COL351: Analysis and Design of Algorithms
Instructor: Ragesh Jaiswal

1. Is the factor-2 approximation for the greedy algorithm for the Minimum Makespan problem shown in class tight?

2. Recall the Minimum Makespan problem discussed in class. We saw a greedy approximation algorithm and showed an approximation guarantee of 2. Consider the following slightly modified algorithm:

```
GreedySortMakespan
  - Consider jobs in decreasing order of duration
  - While all jobs are not assigned
      - Assign the next job (as per the order defined) to a machine with least load
```

Let OPT denote the value of an optimal solution and G be the value of the above greedy solution. Then we will argue that $G \leq (4/3) \cdot OPT$. WLOG, let us assume that $d(1) \geq d(2) \geq ... \geq d(n)$.

Let us call an problem instance $(d(1), ..., d(n))$ *nice* if as per the above greedy algorithm, the job with the maximum finishing time is n. We will first argue that it is sufficient to analyze the approximation guarantee for nice instances.

Claim 1.1: If our greedy algorithm gives a factor c approximation for nice instances, then it gives factor c approximation for all instances.

We can now focus on only nice instances. Again, for any nice instance, we use G to denote the value of the greedy solution and OPT to denote the value of an optimal solution. Let us break the analysis into two cases: (i) $d(n) \leq OPT/3$, and (ii) $d(n) > OPT/3$. For case (i), we can make the following claim:

Claim 1.2: If $d(n) \leq OPT/3$, then $G \leq (4/3) \cdot OPT$.

Now, consider the case when $d(n) > OPT/3$. We first show that $n \leq 2m$.

Claim 1.3: If $d(n) > OPT/3$, then $n \leq 2m$.

Once, we realize the above the next thing we show is that in this case, the greedy algorithm gives an optimal solution.

Claim 1.4 If $d(n) > OPT/3$, then `GreedySortMakespan` returns an optimal solution.

Combining all the above claims, we get that the above greedy algorithm gives an approximation guarantee of (4/3).

3. Given a sorted array A containing n distinct integers, design an algorithm to determine if there is an index i such that $A[i] = i$.