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:
    - Suppose 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
    - Thus 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_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
     }
}
```

Time Complexity:

Loop: |T| iterations

Cost of each iteration?

Task Scheduling – Greedy Algorithm: Time complexity

Algorithm Schedule(T)

m = 0 // number of machines

while (*T not empty*) {

let j be the task in T 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

}

}

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|)

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_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

}

}

Time Complexity: O(|T|*|T|)

Data Structure Choice (2):

Use a table for the schedule:

- one slot for each machine;
- each slot holds a linked list of tasks assigned:

Cost of

- (unordered) insertion: O(1)
- checking for overlaps: O(|T|)

in the worst case (per iteration)