These are the standard 10-point rubrics that we will use for certain types of exam questions. When these problems appear in the homework, a score of x on this 10-point scale corresponds to a score of $\lceil x/3 \rceil$ on the 4-point homework scale.

Proof by Induction

- 2 points for stating a valid **strong** induction hypothesis.
  - The inductive hypothesis need not be stated explicitly if it is a mechanical translation of the theorem (that is, "Assume $P(k)$ for all $k < n$" when the theorem is "$P(n)$ for all n") *and* it is applied correctly. However, if the proof requires a stronger induction hypothesis ("Assume $P(k)$ and $Q(k)$ for all $k < n$") then it must be stated explicitly.
  - By course policy, ***stating a weak inductive hypothesis triggers an automatic zero***, unless the proof is otherwise *perfect*.
  - ***Ambiguous*** induction hypotheses like "Assume the statement is true for all $k < n$." are not valid. *What* statement? The theorem you're trying to prove doesn't use the variable k, so that can't possibly be the statement you mean.
  - ***Meaningless*** induction hypotheses like "Assume that k is true for all $k < n$" are not valid. Only propositions can be true or false; k is an integer, not a proposition.
  - ***False*** induction hypotheses like "Assume that $k < n$ for all k" are not valid. The inequality $k < n$ does *not* hold for all k, because it does not hold when $k = n + 5$.
- 1 point for explicit and clearly exhaustive case analysis.
  - No penalty for overlapping or redundant cases. However, mistakes in redundant cases are still penalized.
- 2 points for the base case(s).
- 2 point for correctly applying the *stated* inductive hypothesis.
  - It is not possible to correctly apply an invalid inductive hypothesis.
  - No credit for correctly applying a different induction hypothesis than the one stated.
- 3 points for other details of the inductive case(s).

Dynamic Programming

- **6 points for a correct recurrence**, described either using functional notation or as pseudocode for a recursive algorithm.
  - + 1 point for a clear **English** description of the function you are trying to evaluate. (Otherwise, we don't even know what you're *trying* to do.) ***Automatic zero if the English description is missing.***
  - + 1 point for stating how to call your recursive function to get the final answer.
  - + 1 point for the base case(s). −½ for one *minor* bug, like a typo or an off-by-one error.
  - + 3 points for the recursive case(s). −1 for each *minor* bug, like a typo or an off-by-one error. **No credit for the rest of the problem if the recursive case(s) are incorrect.**
- **4 points for iterative details**
  - + 1 point for describing the memoization data structure; a clear picture may be sufficient.
  - + 2 points for describing a correct evaluation order; a clear picture may be sufficient. If you use nested loops, be sure to specify the nesting order.
  - + 1 point for running time
- Proofs of correctness are not required for full credit on exams, unless the problem specifically asks for one.
- Do not analyze (or optimize) space.
- For problems that ask for an algorithm that computes an optimal *structure*—such as a subset, partition, subsequence, or tree—an algorithm that computes only the *value* or *cost* of the optimal structure is sufficient for full credit, unless the problem says otherwise.
- Official solutions usually include pseudocode for the final iterative dynamic programming algorithm, ***but iterative psuedocode is not required for full credit***. If your solution includes iterative pseudocode, you do not need to separately describe the recurrence, memoization structure, or evaluation order. However, you ***must*** give an English description of the underlying recursive function.
- Official solutions will provide target time bounds. Algorithms that are faster than this target are worth more points; slower algorithms are worth fewer points, typically by 2 or 3 points (out of 10) for each factor of n. Partial credit is ***scaled*** to the new maximum score. All points above 10 are recorded as extra credit.

  We rarely include these target time bounds in the actual questions, because when we have included them, significantly more students turned in algorithms that meet the target time bound but didn't work (earning 0/10) instead of correct algorithms that are slower than the target time bound (earning 8/10).

Graph Reductions

For problems solved by reducing them to a standard graph algorithm covered either in class or in a prerequisite class (for example: shortest paths, topological sort, minimum spanning trees, maximum flows, bipartite maximum matching, vertex-disjoint paths, . . .):

- **1 point for listing the vertices of the graph.** (If the original input is a graph, describing how to modify that graph is fine.)
- **1 point for listing the edges of the graph**, including whether the edges are directed or undirected. (If the original input is a graph, describing how to modify that graph is fine.)
- **1 point for describing appropriate weights** and/or lengths and/or capacities and/or costs and/or demands and/or whatever for the vertices and edges.
- **2 points for an explicit description of the problem being solved on that graph.** (For example: "We compute the maximum number of vertex-disjoint paths in G from v to z.")
- **3 points for other algorithmic details**, assuming the rest of the reduction is correct.
  - + 1 point for describing how to build the graph from the original input (for example: "by brute force")
  - + 1 point for describing the algorithm you use to solve the graph problem (for example: "Orlin's algorithm" or "as described in class")
  - + 1 point for describing how to extract the output for the original problem from the output of the graph algorithm.
- **2 points for the running time**, expressed in terms of the original input parameters, not just V and E.
- **If the problem explicitly asks for a proof of correctness,** divide all previous points in half and add **5 points for proof of correctness.** These proofs almost always have two parts; for example, for algorithms that return TRUE or FALSE:
  - – 2½ points for proving that if your algorithm returns TRUE, then the correct answer is TRUE.
  - – 2½ points for proving that if your algorithm returns FALSE, then the correct answer is FALSE.

  These proofs do not need to be as detailed as in the homeworks; we are really just looking for compelling evidence that *you* understand why your reduction is correct.
- It is still possible to get partial credit for an incorrect algorithm. For example, if you describe an algorithm that sometimes reports false positives, but you prove that all FALSE answers are correct, you would still get 2½ points for half of the correctness proof.

NP-Hardness Reductions

For problems that ask "***Prove*** that X is NP-hard":

- **4 points for the polynomial-time reduction:**
  - 1 point for explicitly naming the NP-hard problem Y to reduce from. You may use any of the problems listed in the lecture notes; a list of NP-hard problems will appear on the back page of the exam.
  - 2 points for describing the polynomial-time algorithm to transform arbitrary instances of Y into inputs to the black-box algorithm for X
  - 1 point for describing the polynomial-time algorithm to transform the output of the black-box algorithm for X into the output for Y.
  - Reductions that call the black-box algorithm for X more than once are perfectly acceptable. You do *not* need to explicitly analyze the running time of your resulting algorithm for Y, but it must be polynomial in the size of the input instance of Y.
- **6 points for the proof of correctness.** **This is the entire point of the problem.** These proofs always have two parts; for example, if X and Y are both decision problems:
  - 3 points for proving that your reduction transforms positive instances of Y into positive instances of X.
  - 3 points for proving that your reduction transforms negative instances of Y into negative instances of X.

  These proofs do not need to be as detailed as in the homeworks; however, it must be clear that you have at least considered all possible cases. We are really just looking for compelling evidence that *you* understand why your reduction is correct.
- It is still possible to get partial credit for an incorrect reduction. For example, if you describe a reduction that sometimes reports false positives, but you prove that all FALSE answers are correct, you would still get 3 points for half of the correctness proof.
- Zero points for reducing X ***to*** some NP-hard problem Y.
- Zero points for attempting to solve X.

Approximation Algorithms

For problems that ask you to describe a polynomial-time approximation algorithm for some NP-hard problem X, analyze its approximation ratio, and prove that your approximation analysis is correct:

- **4 points for the actual approximation algorithm.** You do not need to analyze the running time of your algorithm (unless we explicitly ask for the running time), but it must clearly run in polynomial time. If we give you the algorithm, ignore this part and scale the rest of the rubric up to 10 points.
- **2 points for stating the correct approximation ratio.** If we give you the approximation ratio, ignore this part and scale the rest of the rubric up to 10 points.
- **4 points for proving that the stated approximation ratio is correct.** If we do not *explicitly* ask for a proof, ignore this part and scale the rest of the rubric up to 10 points.

For example, suppose we give you an algorithm and ask for its approximation ratio, but we do not explicitly ask for a proof. If the given algorithm is a 3-approximation algorithm, then you would get full credit for writing "3".