CS F364
Design & Analysis of Algorithms

ALGORITHM DESIGN TECHNIQUES - GREEDY

Greedy Algorithms – Example: Task Scheduling

Problem - Task Scheduling

- Given:
  - A set T of n tasks;
  - each task j is associated with an interval $[s_j, f_j)$
    - i.e. task j has start time s_j and finish time f_j such that $s_j < f_j$
- Feasible Solution:
  - A schedule for T such that
    - each task $j \in T$ has to be assigned a machine and
    - each machine can perform at most one task at a time
- Measure: # machines
- Goal: Minimize

Task Scheduling – Algorithm – Brute Force

- Definition:
  - Non-Overlapping tasks:
    - tasks j and k do not overlap if $s_k >= f_j$ or $s_j >= f_k$
- Sched_Bforce(T)
  - Let n = |T|
  - Find all possible ways of assigning n tasks to m machines where:
    - 1 <= m <= |T|
    - such that tasks assigned to a specific machine do not overlap
  - Choose the assignment (i.e. schedule) that requires the least number of machines.

How many possible ways are there?

Design Technique - Greedy

- Greedy Choice:
  - (Repeatedly) make a *local(ly optimal)* choice that results in a *global(ly optimal)* solution.
- In our example of Task Scheduling,
  - Possible local choices (to optimize) are:
    - Start time
    - End time
    - Length of the interval
  - We will use *"earliest start time"* as the greedy choice.

Task Scheduling – Greedy Algorithm

Algorithm Schedule(T)

```
m = 0   // number of machines
while (T not empty) {
    let j be the task with the earliest start time s_j ;
    remove j from T ;
    if  (there is a machine q whose tasks do not overlap with j)
    then {
            assign task j to machine q
     } else {
            m = m+1 ;
            assign task j to machine m
     }
}
```