CSE 202 Final Exam
Spring, 2004
Due Thurs., June 10

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 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 $O(n^3 logn)$,because Dijkstra's algorithm is $O(ElogV)$. 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 of the problem.

**Coffee cart profits** A coffee cart has $k$ grades of caffeination that they serve. Customers at a coffee cart always buy the highest grade of caffeine that fits into their coffee budgets. With an extensive market survey, the cart has a complete list of the customers' coffee budgets, $CB[1..n]$. The cart wants to determine the prices for each grade of coffee that will maximize their revenues. If the prices are $p_1 \leq p_2 ... \leq p_k$, then the revenue will be $\sum_{i=1}^{i=k} p_i |\{j | p_i \leq CB[j] < p_{i+1}\}|$ where we view $p_{k+1}$ as infinity. Give an efficient algorithm for determining the optimal list of prices. [6pts correct poly-time algorithm, 2pts explanation of correctness, 2pts efficiency].

**Weighted maximum independent set in a tree** Consider a rooted tree $T$ (not necessarily binary) where each node $u$ has a non-negative weights $w(u) \geq 0$ . Give an efficient algorithm for finding the independent set $S$ in $T$ of maximum total weight, $\sum_{u \in S} w(u)$ Assume $T$ is given by having a parent pointer for each node. (4 pts correct poly-time algorithm, 2 pts correctness explanation, 4 pts. efficiency)

**Load balancing** You have a list of $n$ jobs and $m$ machines, and for each job, a subset of machines that can perform the job. You need to assign each job to a machine that can perform it. Your objective is to minimize the maximum number of jobs assigned to any one machine. (4 pts correct algorithm, 3 pts correctness proof, 3 pts efficiency)

**Filing clerk** There is a long row of pigeonholes labelled as integers from 1 to $P$. A filing clerk will need to access the pigeonholes $p_1, ..p_n$ in that order. Note that the pigeonholes she needs to access are not necessarily consecutive, sorted, or distinct. (See example.) Standing in front of hole $j$, the clerk can reach any hole between $j - r$ and $j + r$. Give an efficient algorithm to determine the minimum number of times the clerk needs to move in order to access the sequence of holes. (The clerk can start standing in front of any pigeonhole the algorithm chooses, and every time the clerk moves, it is to a new location the algorithm chooses. We count a move of any distance as a single move.)

Example: $r = 3, n = 10, P = 20, p[1..10] = 7, 10, 9, 4, 1, 13, 19, 16, 13, 11$. One solution is: start in position 7. Service the first four requests (7,10,9,4) Move to position 1, service request 5. Move to postion 16, service the next 4 requests. Move to position 11, and finish. Cost=4 positions. (4pts correct algorithm, 5 pts correctness proof, 1 pt efficiency)