge, speech, etc, built up repository for what is likely to come after something else

I am a student at Brown _____
=> predicts what comes next

We don't know (and it's really hard to predict) what inputs it's drawing from. It's not smart. It just has a lot of data about the words and terms, so this seems likely to come next.

Whose text did it take? Where did it come from? Some of your code, my code? Is it correct?