Greedy Algorithms:

Activity Selection Problem

Fractional Knapsack problem

Dr. G P Gupta

Elements of Greedy Strategy

- An greedy algorithm ***makes a sequence of choices***, each of the choices that **seems best at the moment, it is chosen.**
  - **NOT always produce an optimal solution**
- ***Two ingredients*** that are exhibited by most problems that lend themselves to a greedy strategy
  - ***Greedy-choice property***
  - ***Optimal substructure***

Greedy-Choice Property

- ***A globally optimal solution can be arrived at by making a locally optimal (greedy) choice***
  - *Make whatever choice seems best at the moment and then solve the sub-problem arising after the choice is made*
  - *The choice made by a greedy algorithm may depend on choices so far, but it cannot depend on any future choices or on the solutions to sub-problems*
- Of course, we must prove that a greedy choice at each step yields a globally optimal solution

Greedy algorithms

- A *greedy algorithm* always makes the choice that looks best at the moment
  - **My everyday examples:**
    - Driving in Delhi, Mumbai , Raipur for that matter
    - Playing cards
    - ***Invest on stocks***
    - ***Choose a university***
- **The hope:**
  - ***a locally optimal choice will lead to a globally optimal solution***
- For some problems, it works
- greedy algorithms tend to be easier to code

Greedy algorithms: Examples

- Activity Selection Problem
- Huffman codes
- fractional knapsack Problem
- Kruskal's MST algorithm
- Prim's MST algorithm
- Dijkstra's SSSP algorithm

An Activity Selection Problem

- **Input:** **A set of activities S = $\{a_1, \ldots, a_n\}$**
- Each activity has ***start time*** and a ***finish time***
  - $a_i = (s_i, f_i)$
- **Two activities are *compatible*** if and only if ***their interval does not overlap***
- **Output:** *a maximum-size subset of mutually compatible activities*

The Activity Selection Problem

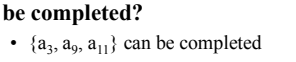
- Here are a set of start and finish times

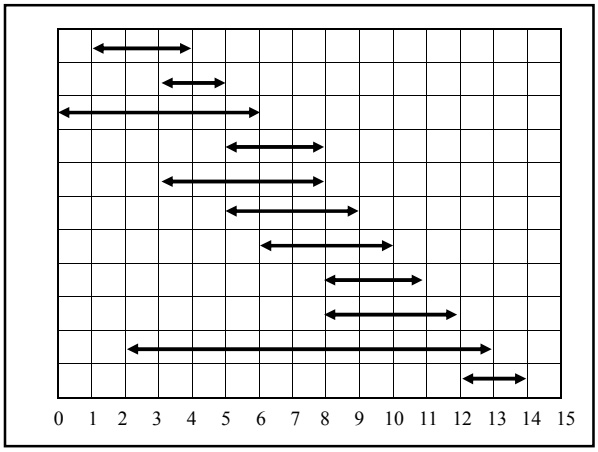
i	1	2	3	4	5	6	7	8	9	10	11
s_i	1	3	0	5	3	5	6	8	8	2	12
f_i	4	5	6	7	8	9	10	11	12	13	14

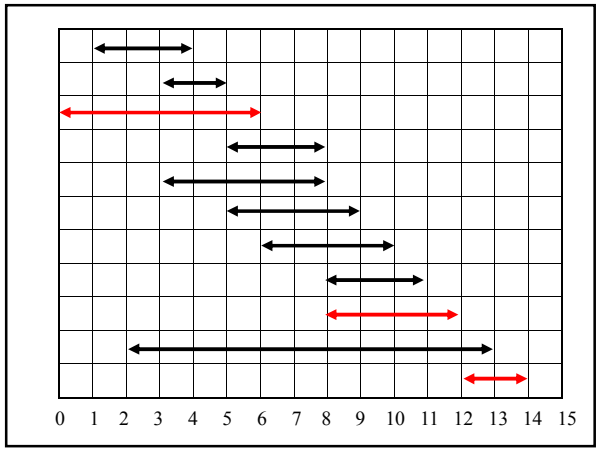
- **What is the maximum number of activities that can be completed?**
  - $\{a_3, a_9, a_{11}\}$ can be completed
  - But so can $\{a_1, a_4, a_8, a_{11}\}$ which is a larger set
  - But it is not unique, consider $\{a_2, a_4, a_9, a_{11}\}$

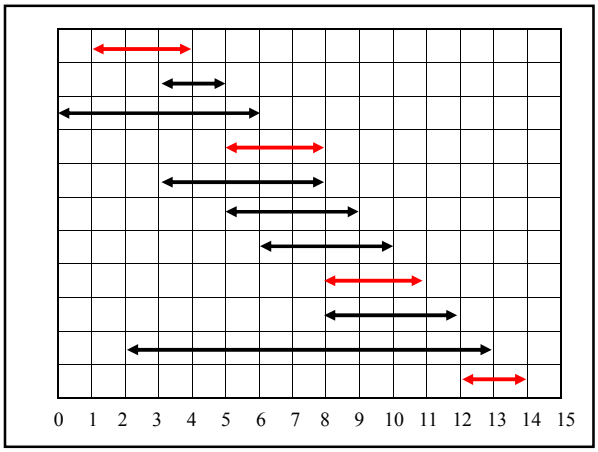
Interval Representation

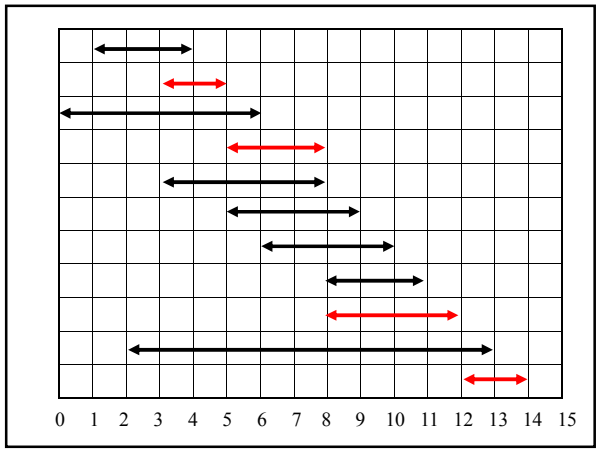
i	1	2	3	4	5	6	7	8	9	10	11
s_i	1	3	0	5	3	5	6	8	8	2	12
f_i	4	5	6	7	8	9	10	11	12	13	14





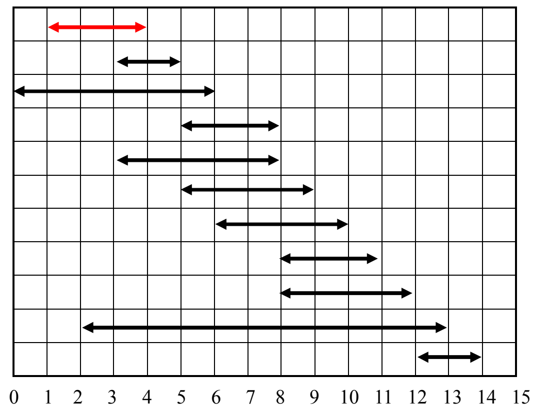


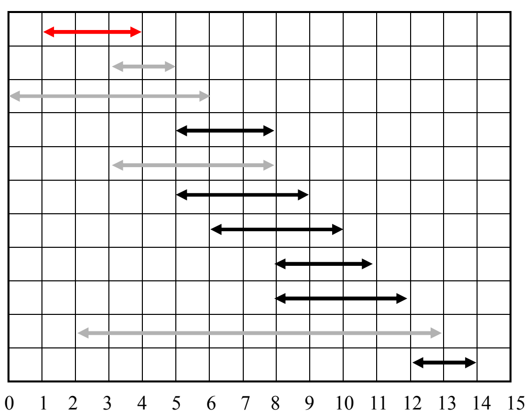


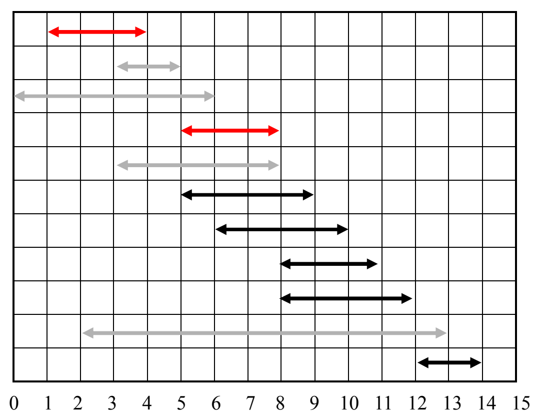


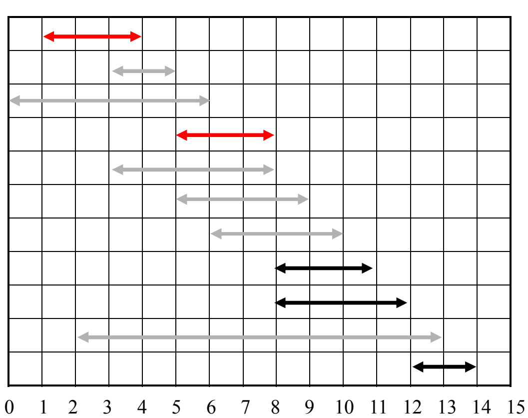
Early Finish Greedy

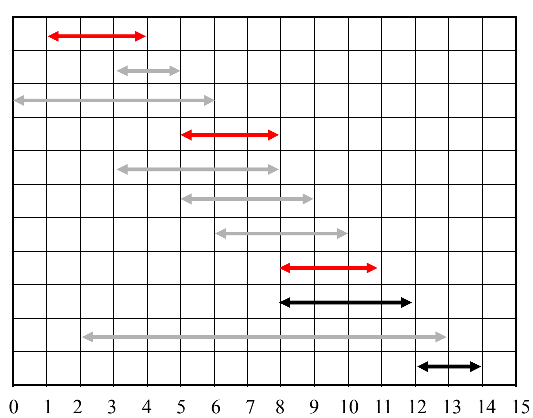
- Select the activity with the earliest finish
- Eliminate the activities that could not be scheduled
- Repeat!

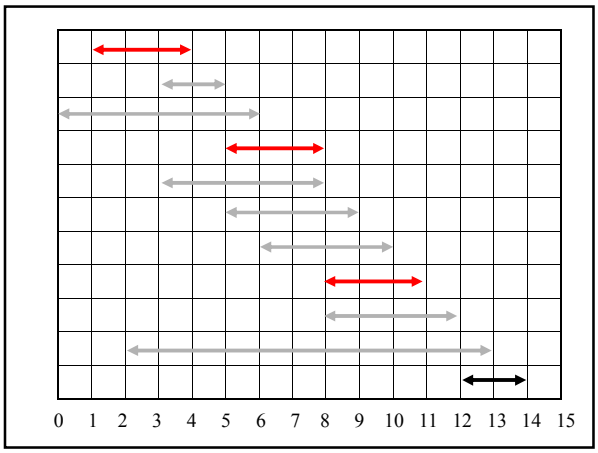


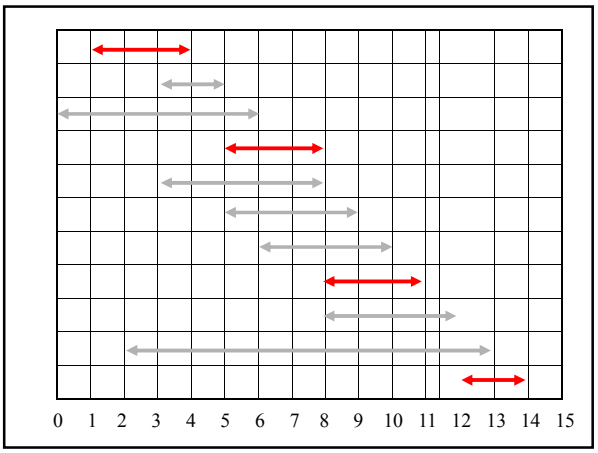












A recursive greedy algorithm

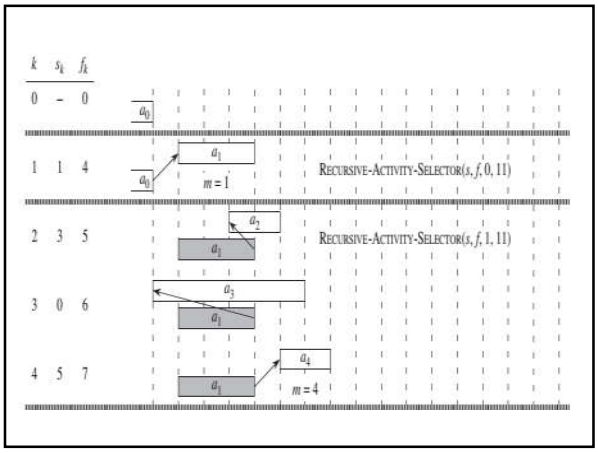
- **The initial call**, which solves the entire problem, is RECURSIVE-ACTIVITY-SELECTOR (s, f, 0,n).

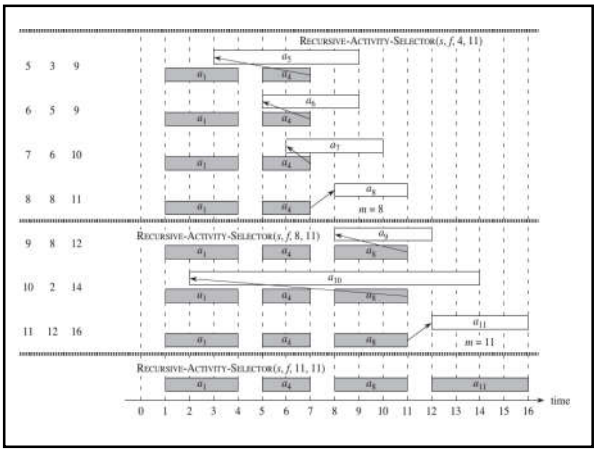
```
RECURSIVE-ACTIVITY-SELECTOR(s, f, k, n)
1  m = k + 1
2  while m ≤ n and s[m] < f[k]      // find the first activity in S_k to finish
3      m = m + 1
4  if m ≤ n
5      return {a_m} ∪ RECURSIVE-ACTIVITY-SELECTOR(s, f, m, n)
6  else return ∅
```

Example

consider the following set S of activities:

i	1	2	3	4	5	6	7	8	9	10	11
s_i	1	3	0	5	3	5	6	8	8	2	12
f_i	4	5	6	7	9	9	10	11	12	14	16





Knapsack Problem

Knapsack Problem

- One wants to pack n items in a luggage
  - The ith item is worth v_i dollars and weighs w_i pounds
  - Maximize the value but cannot exceed W pounds
  - v_i, w_i, W are integers
- **0-1 knapsack** → *each item is taken or not taken*
- **Fractional knapsack** → *fractions of items can be taken*
- **Both exhibit the optimal-substructure property**
  - **0-1:** If item j is removed from an optimal packing, the remaining packing is an optimal packing with weight at most W-w_j
  - **Fractional:** If w *pounds* of item j is removed from an optimal packing, the remaining packing is an optimal packing with weight at most W-w that can be taken from other n-1 items plus $w_j - w$ of item j

Fractional knapsack Problem

Fractional knapsack Problem

The problem:

Given a knapsack with a certain capacity M,
n items, are to be put into the knapsack,
each has a weight $w_1, w_2, \cdots, w_n$ and $p_1, p_2, \cdots, p_n$
a profit if put in the knapsack .

The objective: find $(x_1, x_2, \cdots, x_n)$ where $0 \le x_i \le 1$

$$\text{s.t. } \sum_{i=1}^{n} p_i x_i \text{ is maximized and } \sum_{i=1}^{n} w_i x_i \le M$$

Note: All items can break into small pieces
or x_i can be any fraction between 0 and 1.

Greedy Algorithm for Fractional Knapsack problem

- **Fractional knapsack can be solvable by the greedy strategy**
  - **Compute the value per pound v_i/w_i** for each item
  - ***Obeying a greedy strategy***, take as much as possible of the item with the greatest value per pound.
  - If the supply of that item is exhausted and there is still more room, take as much as possible of the item with the next value per pound, and so forth until there is no more room
  - O(n lg n) (we need to sort the items by value per pound)
  - Greedy Algorithm?

Example:

$n = 3$ $\quad (w_1, w_2, w_3) = (18, 15, 10)$

$M = 20$ $\quad (p_1, p_2, p_3) = (25, 24, 15)$

Greedy Strategy: p/w are ordered in nonincreasing order (2,3,1)

$$\left.\begin{aligned} \frac{p_1}{w_1} &= \frac{25}{18} \approx 1.4 \\ \frac{p_2}{w_2} &= \frac{24}{15} = 1.6 \\ \frac{p_3}{w_3} &= \frac{15}{10} = 1.5 \end{aligned}\right\} \Rightarrow (2,3,1)$$

$$(x_1, x_2, x_3) = (0, 1, \frac{1}{2})$$

Optimal solution

$$\sum_{i=1}^{3} p_i x_i = 25 \times 0 + 24 \times 1 + 15 \times \frac{1}{2} = \underline{31.5}$$

Greedy Algorithm for Fractional knapsack Problem

1. Calculate $v_i = p_i / w_i$ for i = 1, 2, …, n — ***O(n)***
2. Sort items by nonincreasing v_i. — ***O(nlogn)***
   (all w_i are also reordered correspondingly)
3. Let M' be the current weight limit (Initially, $M' = M$ and $x_i=0$).In each iteration, choose item i from the head of the unselected list.
   If $M' >= w_i$, set x_i=1, and $M' = M'-w_i$ — ***O(n)***
   If $M' < w_i$, set $x_i=M'/w_i$ and the algorithm is finished.

O(nlogn)

O-1 knapsack is harder!

O-1 knapsack is harder!

- **0-1 knapsack cannot be solved by the greedy strategy**
  - Unable to fill the knapsack to capacity, and the empty space lowers the effective value per pound of the packing
  - We must compare the solution to the sub-problem in which the item is included with the solution to the sub-problem in which the item is excluded before we can make the choice
  - **Dynamic Programming**

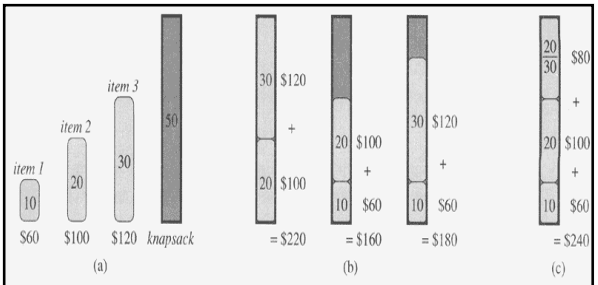


Figure 16.2 The greedy strategy does not work for the 0-1 knapsack problem. **(a)** The thief must select a subset of the three items shown whose weight must not exceed 50 pounds. **(b)** The optimal subset includes items 2 and 3. Any solution with item 1 is suboptimal, even though item 1 has the greatest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

End