Exercise R-5.2, p. 282

May 16, 2002

Exercise: Describe how to implement the TaskSchedule method to run in $O(n \log n)$ time where $n$ is the number of tasks to schedule.

Augmented TaskSchedule Algorithm (from p. 261)

Algorithm TaskScheduler(T):
Input: A set $T$ of $n$ tasks, such that each task has a start time $s_i$ and a finish time $f_i$
Output: A nonconflicting schedule of tasks in $T$ using a minimum number of machines

$m \leftarrow 0$ // Optimal number of machines.
$T \leftarrow$ a heap with earliest start time at top of heap
$S \leftarrow$ empty heap
while (heap $T$ not empty)
    remove from $T$ the task $i$ with earliest start time $s_i$
    if there is a machine $j$, $1 \leq j \leq m$, with no task conflicting with task $i$ then
        schedule task $i$ on machine $j$
    else
        $m \leftarrow m + 1$
        schedule task $i$ on machine $m$

Answer: The key here is to use two heaps: The first, $T$, to represent the set of tasks to be scheduled as shown in the augmented algorithm description above. The second, $S$, represents the machines on which tasks have been scheduled and the ending time of the last scheduled task on each machine, as shown in the code below.

The formation of the heap can be done in time $\leq n \log n$ as was shown when we discussed the heap data structure. The while loop is executed $n$ times (once for each task to schedule). The removal of the top element from the heap (which will be the remaining meeting with the earliest start time) takes time $\leq \log n$. 

1
Now we have to quickly find a machine, if any, that does not have a conflicting task. To do this we need another heap, call it $S$, that contains a pair $(j, F_j)$ where $j$ is the index of the machine $m_j$ and $F_j$ is the ending time of the currently last scheduled task on machine $m_j$. Organize this heap so that the element on top is the pair $(j, F_j)$ where $F_j \leq F_k$ for all pairs $(m_k, F_k)$ on the heap. Thus, if $s_i \geq F_j$, schedule task $s_i$ on $m_j$; otherwise another machine is needed. Thus, here is the code for the if-then-else clause that appears with the while loop.

```plaintext
// Schedule task $(s_i, f_i)$ on an appropriate (possibly new) machine.
(j, F_j) <- top element of heap S (top element is deleted from S)
if $(s_i >= F_j)$ then
    add $(j, f_i)$ to heap S
else
    m <- m+1
    add $(m, f_i)$ to heap S
```

A single execution of this if-then-else can be done in time $\leq \log n$ and, hence, all $n$ executions can be done in time $\leq n \log n$.

B. F. Caviness
Univ. of Delaware