Greedy Algorithm Patterns & Easy Identification Cheat Sheet with Pseudocode (Max/Min/Count/Keywords Focused)

1️⃣Key Patterns That Signal Greedy Algorithm

A. Maximize / Minimize Count

- **Pattern:** Select maximum or minimum number of elements satisfying a condition.
- **Clues:** “maximum number of intervals,” “minimum number of steps,” “largest/smallest subset.”
- **Pseudocode:**

```
sort elements by criteria (e.g., end time ascending for intervals)
count = 0
last_taken = -inf
for element in elements:
    if element.start >= last_taken:
        count += 1
        last_taken = element.end
return count
```

- **Example:** Activity Selection, Minimum Number of Arrows to Burst Balloons

B. Maximum / Minimum Value / Profit

- **Pattern:** Pick elements to maximize or minimize total value/profit.
- **Clues:** “maximize profit,” “maximize sum,” “minimize cost.”
- **Pseudocode:**

```
sort items by value descending or value/weight ratio descending
total_value = 0
for item in items:
    if capacity >= item.weight:
        take full item
    else:
        take fraction
    update capacity and total_value
return total_value
```

- **Example:** Fractional Knapsack, Maximize Stock Profit

C. Choose Largest / Smallest Next

- **Pattern:** Pick next largest/smallest available element.
- **Clues:** “always pick largest next,” “pick smallest available.”

- **Pseudocode:**

```
sort elements ascending/descending
for element in sorted_elements:
    if can_take(element):
        take it
```

- **Example:** Jump Game, Minimum Platforms, Gas Station Problem

D. Merge / Combine Smallest Elements

- **Pattern:** Combine smallest elements to minimize cost.
- **Clues:** “merge files,” “minimize total cost.”
- **Pseudocode:**

```
priority_queue = min-heap of weights
total_cost = 0
while size(priority_queue) > 1:
    a = extract_min(priority_queue)
    b = extract_min(priority_queue)
    total_cost += a + b
    insert(priority_queue, a + b)
return total_cost
```

- **Example:** Huffman Coding, Optimal Merge Pattern

E. Maximize Coverage / Range

- **Pattern:** Pick elements to cover maximum range or intervals.
- **Clues:** “cover all intervals,” “maximum range covered.”
- **Pseudocode:**

```
sort intervals by start time
end = -inf
for interval in intervals:
    if interval.start > end:
        take interval
        end = interval.end
```

- **Example:** Interval Coverage, Set Cover Approximation

2️⃣ Key Words Signaling Greedy Approach

Category	Keywords / Phrases	Meaning / Hint
Count / Maximize / Minimize	max, minimum, maximum, count, largest, smallest	Pick elements sequentially to optimize quantity or size
Intervals / Scheduling	earliest finish, non-overlapping, interval, activity, time slot, schedule	Select events/intervals in sorted order
Value / Profit / Weight	profit, value, cost, ratio, weight, reward	Maximize/minimize metric, sort by ratio or value
Coverage / Selection	cover, select, subset, range, elements	Pick elements to cover or satisfy constraints efficiently
Merge / Combine	merge, combine, join, total cost	Huffman coding, optimal merge, combine smallest first
Step / Local Choice	pick next, choose best, locally optimal, greedy choice	Step-by-step selection, local optimum may yield global optimum
Sorting / Priority	sort by, order by, priority, earliest/largest/smallest first	Sort items by key metric before selecting

---

3️⃣ Easy Method to Identify Greedy Problems (Max/Min/Count/Keywords Focus)

Step	Question to Ask	Action
1	Need **max/min count** of elements?	Sort + pick sequentially
2	Need **maximize value/profit**?	Sort by value or value/weight ratio
3	Need **pick largest/smallest next**?	Sort and choose sequentially
4	Need **combine smallest elements** to minimize cost?	Use min-heap
5	Need **maximize coverage/range**?	Sort intervals by start/end time
6	Problem contains **keywords indicating greedy**?	Check for locally optimal choice + optimal substructure

---

4️⃣ Quick Tips / Tricks

- Sorting by key metric often unlocks greedy approach.

- Max/Min/Count problems usually involve **sequential selection**.
- Use **heap/priority queue** when repeatedly selecting largest/smallest.
- Always verify greedy choice property: locally optimal choice leads to global optimum.
- Combine **greedy + sorting + heap** for complex optimization problems.
- Use keywords as a **hint**, but always validate the approach with examples.

Summary: Focus on **maximizing/minimizing values, counts, or coverage**, identify greedy keywords, and choose elements sequentially based on sorted key or priority to apply greedy effectively.