Introduction to Greedy Algorithms

Greedy Algorithms Part 1

What is a greedy algorithm?

- This design pattern is applied to optimization problems
- In order to solve a given optimization problem using the greedy method, we proceed by a sequence of choices.
- This approach does not always lead to an optimal solution.
- But there are several problems that it does work for, and such problems are said to possess the **greedy-choice property**.
- This is the property that a global optimal condition can be reached by a series of locally optimal choices (that is, choices that are each the current best from among the possibilities available at the time), starting from a well-defined starting condition.

Chocolate Party

- You are hosting a party for six of your friends to celebrate the results of the previous semester.
- You want to distribute chocolates to your friends based on their CGPA (out of 10).
- However, you want to be fair and follow these rules:
- ***Rule 1: Each friend must receive at least one chocolate.***
- ***Rule 2: A friend with a higher CGPA than their neighbor (the friend sitting next to them) must receive more chocolates than that neighbor.***
- Your goal is to ***minimize*** the total number of chocolates you distribute while satisfying the above rules.

Example

Friend	CGPA (out of 10)
Alice	8.5
Bob	7.0
Carol	9.0
Dave	6.5
Eve	8.0
Frank	7.5

Example

- Let’s say the friends are sitting in this order: Alice, Bob, Carol, Dave, Eve, Frank.
- Alice (8.0) has a higher CGPA than Bob (7.0), so Alice must receive more chocolates than Bob.
- Bob (7.0) has a lower CGPA than Carol (9.0), so Carol must receive more chocolates than Bob.
- Carol (9.0) has a higher CGPA than Dave (8.0), so Carol must receive more chocolates than Dave.
- Dave (8.0) has a lower CGPA than Eve (10.0), so Eve must receive more chocolates than Dave.
- Eve (10.0) has a higher CGPA than Frank (9.0), so Eve must receive more chocolates than Frank.

Two-pass greedy algorithm

- ***Left-to-Right Pass:***
- Give each friend one chocolate.
- If a friend has a higher CGPA than the friend to their left, give them one more chocolate than the left friend.

```
if cgpa[i] > cgpa[i - 1]:
                chocolates[i] = chocolates[i - 1] + 1
```

- ***Right-to-Left Pass:***
- If a friend has a higher CGPA than the friend to their right, ensure they have at least one more chocolate than the right friend.

```
if cgpa[i] > cgpa[i + 1]:
        chocolates[i] = max(chocolates[i], chocolates[i + 1] + 1)
```

Left-to-Right Pass

- Alice: 1 chocolate (no one to her left).
- Bob: 1 chocolate (CGPA ≤ Alice).
- Carol: 2 chocolates (CGPA > Bob).
- Dave: 1 chocolate (CGPA ≤ Carol).
- Eve: 2 chocolates (CGPA > Dave).
- Frank: 1 chocolate (CGPA ≤ Eve).

Left-to-Right Pass

Friend	CGPA	Left-to-Right Pass
Alice	8	1
Bob	7	1
Carol	9	2
Dave	8	1
Eve	10	2
Frank	9	1

Similarly, right to left pass

Friend	CGPA	Left-to-Right Pass	Right-to-Left Pass
Alice	8	1	2
Bob	7	1	1
Carol	9	2	2
Dave	8	1	1
Eve	10	2	2
Frank	9	1	1

- **Time Complexity: O(n)**
- **Space Complexity: O(n) additional space required.**

Solve

- Leetcode 135, Candy
- https://leetcode.com/problems/candy/

Single pass greedy solution (Optional)

- If we are not interested in knowing who gets how many chocolates and only interested in knowing the total number of chocolates.
- ***Up***:
  - Counts how many friends have increasing CGPA from the previous friend.
  - This helps us determine how many chocolates we need for a friend with a higher CGPA than the previous friend.
- ***Down***:
  - Counts how many friends have decreasing CGPA from the last friend.
  - This helps us determine how many chocolates we need for a friend with a lower CGPA than the previous friend.
- ***Peak***:
  - Keeps track of the last highest point in an increasing sequence.
  - When we have a decreasing sequence after the peak, we can refer to the Peak to adjust the number of chocolates if needed.

Single pass greedy solution (Optional) cont’d

- Initialize Your Counters. Start with ret = 1 because each friend must have at least one chocolates. Initialize up, down, and peak to 0.
- Loop Through CGPA
- For each pair of adjacent friends, compare their CGPA. Here are the scenarios:
- If the CGPA is increasing: Update up and peak by incrementing them by 1. Set down to 0. Add up + 1 to ret because the current friend must have one more chocolates than the previous friend.
- If the CGPA is the same: Reset up, down, and peak to 0, because neither an increasing nor a decreasing trend is maintained. Add 1 to ret because the current friend must have at least one chocolates.
- If the CGPA is decreasing: Update down by incrementing it by 1. Reset up to 0. Add down to ret. Additionally, if peak is greater than or equal to down, decrement ret by 1. This is because the peak friend can share the same number of chocolates as one of the friends in the decreasing sequence, which allows us to reduce the total number of chocolates.
- Return the Total chocolates Count
- At the end of the loop, ret will contain the minimum total number of chocolates needed for all the friends, so return ret.