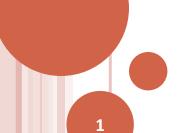
## CS F364 Design & Analysis of Algorithms

### **ALGORITHM DESIGN TECHNIQUES - GREEDY**

# **Greedy Algorithm for Task Scheduling**

- Correctness
- Time Complexity



## TASK SCHEDULING: CORRECTNESS OF GREEDY CHOICE

- If the greedy choice property does not hold for a problem then a greedy algorithm (i.e. one that makes local choices) will be incorrect
- Correctness of Scheduling algorithm:
  - Proof by contradiction:
    - oSuppose there is a schedule using p-1 machines, but the algorithm results in the use of p machines

• • • •

- Derive a Contradiction.
- QED

# TASK SCHEDULING: CORRECTNESS OF GREEDY CHOICE [2]

- Correctness of Scheduling algorithm: Proof by contradiction:
  - Suppose there is a schedule using p-1 machines,
     but the algorithm results in the use of p machines
  - If p is the last machine allocated and
     i is the first task scheduled on p then
    - each machine in 1 .. p-1 has tasks overlapping with task i
    - oThus there are p (mutually) overlapping tasks and therefore at least p machines are needed.
      - Contradiction.
  - QED

## 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<sub>i</sub>;
   remove j from T;
   if (there is a machine a whose tasks do not overlap with j)
   then {
        assign task j to machine q
                                          Time Complexity:
   } else {
                                           Loop: |T| iterations
                                           Cost of each iteration?
        m = m+1;
        assign task j to machine m
```

#### TASK SCHEDULING - GREEDY ALGORITHM: TIME COMPLEXITY

```
Algorithm Schedule(T)
```

```
m = 0 // number of machines
while (T not empty) {
  let j be the task with the earliest
  start time s<sub>i</sub>;
  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
```

```
Data Structure Choice (1):
List of tasks:
findMin must be efficient:
Use a minHeap:
```

- cost O(1) for finding the task and O(log|T|) for deleting it; (per iteration)
- (one-time) cost of heapification: O(|T|)

CSIS, BITS, Pilar

Sundar B

#### TASK SCHEDULING — GREEDY ALGORITHM: TIME COMPLEXITY

```
Algorithm Schedule(T)
m = 0 // number of machines
while (T not empty) {
  let j be the task with the earliest
  start time s<sub>i</sub>;
  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
```

Data Structure Choices (2):
Table for the schedule:
one slot for each machine; each slot
holds a linked list of tasks assigned:

- cost of (unordered) insertion: O(1)
- cost of checking for overlaps:
   O(|T|) in the worst case (per iteration)