Tutorial 7:

1. Consider the Max-Cut Problem which can be described as follows.

   Suppose $G = (V, E)$ is an undirected graph. (X, Y) is said to be a cut of G, if (X, Y) is a partition of V. That is, $X \cap Y = \emptyset$ and $X \cup Y = V$.

   Size of a cut (X, Y) of G, is $|\{(v, w) \mid v \in X \bigwedge w \in Y \bigwedge (v, w) \in E\}|$. That is, size of a cut (X, Y) is the number of edges in G which connect X and Y.

   Max-Cut problem is to find a cut with maximum size. This optimization problem can be shown to be NP-Hard.

   Below is an approximation algorithm for Max-Cut.

   For the following, let CutSize(X, Y) denote the size of the cut (X, Y) in the input graph G.

   App_MAXCUT
   Input: $G = (V, E)$, an undirected graph with vertex set V and edge set E.
   Output: a cut (X, Y) of G. (* with reasonably good size compared to the maxcut *)
   1. Let $X = \emptyset$, $Y = V$.
      Done = False.
   2. While not Done do
      (* Intuitively, each iteration of while loop checks, whether moving some vertex from one of the partitions to another, increases the size of the cut. If so, then one such vertex is moved. If there is no such vertex, then the procedure halts. *)
      If there exists a $v \in X$ such that CutSize$(X - \{v\}, Y \cup \{v\}) >$ CutSize(X, Y), Then
      Let $X = X - \{v\}$, $Y = Y \cup \{v\}$.
      Else If there exists a $w \in Y$ such that CutSize$(X \cup \{w\}, Y - \{w\}) >$ CutSize(X, Y), Then
      Let $X = X \cup \{w\}$, $Y = Y - \{w\}$.
      Else Done = True.
      EndWhile
   3. Output (X, Y).
   End App_MAXCUT

   Show that the above alogrithm gives a cut which is of size within a factor of 2 of optimal.

2. Recall the Multiprocessor Scheduling problem which is **NP**-complete. Thus the corresponding problem of finding an optimal schedule S (which minimizes $\mathbf{T}(S)$, the time taken by schedule S) is **NP**-hard. Consider the following approximation algorithm for Multiprocessor Scheduling. Intuitively the idea is to schedule the tasks in order of decreasing length (time taken to execute the task), where the tasks are assigned to the processors in rotating order.

   Multi_Schedule (A, ℓ, m)
   (* A is a set of n tasks. ℓ is an array, where $\ell(a)$ denotes the time taken by task a. m is the number of processors. *)

1. Sort all the tasks based on non-increasing order of $\ell(\cdot)$.
Say the sorted order is $a_0, a_1, \ldots, a_{n-1}$, where $\ell(a_i) \geq \ell(a_{i+1})$, for $i < n-1$.
2. For $0 \leq i < m$, let $A_i = \{a_j \mid j < n \text{ and } (j \bmod m) = i\}$
3. $S = (A_0, A_1, \ldots, A_{m-1})$ is the schedule for the m processors, where A_i is the set of tasks assigned to processor i.

End

Show that the above algorithm is a good approximation algorithm by showing that the completion time for the schedule generated by the above algorithm is no worse than twice the completion time of the optimal schedule.

Hint: Consider the load due to the longest task and the rest of the tasks.

3. Recall that the satisfiability problem is:

INPUT: A set $V = \{x_1, x_2, \ldots, x_n\}$ of variables, and a set $C = \{c_1, c_2, \ldots, c_m\}$, of clauses, where each c_i is a disjunction of some literals.
(x_j and $\overline{x_j}$ are called literals, where $\overline{x_j}$ denotes the negation of x_j).
QUESTION: Is there a truth assignment to the variables such that all the clauses are satisfied. That is, for each clause c_i, there exists a j such that [(x_j is true and x_j is a literal in c_i) or (x_j is false and $\overline{x_j}$ is a literal in c_i)].

The corresponding optimization problem is to find a truth assignment to the variables which maximizes the number of clauses that can be satisfied. Consider the following approximation algorithm for this problem.

ApproxSAT(V, C)
(* V is a set of variables, and C is a set of clauses over V. Assume without loss of generality that no clause contains both x and $\overline{x}$ for any variable x in V. Furthermore assume that no clause contains multiple occurrence of the same literal.*)
1. Let $CLeft = C$.
(* Intuitively, $CLeft$ gives the clauses yet to be satisfied. *)
2. For $i = 1$ to n to do
(* Intuitively, we assign value to variable x_i in this loop. *)
If the number of clauses in $CLeft$ that x_i appears in is more than the number of clauses in $CLeft$ in which $\overline{x_i}$ appears in, then
Assign x_i to be true.
Let $CLeft = CLeft - \{c : c$ is a clause in $CLeft$ which contains $x_i\}$.
Otherwise,
Assign x_i to be false.
Let $CLeft = CLeft - \{c : c$ is a clause in $CLeft$ which contains $\overline{x_i}\}$.
End while

End

Prove that the above is a good approximation algorithm by showing that
$\frac{Alg}{Opt} \geq \frac{k}{k+1}$, where

k is the minimal number of distinct literals in any of the clauses in C (that is, each clause in C has at least k distinct literals),
Opt is the number of clauses that can be satisfied by an optimal truth assignment to the variables, and
Alg is the number of clauses that the above algorithm satisfies.