# CS 473: Algorithms, Fall 2016
# HW 11 (for self study only)

---

This homework will not be graded.
However, material covered by this homework may appear on the final exam.

---

1. See Jeff's homework 11 from Spring 2016. `https://courses.engr.illinois.edu/cs473/sp2016/hw/hw11.pdf`

2. In the Max-SAT problem we are given a SAT formula $\varphi$ and the goal is to find an assignment that satisfies the maximum number of clauses. Consider an oblivious randomized algorithm that sets each variable independently to TRUE with probability exactly $1/2$.

   - Suppose the formula is a $k$-SAT formula where each clause has exactly $k$ distinct literals. What is the expected number of clauses satisfied by a random assignment? Interestingly for 3-SAT, unless $P = NP$ the ratio provided by this simple algorithm cannot be improved!
   - Prove that for a general SAT formula, the expected number of clauses that are satisfied is at least $m/2$ where $m$ is the number of clauses.

3. We saw an LP based 2-approximation for weighted Vertex Cover. Write an LP relaxation for weighted Set Cover. Recall that we are given $m$ sets $S_1, S_2, \ldots, S_m$ over a universe of size $n$ and the goal is to find a minimum weight sub-collection of the sets which together cover the universe. Obtain a $k$-approximation for instances in which each element is contained in at most $k$ sets. Note that Vertex Cover is the special case when $k = 2$.

4. Consider the LP relaxation for Set Cover from the previous problem. Let $x_i$ be the variable in the relaxation for set $S_i$. Suppose $x^*$ is an optimum solution to the LP relaxation. Define $y_i = \min\{1, 2\ln n \cdot x_i^*\}$ for each set $S_i$. Pick each set $S_i$ independently with probability $y_i$.

   - Prove that the expected weight of the sets chosen is at most $2\ln n \cdot OPT$.
   - Prove that the probability that any fixed element in the universe is *not* covered by the chosen sets is at most $1/n^2$.
   - Prove that, with probability at least $1 - 1/n$ all the elements of the universe are covered by the chosen sets. *Hint:* Use union bound.
   - Prove that with probability $1/2 - 1/n$ the algorithm outputs a set cover for the universe whose weight is at most $4\ln n \cdot OPT$ where $OPT$ is the weight of an optimum Set Cover. *Hint:* Use Markov's inequality.

5. In the Metric-TSP problem the goal is to find a minimum cost tour in a metric $(V, d)$ that visits all the vertices. We saw Christofides's heuristic that gives a 3/2-approximation. Now consider the $s$-$t$ TSP-Path problem in a metric space $(V, d)$. Here the goal is to find an $s$-$t$ walk of minimum cost that visits all the vertices. This differs from the tour version in that one does not need to come back to $s$ after reaching $t$.

- Give an example to show that the TSP tour can be twice the cost of a TSP Path. Also show that TSP tour is always at most twice the cost of a TSP path.
- Obtain a simple 2-approximation for the TSP-Path problem via the MST heuristic.
- **Hard:** Obtain a 5/3-approximation for the TSP-Path problem by modifying the Christofides heuristic appropriately.

6. **Hard:** Consider the load balancing problem we discussed in lecture. One can obtain a $(1+\epsilon)$-approximation in polynomial time for any fixed $\epsilon > 0$. The goal of this problem is to give you an outline of this algorithm. Suppose we knew the optimum load is $\alpha^*$. Partition the jobs into "large jobs" $L$ which consists of all jobs which are bigger than $\epsilon\alpha^*$ and "small jobs" $S$ which consists of all jobs which are smaller than $\epsilon\alpha^*$.

   - Suppose we have scheduled the big jobs $L$ first and obtained a schedule with makespan at most $(1+\epsilon)\alpha^*$. Describe an adaptation of the greedy list scheduling we discussed in class to schedule the small jobs on top of the schedule for big jobs, and show that the resulting makespan is at most $(1+2\epsilon)\alpha^*$.
   - Consider the big jobs $L$. Round up each job's size to next highest power of $(1+\epsilon)$. That is, if a job's size is between $(1+\epsilon)^i$ and $(1+\epsilon)^{i+1}$ we treat it as a job of size $(1+\epsilon)^{i+1}$.
     - Show that the number of *distinct* job sizes that remain after the rounding is $O(1/\epsilon^2)$.
     - Describe a dynamic programming based algorithm to find an optimum schedule for the rounded up jobs — recall that we saw a special case of this for 3 job sizes in homework 2. What is the running time of your algorithm?
     - Prove that if there is a schedule of makespan $\alpha^*$ for the original big jobs then there is a schedule of makespan at most $(1+\epsilon)\alpha^*$ for the rounded up big jobs.
   - Can you put the ingredients together to obtain a $(1+\epsilon)$-approximation in $n^{\text{poly}(1/\epsilon)}$ time? In particular, you also need to show how to guess $\alpha^*$ via binary search. This last step may be a bit hard.