# Advanced Analysis of Algorithms - Final

K. Subramani
LCSEE,
West Virginia University,
Morgantown, WV
{ksmani@csee.wvu.edu}

## 1 Instructions

1. Attempt as many problems as you can. You will be given partial credit.
2. The duration of this final is 2 hours, i.e., 3 : 00-5 : 00 PM.
3. Each question is worth 6 points for a total of 30 points.

## 2 Problems

1. Let $\mathbf{A}[1..n]$ be an array of $n$ distinct numbers. If $i < j$ and $A[i] > A[j]$, then pair $(i, j)$ is called an inversion of $\mathbf{A}$. Design an algorithm that takes as input such an array and outputs the *number* of inversions of the array. Your algorithm should run in time $O(n \cdot \log n)$ in the worst case. (*Hint: Merge-Sort.*)
2. Let $P$ be a convex polygon in 2-dimensional space, having $n$ vertices. A triangulation of $P$ is an addition of diagonals connecting the vertices of $P$, so that each interior face is a triangle. The weight of a triangulation is the sum of the lengths of the diagonals. Assuming that we can compute lengths, add, and compare them in constant time, give an efficient algorithm for calculating the minimum weight triangulation of $P$. (*Hint: Dynamic Programming.*)
3. Let $\mathcal{T} = \{T_1, T_2, \ldots, T_n\}$ denote a collection of $n$ tasks. Task $T_i$ has start time $s_i$ and finish time $f_i$, with $s_i \leq f_i$, i.e., task $T_i$ *must* start at time $s_i$ and it will finish at time $f_i$. Two tasks $T_i$ and $T_j$ are non-conflicting, if either $f_i \leq s_j$ or $f_j \leq s_i$. The tasks in $\mathcal{T}$ are to be assigned to machines, so that the resultant schedule is non-conflicting. Design an algorithm that schedules the tasks in $\mathcal{T}$ using the *fewest* number of machines. Clearly, I can obtain a non-conflicting schedule, by assigning each job to a different machine! (*Hint: Use a greedy strategy that sorts the tasks in $\mathcal{T}$ by their start times.*)
4. In class, we showed that the MAX2SAT problem is `NP-complete`. What is the complexity of the MAX1SAT problem? Either design a polynomial time algorithm for MAX1SAT or show that the problem is `NP-complete`.
5. The Satisfiability problem (SAT) is concerned with a finding a satisfying assignment to a conjunction of clauses. $kSAT$ is defined as the restriction of SAT in which each clause has exactly $k$ literals. HornSAT is the restriction of SAT in which each clause is `Horn`, i.e., each clause has at most one positive literal. In class, we showed that 3SAT is `NP-complete`, whereas, 2SAT and HornSAT are decidable in polynomial time. The HornSAT$\oplus$2SAT problem is the restriction of SAT in which each clause is either Horn or has exactly 2 literals. Argue that the HornSAT$\oplus$2SAT problem is `NP-complete`. (*Hint: Use a reduction from 3SAT.*)