Greedy Algorithms Handbook — Python Edition

Light Theme • Inline Explanations • Interview-Focused

Overview:
A greedy algorithm builds up a solution step-by-step by always choosing the option that looks best at the current moment. It works well when local optima lead to a global optimum (Greedy Choice Property + Optimal Substructure).

Common examples include Jump Game, Gas Station, Fractional Knapsack, and Dijkstra's Algorithm.

1■■ Best Time to Buy and Sell Stock

```
def max_profit(prices):
    mn = float('inf')
    ans = 0
    for p in prices:
        if p < mn:
            mn = p
        ans = max(ans, p - mn)
    return ans
```

Track the minimum price (`mn`) and calculate profit at each step. Return the maximum difference seen. Time O(n), Space O(1).

2■■ Jump Game II — Minimum Jumps

```
def jump(nums):
    jumps = 0
    cur_end = 0
    far = 0
    for i in range(len(nums) - 1):
        far = max(far, i + nums[i])
        if i == cur_end:
            jumps += 1
            cur_end = far
    return jumps
```

Greedy window expansion: expand the farthest reachable index; when current window ends, increment jump count. Ensures minimal jumps. Time O(n), Space O(1).

3■■ Gas Station

```
def can_complete_circuit(gas, cost):
    total = tank = start = 0
    for i in range(len(gas)):
        diff = gas[i] - cost[i]
        total += diff
        tank += diff
        if tank < 0:
            start = i + 1
            tank = 0
    return start if total >= 0 else -1
```

Track total and current tank fuel; reset start when tank < 0. If total gas >= total cost, return last start. Time O(n), Space O(1).

4■■ Candy Distribution

```
def candy(ratings):
    n = len(ratings)
    candies = [1] * n
    for i in range(1, n):
        if ratings[i] > ratings[i-1]:
            candies[i] = candies[i-1] + 1
    for i in range(n-2, -1, -1):
        if ratings[i] > ratings[i+1]:
            candies[i] = max(candies[i], candies[i+1] + 1)
    return sum(candies)
```

Forward pass ensures right condition; backward fixes left condition. Sum gives minimal candies. Time O(n), Space O(n).

5■■ Fractional Knapsack

```
def fractional_knapsack(weights, values, capacity):
    items = sorted(zip(values, weights), key=lambda x: x[0]/x[1], reverse=True)
    total = 0.0
    for val, wt in items:
        if capacity >= wt:
            total += val
            capacity -= wt
        else:
            total += val * (capacity / wt)
            break
    return total
```

Sort by value/weight ratio and fill capacity greedily. Allows fractional part. Time O(n log n).

■ **Summary Cheat Sheet**

ProblemApproachTimeSpaceKey Idea Best Time to Buy/SellTrack min & profitO(n)O(1)Local diff Jump Game IIGreedy windowO(n)O(1)Max reach layer Gas StationRestart on deficitO(n)O(1)Total ≥ 0 rule CandyForward & backwardO(n)O(n)Two pass constraint KnapsackSort by ratioO(n log n)O(1)Best value density