# Lecture 06: Greedy Algorithms

Swakkhar Shatabda

CSI 227: Algorithms, Summer 2014
Department of Computer Science and Engineering
United International University

United International University
QUEST FOR EXCELLENCE

# Greedy Algorithms

- Always makes the choice that looks best at the moment.
- Do not always yield optimal solutions.

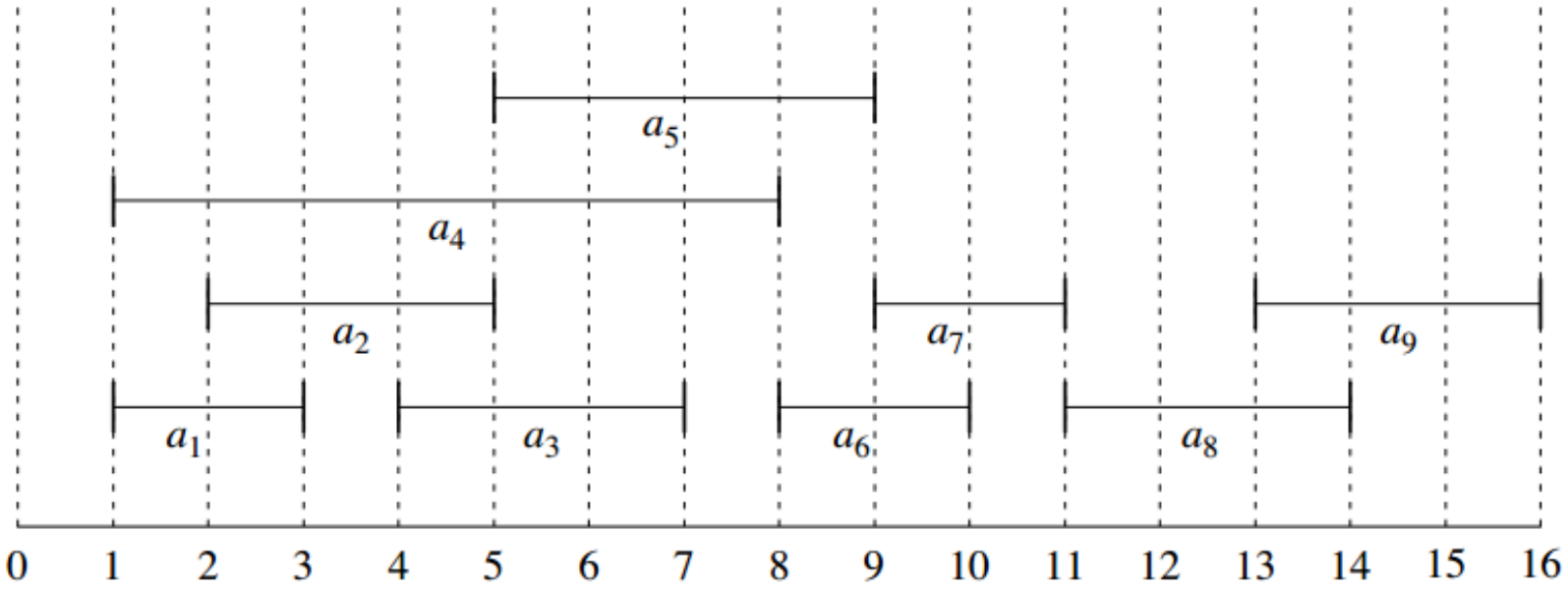
# An Activity Selection Problem

- We are given a set of activities.

| i | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| start, $s_i$ | 1 | 3 | 0 | 5 | 3 | 5 | 6 | 8 | 8 | 2 | 12 |
| finish, $f_i$ | 4 | 5 | 6 | 7 | 9 | 9 | 10 | 11 | 12 | 14 | 16 |

- Two activities are compatible if their intervals do not overlap.
- We have to select the maximum-size subset of activities that are mutually compatible.
- Subset $\{a_1, a_4, a_8, a_{11}\}$ is better than the subset Subset $\{a_3, a_9, a_{11}\}$

# Another Example




Maximum-size mutually compatible set: $\{a_1, a_3, a_6, a_8\}$.

Not unique: also $\{a_2, a_5, a_7, a_9\}$.

# Making Greedy Choice

1. Suppose the activities are sorted according to finishing time.
2. What is the intuition? Select a job that finishes first.
3. This job must be in the optimal solution. why?

# Iterative Solution

GREEDY-ACTIVITY-SELECTOR$(s[], f[], n)$

```
1  A = {a_1}
2  i = 1
3  for m = 2 to n
4       if s[m] ≥ f[i]
5            A = A ∪ {a_m}
6            i = m
7  return A
```

# Elements of Greedy Strategy

1. Greedy Choice Property
   1. Make a choice each step
   2. Make the choice before solving the sub-problem
   3. solve top-bottom
2. Optimal Substructure
3. Look at the globally optimal solution. If it contains the greedy choice, done! Else modify it to contain the greedy choice.
4. Another Example: Knapsack Problem

# Knapsack Problem

1. n items
2. Each item weighs $w[i]$ and is worth $v[i]$
3. Find subset of the items with total weight $\leq W$
4. If the thief can't take a part of any item then its 0-1 Knapsack
5. Otherwise its fractional Knapsack
6. In fractional knapsack the greedy choice exists!

# Knapsack Problem

FRACTIONAL-KNAPSACK($v[], w[], W$)

```
1  load = 0
2  i = 0
3  while load ≤ W and i < n
4        if w[i] ≤ W − load
5              take all of item i
6        else take (W − load)/w[i] of item i
7        update load by what was taken for item i
8        i = i + 1
```

# 0-1 Knapsack

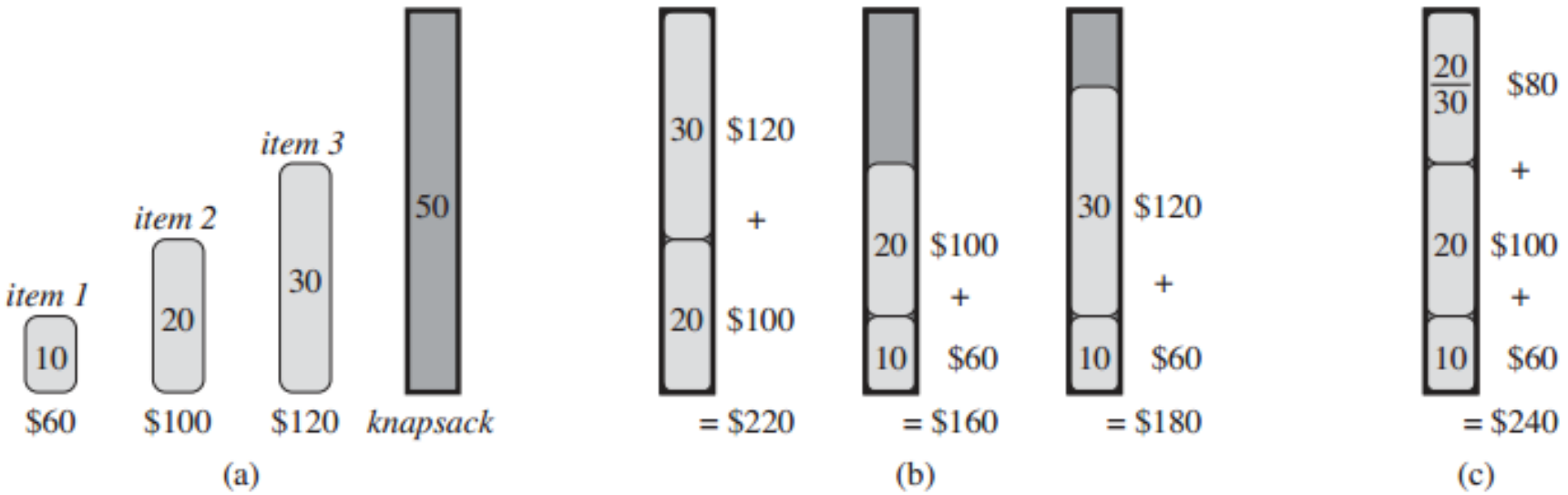



**Figure 16.2** An example showing that the greedy strategy does not work for the 0-1 knapsack problem. **(a)** The thief must select a subset of the three items shown whose weight must not exceed 50 pounds. **(b)** The optimal subset includes items 2 and 3. Any solution with item 1 is suboptimal, even though item 1 has the greatest value per pound. **(c)** For the fractional knapsack problem, taking the items in order of greatest value per pound yields an optimal solution.

# Reading

Chapter 16

# Thats it!

Thank you