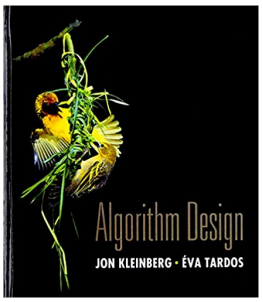
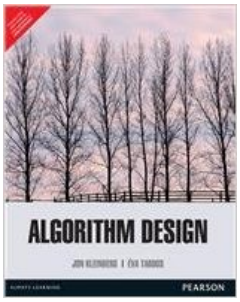
Advanced Algorithms
Exercises 1
DI – NOVA FCT





[Kleinberg and Tardos 2005]
Jon Kleinberg and Éva Tardos.
Algorithm Design.
Pearson (hardcover), 2005.
Pearson (paperback), 2013.

1. [Kleinberg and Tardos 2005] You are asked to consult for a business where clients bring in n jobs each day for processing. Each job has a processing time t_j that is known when the job arrives. The company has a set of ten machines and each job can be processed on any of these machines.

   At the moment the business is running algorithm greedyLB1 (on page 3 of adAlg02-greedy-part1.pdf). They have been told that this may not be the best approximation algorithm and they are wondering if they should be afraid of bad performance. However, they are reluctant to change the scheduling as they really like the simplicity of the current algorithm: jobs can be assigned to machines as soon as they arrive, without having to defer the decision until later jobs arrive.

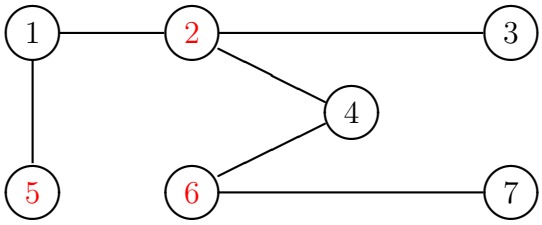
   In particular, they have heard that this algorithm can produce solutions with makespan as much as twice the minimum possible. But their experience with the algorithm has been quite good. They have been running it each day for the last month and they have not observed it to produce a makespan more than 20 percent above the average load (which is $\frac{1}{10}\sum_{j=1}^{n} t_j$).

   To try to understand why they do not seem to be encountering this factor-of-two behaviour, you ask a bit about the kind of jobs and loads they see. You find out that the sizes of jobs range between 1 and 50 (that is, $1 \leq t_j \leq 50$, for every $j = 1, \ldots, n$) and the total load ($\sum_{j=1}^{n} t_j$) is quite high each day: it is always at least 3 000.

   Prove that, on the type of inputs the company sees, greedyLB1 will always find a solution whose makespan is less than 17 percent above the average load.

2. A *vertex cover* of an undirected graph $G = (V, E)$ is a subset C of vertices that "touches" every edge: $C \subseteq V$ and $\forall (v, w) \in E : \; v \in C$ or $w \in C$.
In the **Minimum Vertex Cover Problem** we are given an undirected graph and the goal is to find a vertex cover with minimum size.

   For example, $\{2, 5, 6\}$ is an optimal solution of the instance depicted in the figure below.



   (a) Consider algorithm **greedyVC1**.

```
greedyVC1( (V, E) )
    let inCover[v] be true iff v ∈ cover
    cover = ∅
    for every v ∈ V
        inCover[v] = false
    for every (v, w) ∈ E
        if not inCover[v] and not inCover[w]
            aux = min(v, w)
            cover.add(aux)
            inCover[aux] = true
    return cover
```

   Is **greedyVC1** a ρ-approximation algorithm for the Minimum Vertex Cover Problem for some constant ρ? Justify your answer.

   (b) Now consider algorithm **greedyVC2**.

```
greedyVC2( (V, E) )
    let inCover[v] be true iff v ∈ cover
    cover = ∅
    for every v ∈ V
        inCover[v] = false
    for every (v, w) ∈ E
        if not inCover[v] and not inCover[w]
            cover.add(v)
            inCover[v] = true
            cover.add(w)
            inCover[w] = true
    return cover
```

   Prove that **greedyVC2** is a 2-approximation algorithm for the Minimum Vertex Cover Problem.

   Hint: Consider the set of edges whose both endpoints are added into the cover set.