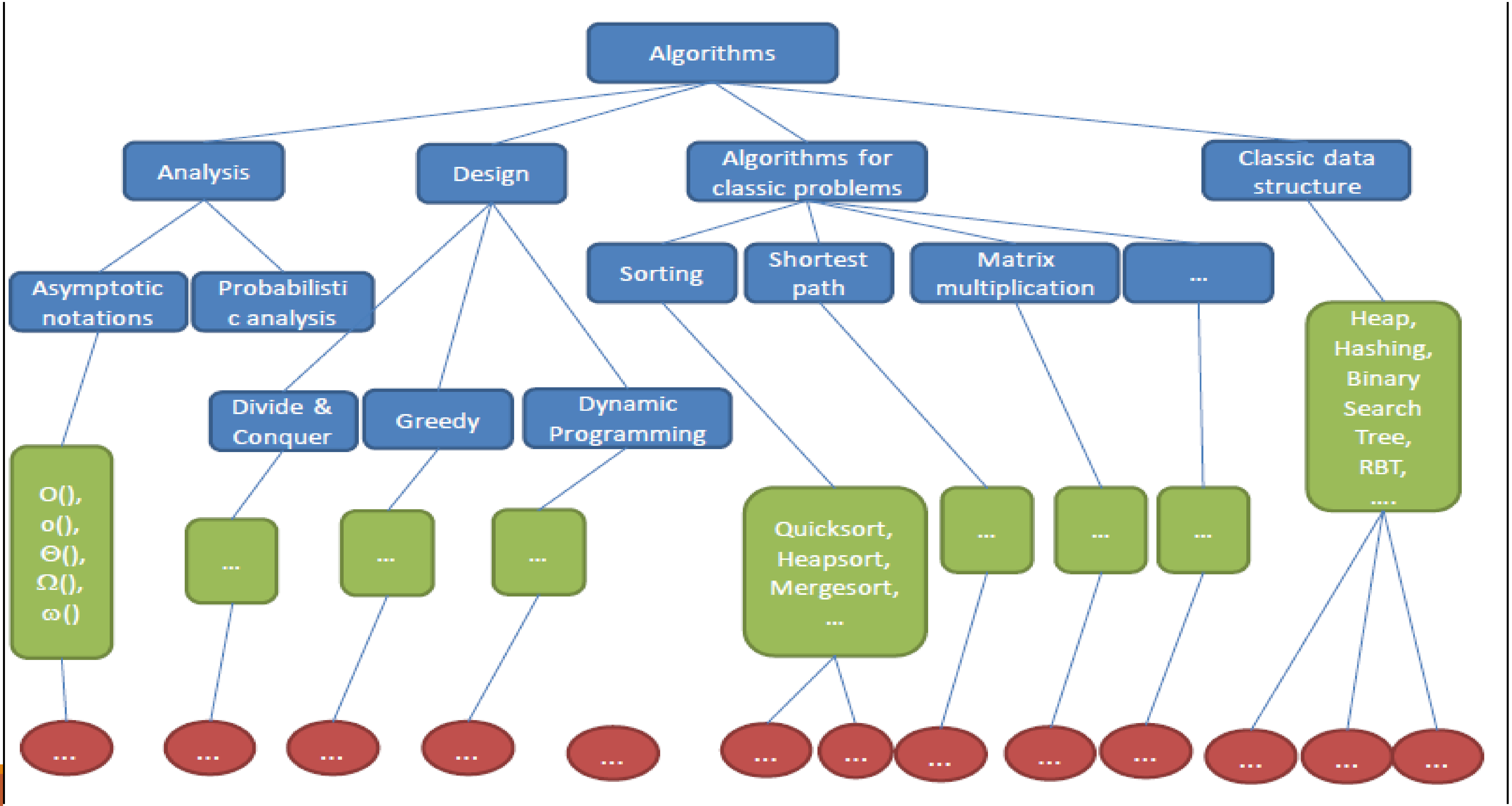
National University
of computer and emerging sciences

CS302
Design and Analysis of Algorithms

MUHAMMAD SOHAIL AFZAL

Knowledge tree



Different algorithm approaches

Some important algorithm types :

- Brute-force
- Divide and conquer
- Greedy
- Dynamic

Brute Force

A brute force algorithm simply tries *all* possibilities until a satisfactory solution is found

- Advantages:

If the solution exists then it will find it because it will try each and every possibility looking for the solution

- Disadvantages:

As it checks each and every possibility to find solution so it takes a lot of time.

Divide and Conquer

- It works by dividing problem into smaller independent sub-problems and then recursively solves these sub-problems to build solution.

  E.g. MergeSort, QuickSort etc

- It has optimal sub-structure property which means solution to problem can be built from solution of sub-problems

Greedy

- It is used to find the best solution by taking best sub-solution at every step.
- A “greedy algorithm” sometimes works well for optimization problems
- It also has optimal sub-structure property
- A greedy algorithm works in phases and at each phase:
  - You take the best you can get right now, without regard for future consequences
  - You hope that by choosing a *local* optimum at each step, you will end up at a *global* optimum

Greedy

It is good for many problems but not for all. For example :

- It is good for activity selection problem
- It is good for fractional knapsack problem
- It is not always good for 0/1 knapsack problem and coin change problem for which dynamic programming is more better approach

Dynamic Programming

- A dynamic programming algorithm works by breaking down problem into smaller overlapping sub-problems. Sub-problems that are repeated are solved only once and result is stored (which is called **memoization**) and this stored result is used next time rather than recomputing solution for another same sub-problem.

- It also has optimal substructure property

- Dynamic programming is generally used for optimization problems
  - Requires “optimal substructure” and “overlapping subproblems”
  - Optimal substructure: Optimal solution contains optimal solutions to subproblems
  - Overlapping subproblems: Solutions to subproblems can be stored and reused in a bottom-up fashion

- This differs from Divide and Conquer, where subproblems are not overlapped but independent.

Dynamic Programming

- This approach is good for lot of problems. Some of them are given below :

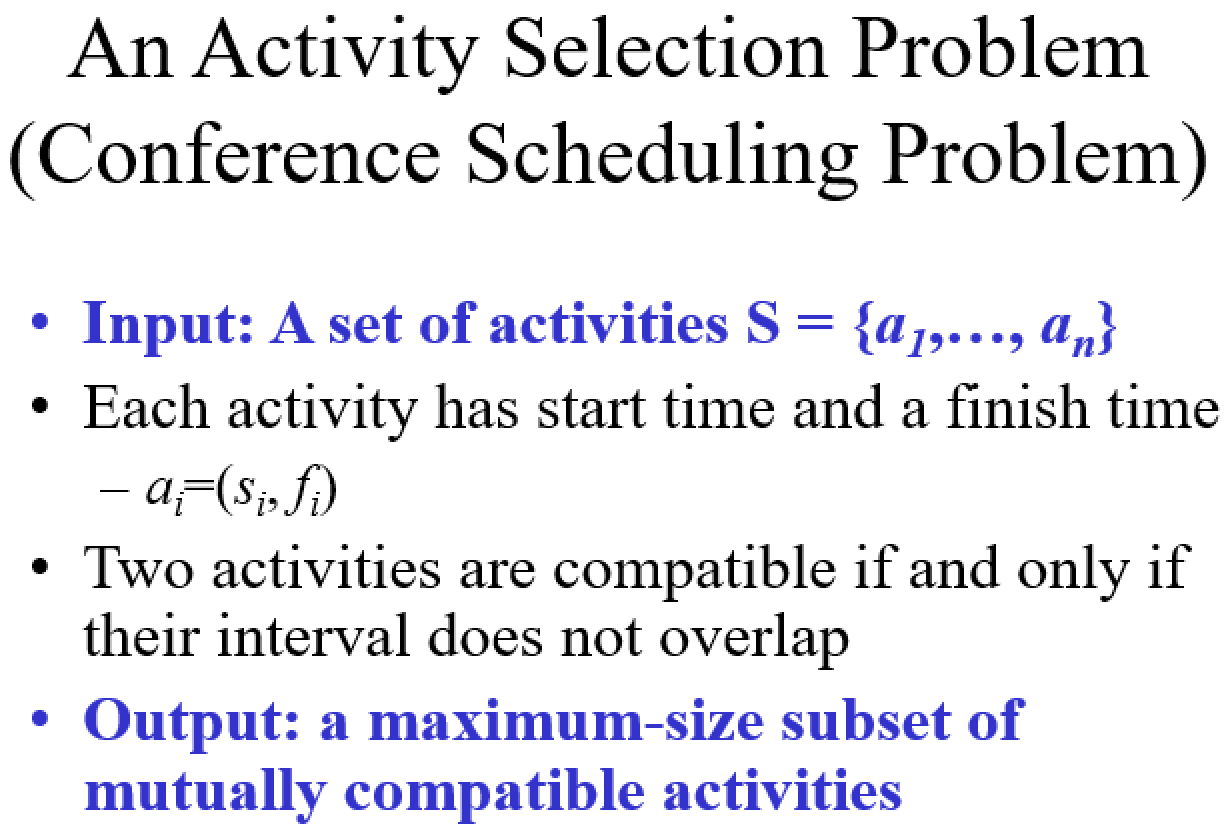
0/1 knapsack problem

Longest common sub-sequence problem

Matrix chain multiplication problem

Coin change problem

Activity Selection problem: Greedy Approach

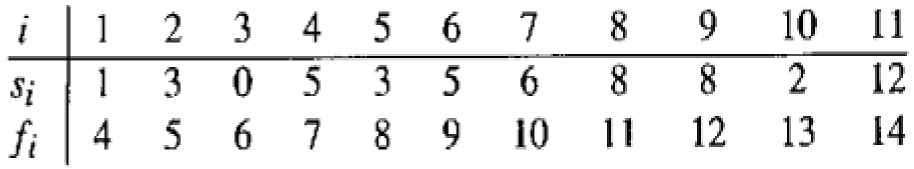


An Activity Selection Problem (Conference Scheduling Problem)

- **Input: A set of activities S = {*a_1*,…, *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**

Activity Selection problem: Greedy Approach

Here are a set of start and finish times

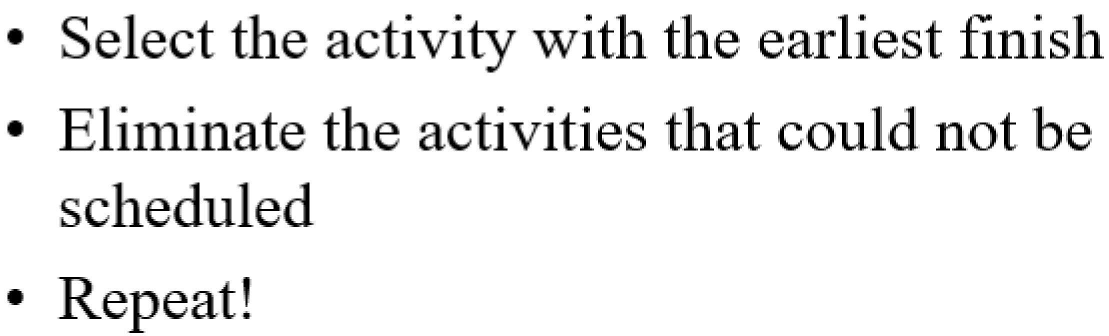


i	1	2	3	4	5	6	7	8	9	10	11
s_i	1	3	0	5	3	5	6	8	8	2	12
f_i	4	5	6	7	8	9	10	11	12	13	14

What is the maximum number of activities that can be completed?

Activity Selection problem: Greedy Approach

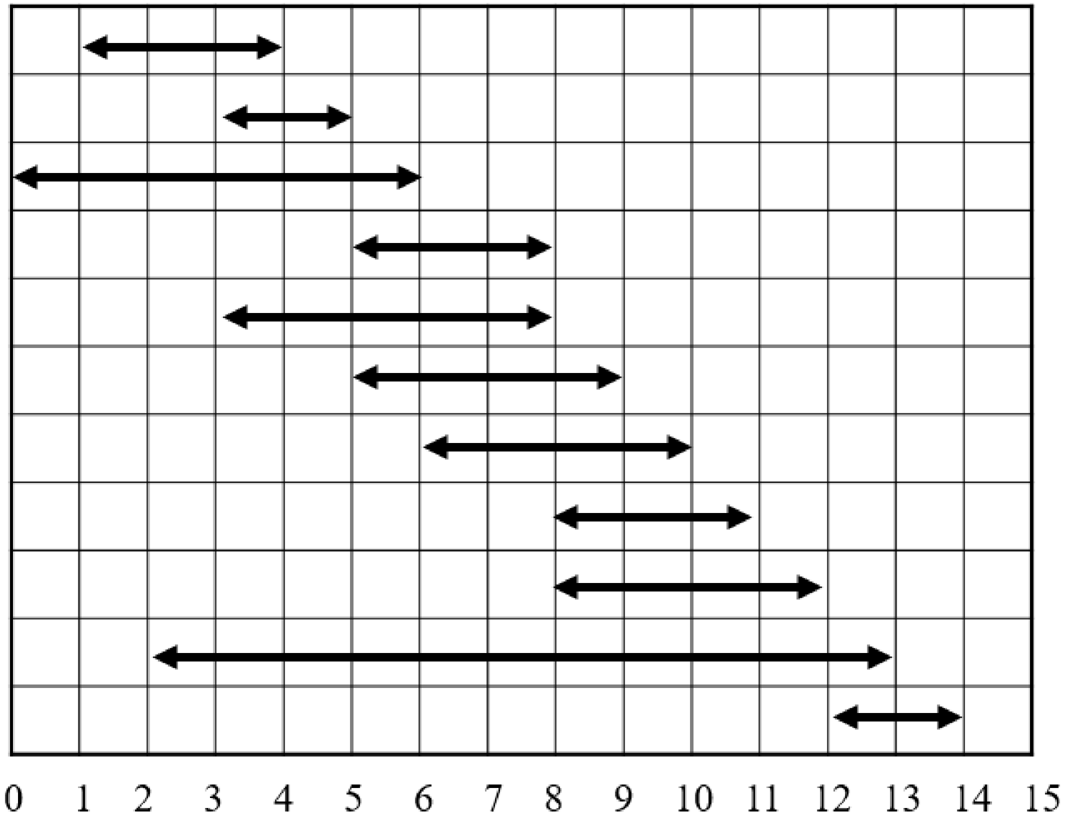
Greedy approach is :



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

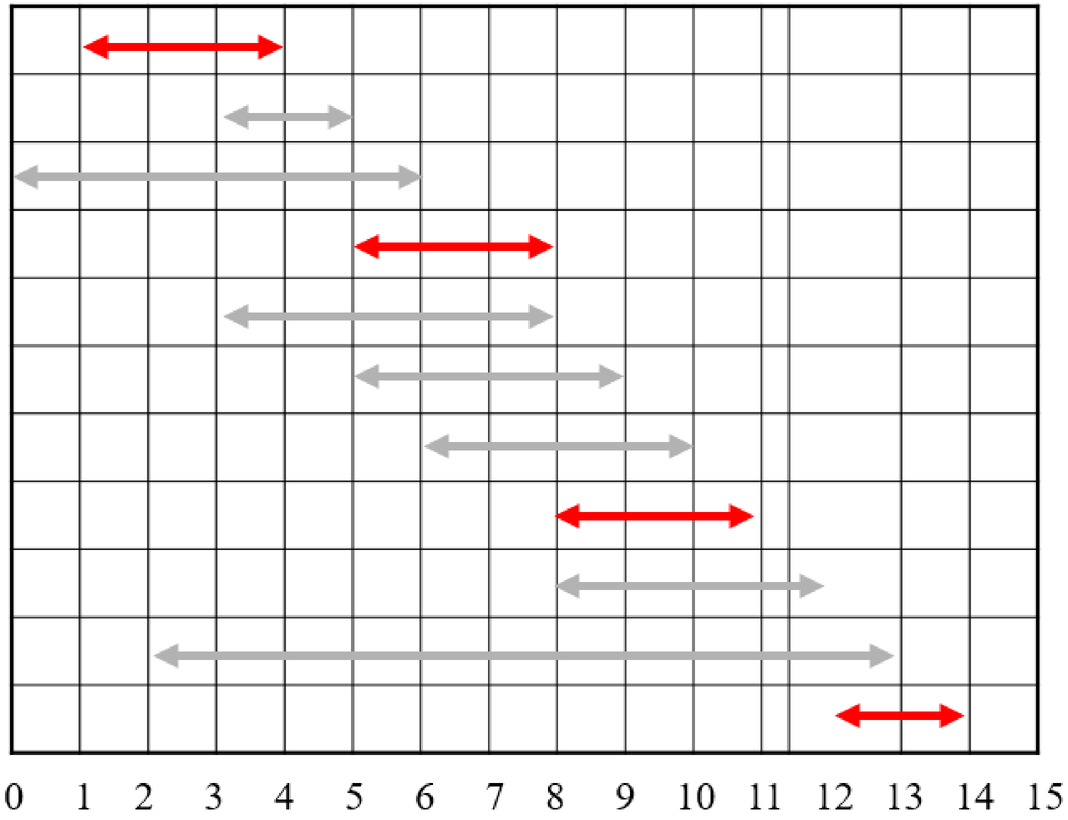
Activity Selection problem: Greedy Approach

All activities :



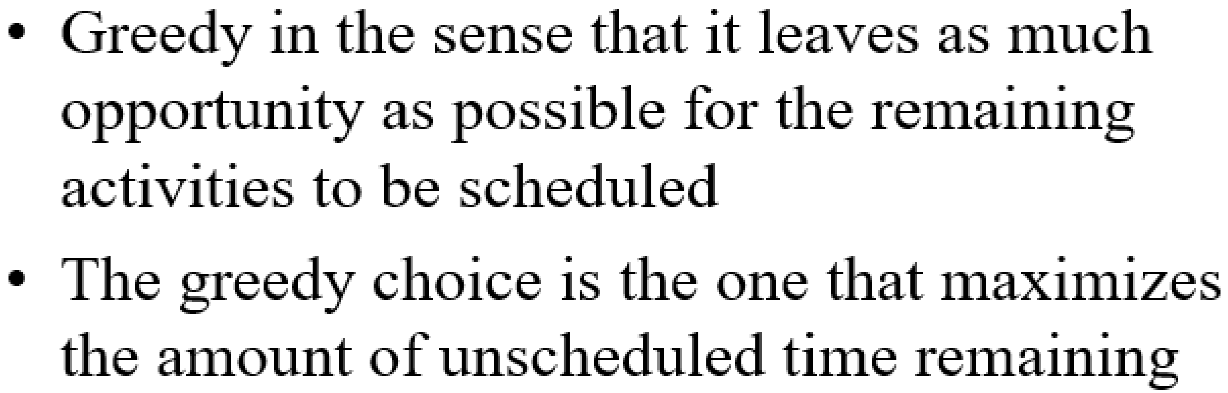
Activity Selection problem: Greedy Approach

Selected activities:



Activity Selection problem: Greedy Approach

Greedy approach is good for activity selection problem.



- Greedy in the sense that it leaves as much opportunity as possible for the remaining activities to be scheduled
- The greedy choice is the one that maximizes the amount of unscheduled time remaining