Tutorial Exercise 11: Approximation Algorithms

1. **Q5, Chp 11, Kleinberg and Tardos.** The first algorithm presented in the lecture notes for Load Balancing is an on-line algorithm, that is, the jobs can be processed as soon as they arrive. We refer to it as the **FirstArrival** algorithm. While the LPT algorithm obtains a better approximation ratio, it does not have this on-line property. Here we show that the **FirstArrival** algorithm can have an approximation ratio that is less than the worst case of 2 if the mix of jobs that it is asked to schedule is somehow restricted.

   For example, suppose that you have 10 machines, and need to schedule $N$ jobs, where the $n^{th}$ such job takes time $t_n$. In addition, you know the total time required for all the jobs is $T = \sum_{n=1}^{N} t_n = 3000$, and the time required for each individual job satisfies $1 \leq t_n \leq 50$. Show that the approximation ratio of FirstArrival for this mix of jobs is no worse than $7/6$.

2. **Q10, Chp 11, Kleinberg and Tardos.** Suppose you are given a weighted graph $G = (V, E, w)$ where $G$ has the form of an $n \times n$ grid graph (see figure below). Assume the weights $w(v)$ are non-negative integers.

   Prof. Jot proposes the following greedy algorithm for obtaining an approximate solution to the maximally weighted independent set problem for this type of graph:

   $[S] = \text{wIndSet}(V, E, w)$

   Initialize $F \leftarrow (V, E)$ and $S \leftarrow \emptyset$.

   While the graph $F$ is not empty:

   Find a vertex $u$ in $F$ with the largest weight $w(u)$.

   $S \leftarrow S \cup \{u\}$

   Update $F$ by deleting the vertex $u$ and all its neighbouring vertices $v$ (i.e., all vertices $v$ with an edge $(u, v)$ still in $F$), and delete all the edges ending at any of these deleted vertices.

   End while

   return $S$

   (a) Write a loop invariant for the above code that is useful for proving that the set $S$ returned by $\text{wIndSet}$ is an independent set for the graph $G$.

   (b) Prove the loop invariant in part (a), and that the returned set $S$ is an independent set for the graph $G$.

   (c) Show that $w(S) = \sum_{v \in S} w(v)$ is at least $(1/4)w(S^*)$, where $S^*$ is an independent set of $G$ with the maximum possible weight $w(S^*)$.

3. **Modified Q3 Chp 11 Kleiberg and Tardos.** Suppose you are given a list of $N$ integers $L = [a_1, a_2, \ldots, a_N]$, and a positive integer $C$. The problem is to find a subset $S \subseteq \{1, 2, \ldots, N\}$ such that

   \[ T(S) = \sum_{i \in S} a_i \leq C, \]

   and $T(S)$ is as large as possible.
(a) Prof. Jot proposes the following greedy algorithm for obtaining an approximate solution to this maximization problem:

\[
[S] = \text{maxBoundedSetSum}([a_1, \ldots, a_N], C)
\]

Initialize \( S \leftarrow \{ \}, T = 0 \)

For \( i = 1, 2, \ldots, N \):
  
  If \( T + a_i \leq C \):
  
  \( S \leftarrow S \cup \{i\} \)
  
  \( T \leftarrow T + a_i \)

End for

return \( S \)

Show that Prof. Jot’s algorithm is not a \( \rho \)-approximation algorithm for any fixed value \( \rho \). (Use the convention that \( \rho > 1 \).)

(b) Describe a 2-approximation algorithm for this maximization problem that runs in \( O(N \log(N)) \) time.