

University of Delaware -- Computer and Information Science

CIS621 -- Homework 3

Handed out: October 4, 2011
Due date: October 18, 2011

7. [*Individual Problem*] Consider the following game: I think of a positive integer: n , which you are to determine by guessing numbers and being told whether each guess is too high or too low or is n . There is no apriori upper bound on n -- I am free to pick any n .

- Give a strategy for determining n using at most $2\log n + c$ guesses (c is a constant).
- Give a strategy for determining n using at most $\log n + 2\log \log n + k$ guesses (k is a constant).

8. [*Group Problem*]

Suppose you are given the following **task system**:

- a set of *unit execution time tasks* $\{T_1, \dots, T_n\}$
- a set of *precedence constraints* each of which is of the form $T_i < T_j$
Here, task T_i is said to be a **predecessor** of task T_j .
- a *list* L of all the tasks where if $T_i < T_j$ then T_i appears before T_j in that list
- 3 *processors*

A **schedule** for such a task system is an assignment of tasks to processors and times such that:

- If $T_i < T_j$ then task T_i must complete execution before T_j begins execution
- At most 3 tasks execute at any given time

A **time slot** in a schedule is **available** if fewer than 3 tasks have been assigned to execute in that time slot.

A **list schedule** for such a task system is a schedule generated from the list L as follows:

while L is not empty **do**

 Remove the first task T_j from L

 Schedule T_j to execute in the earliest available time slot such that all predecessors of T_j have completed execution prior to that time slot.

Design an efficient algorithm for producing a list schedule for a task system. You will need to check with Professor Lloyd to see if your algorithm is "efficient enough".

9. [*Group Problem*]

[Brassard and Bratley, *Algorithmics, Theory and Practice*, Prentice Hall, 1988]

Given an array A of n elements, a value x is a **majority element** of A if more than half of the elements of A have the value x . Note that there can be at most one majority element in an array.

Give a linear time algorithm for determining whether or not an array has a majority element. Assume that the only comparisons allowed between values are tests of equality. That is, you may not assume that an order relation exists between elements.