0.0.1 Designing an algorithm

“Flawed algorithms” may refer either to incorrect algorithms, or to correct algorithms that do not run in the correct time complexity.

100%: A pseudocode or English description of the algorithm that is completely correct and runs in the correct time complexity.

90%: The algorithm is incorrect due to minor errors that are not core to the functionality of the algorithm (e.g. in most cases, off-by-one errors).

50%: The algorithm is flawed in a way that cannot be salvaged with minor changes to the code, but could be salvaged given a new idea or insight. This bucket is for solutions that are on the right track but missing an important piece.

20%: There is a sensible initial attempt at a solution.

0%: The solution is missing or completely wrong.

0.0.2 Formal proofs

100%: A flawless formal proof.

80-90%: A formal proof with minor errors.

80%: A formal proof that is missing a small but important step (e.g. an induction proof with no base case, unless the base case was the hard part).

50%: A formal proof that is missing a critical step that is nontrivial to come up with.

50%: A handwavy informal justification.

20%: An English description of the algorithm we are proving the correctness of.

20%: A formal proof that is missing almost all of the important steps.

0%: A “correctness proof” for a fatally flawed algorithm.

0%: The solution is missing or completely wrong.
0.0.3 Informal explanations

100%: A handwavy justification that hits all the key points.
80%: A handwavy justification that hits most of the key points.
20%: An explanation that has some merit, but misses most of the main points.
0%: A justification for a fatally flawed algorithm.
0%: The solution is missing or completely wrong.

0.0.4 Runtime analyses of (most) algorithms

Most runtime analyses in this class are straightforward, and will not be worth very many points. Occasionally, runtime analyses will be kind of hard (like the analysis of BuildMaxHeap), and in that case they will be treated like formal proofs. (Note: generally, solving recurrences will fall under “formal proofs”.)

100%: A reasonable one-paragraph justification that accounts for all the important steps in the algorithm and leads to the correct runtime bound.
0%: A proof of an overly tight or loose runtime bound.
0%: The solution is missing or completely wrong.

0.0.5 Nontrivial runtime analyses

See “formal proofs”.

0.0.6 Mathematical calculations

100%: A correct answer that shows all important steps.
90%: An answer that shows all important steps but is incorrect due to minor arithmetic errors.
40%: An answer that is incorrect due to major errors.
20%: An initial attempt towards a solution.
20%: Correct answer with no justification.
0%: The solution is missing or completely wrong.