Recipes

Module 1
Recipe for exhaustive search:
1. Determine what the possibilities are.
2. Use the possibilities to solve the problem.

Module 2
Recipe for calculating costs:
1. Break the lines into blocks.
2. Determine the cost of each block.
3. Compute the sum of the costs.

Module 3
Recipe for a greedy algorithm:
1. Build up the solution in steps.
2. Choose the next part of the solution in a “greedy” manner.
3. Once a decision has been made, never undo it.

Module 4
Recipe for substitution method:
1. Guess an upper bound for $T(n)$.
2. Use the guess for values on the right hand side.
3. Simplify the right hand side to prove the bound.
4. Check that the bound holds for the base cases.

Module 5
Recipe for dynamic programming:
1. Characterize an optimal solution.
2. Define a subproblem in terms of smaller subproblems.
3. Determine what information should be stored in each table entry.
4. Determine the base cases.
5. Choose an order of evaluation.

6. Determine the shape of the table or tables needed to store the solutions to the smaller problems.

7. Extract the solution from the table.

Module 6
Recipe for an adversary lower bound:

1. Choose a strategy for the adversary.

2. To prove a lower bound of $T$ steps, show that after $T - 1$ steps, there are at least two possible inputs consistent with the answers that yield two different outputs.

Recipe for membership in NP:

1. Give a polynomial-size certificate for each yes-instance.

2. Give a polynomial-time verification algorithm.

3. Show that the algorithm answers “Yes” for any yes-instance and its certificate.

4. Show that the algorithm is not fooled by false certificates for any no-instances.

Recipe for NP-completeness:

1. Prove $C$ is in NP.

2. Select $B$ that is known to be NP-complete.

3. Give an algorithm to compute a function $f$ mapping each instance of $B$ to an instance of $C$ (it needn’t map to all of $C$).

4. Prove that for any string $x$, if $x$ is a yes-instance for $B$ then $f(x)$ is a yes-instance for $C$.

5. Prove that for any string $x$, if $f(x)$ is a yes-instance for $C$ then $x$ is a yes-instance for $B$.

6. Prove that the algorithm computing $f$ runs in polynomial time.

Module 7
Recipe for backtracking:

1. Specify a partial solution.

2. Determine how the children of a node are formed.

3. Choose when to backtrack.

Recipe for branch-and-bound:

1. Determine what to store at each node.
2. Decide how to generate the children of a node.

3. Specify what global information should be stored and updated.

4. Choose a bounding function.

**Module 8**

Recipe for analyzing an approximation algorithm:

1. Choose a new measure or problem.

2. Bound the approximate solution in terms of the new measure or problem.

3. Bound the optimal solution in terms of the new measure or problem.

4. Combine the two results to relate the approximate and optimal solutions.