INTRO TO APPROXIMATION, CLASS 4

greed and SAT

EXERCISE ONE Consider SCHEDULING WITH DEPENDENCIES: we schedule jobs of different lengths on m computers (m is a part of the input), but we also have a *dependence graph* on the jobs, and we can schedule a job only when all its dependencies are completed.

1. Prove the following lower bound on the optimum:
   "$OPT \geq$ length of any chain in the input. A *chain* is a sequence of jobs where each one depends on the previous one. Its length is then the total processing time of all the jobs in the chain."
2. Design a greedy 2-approximation algorithm for this problem.

EXERCISE TWO Consider the classic NP-hard KNAPSACK PROBLEM, where we have n objects $a_1, \dots, a_n$, each object has a weight w_i and cost c_i, and our bag has a weight limit of B.

1. Explain why "naive greedy algorithm", i.e. "we put the most expensive item (that fits) into the knapsack and continue the same way" is a bad one.
2. OK, let us try the following: "we sort the items according to their density (ratio price/size), go through them in decreasing order and insert only those that fit in the knapsack."
   Spoiler alert: this algorithm also fails. Show an input where it does.
3. Finally, design a 2-approximation algorithm for this problem. This algorithm does not need to be greedy.
   Hint: When you iterate over the items based on the density, at some point it may happen that P does not fit with the items you have already selected into the knapsack. What should you do then?

EXERCISE THREE You may recall MAX SAT from the last exercise session, where we formulated a randomized approximation algorithm for it. This algorithm was effective for clauses of length 2 or more, but when there were too many clauses of type (x_i) or $(\neg x_j)$, it was only a 1/2-approximation.

Let us prove that we can assume that the input is a little bit nicer:

1. Prove the following: Suppose we have a c-approximation algorithm for a subset of MAX SAT – it only works on inputs which contain no negative mono-clauses like $(\neg x_i)$. Then we can transform it into a c-approximation algorithm for MAX SAT on all inputs.
2. Prove that the same holds for WEIGHTED MAX SAT, where each clause has weight w_i and we maximize the weighted sum of satisfied clauses, i.e. $\max \sum_i w_i C_i$.

EXERCISE FOUR We have learned from the previous exercise that it suffices to deal with MAX SAT on inputs that contain no negative mono-clauses like $(\neg x_i)$. We should use this fact to choose a better probability p, which we use in the randomized algorithm for setting a variable to 1:

1. Prove that if all variables x_i are randomly set to be true with probability $p > \frac{1}{2}$, then the probability of satisfying a clause is at least $\min(p, 1 - p^2)$.
2. Choose a good p and finish the analysis of the suggested randomized algorithm for MAX SAT.