Describe and analyze (prove correct and give a time analysis) algorithms for any three of the following four problems.

Algorithms may be described at "high level" without actual code. You may use without further proof any algorithm or data structure from the text or in class, and their correctness and analysis, but be careful. For example, if you construct a graph with $n^2$ nodes and $n^3$ edges and then run Dijkstra’s algorithm on the resulting graph, the total run-time is $O(n^3 \log n)$, because Dijkstra’s algorithm is $O(E \log V)$. If you want to use the correctness of Dijkstra’s algorithm on the graph, you must check that edge weights are positive, since otherwise the proof that Dijkstra’s algorithm gives shortest paths does not go through.

If a problem has multiple size parameters, you should express the run-time as a function of all relevant parameters; e.g., saying maximum bi-partite matching takes time $O(|V||E|)$ is more accurate than to say it takes time $O(V^3)$, although both are correct.

For some problems, correctness may be trivial; for others the analysis may be trivial. If it is trivial, you do not need to go into detail, but you should at least give a one or two line explanation. (Conversely, if you only give a one or two line explanation, I will view this as implicitly claiming that it is trivial.)

Each problem is worth 10 points. Grading may be based on any of the following that are relevant for the problem: the efficiency of your algorithm; the correctness and proof thereof; and the time analysis. You only get credit for what you state and prove: if you give a time analysis of $O(n^2)$ for your algorithm, and the algorithm is in fact $O(n \log n)$, your score for efficiency will be based on your claimed $O(n^2)$. The number of points depending on each part is given after the problem, as well as a ballpark estimate of the time analysis for my solutions for some of the problem.

**Maximum minimum grade** On the homework, we looked at the problem of dividing $H$ hours between $n$ assignments so as to maximize the total grade on all assignments. Recall that for each assignment $i$, we had as input a non-decreasing function $f_i(h)$, where $0 \leq h \leq H$ is an integer, and $f_i(h)$ is the grade for assignment $i$ if we spend $h$ hours on it. $f_i(0) = 0$ for each assignment. You needed to, given $H$, find integral number of hours $h_i$ to spend on each assignment $i$, so that $\sum h_i = H$. Before, the objective was to maximize the total grade, $\sum f_i(h_i)$.

Instead, consider the objective to maximize the lowest grade on any assignment, $\text{mingrade}(h_1, h_n) = \min_i f_i(h_i)$. You will still spend an integral number of hours on each assignment, and the total hours must sum to $H$. (2 points correct poly. time algorithm, 4 pts correctness proof, 4 points efficiency and time analysis. My best time is $O(H \log n)$. )
Max min matching  The input is a bipartite graph $G = (L, R, E)$ where $|L| = |R| = n$, and non-negative edge weights $w(e) \geq 0$ for $e \in E$. You want to find a perfect matching $M$ of $G$ that maximizes $Value(M) = \min_{e \in M} w(e)$. (3 points correct poly time algorithm, 4 points correctness proof, 3 points efficiency. I know an $O(n|E|)$ time algorithm.)

Monotone matching  Let $G = (L, R, E)$ be a bipartite graph with $|L| = |R| = n$, and where the nodes in $L$ are ordered as $u_1...u_n$ and the nodes in $R$ as $v_1...v_n$.

Then a matching $M$ is monotone if whenever $u_i$ is matched to $v_j$, and $u_{i'}$ is matched to $v_{j'}$, if $i < i'$ then $j < j'$. The problem is: given such a bipartite graph $G$, find the maximum size of a monotone matching in $G$. (5 points correct poly time algorithm, 3 points correctness proof, 2 points efficiency and time analysis. My algorithm is $O(n^2)$.)

Approximate bin packing  In the bin packing problem, you have an unlimited number of bins each of size $B$ and a set of items, with sizes $s_1...s_n$, $0 \leq s_i \leq B$. You need to assign each item to a bin so that for every bin, the total sizes of items assigned to that bin is at most $B$. The objective is to minimize the number of bins that are used, i.e., that have any items assigned to them. Give an approximation algorithm for this problem with a constant approximation ratio. (4 points, small ratio, 4 points correctness proof, 2 points efficiency. I know an algorithm with ratio $2$. This is related to the bin-filling problem on homework 2, but not identical. In some sense, it is the dual problem, because we need to have at most the bin size items in each bin, rather than at least the size. Also, before the bins had variable sizes, here they all have the same size.)