How to Solve It: An Approach to (Algorithmic) Problem Solving

When using the following, not only are you trying to find a solution but especially in the early stages you are trying to deepen you understanding of the problem.

1. What is the problem

What is given: goal, data, situation, constraints

2. Make it concrete. Examples to deepen understanding and generate ideas
   - Write down simple examples of the problem (*toy problem*)
     - Inputs, Outputs, Constraints
   - Draw a picture of the problem
   - Think about the solutions to the toy problems
   - Is there a step by step way to solve the simple examples?
3. Are there similar or related problems
   - What approach worked for that problem?
   - How is this other problem the same or different from the problem you are trying to solve?
   - What is crucial about the differences that prevent the approach from working?
4. Test any proposed solution – be a devil's advocate!!
   - Write it down in high level pseudo code – test it on a variety of small problems
   - If an approach did not work, why did it fail?

How to Prove Algorithms Correct: Basic approaches to Proof

Proofs of correctness

1: Carefully write down exactly what you are trying to prove.

- Premises
- Conclusion
- Any underlying constraints or assumptions that may not be explicit in the premises

Direct Proof:

2: Look for links from the premises to the conclusion

- Examine what you can conclude from the premises especially any previous results
- Are there any previous results that imply the conclusion

3: Clearly write down each step of your argument –

- avoid paragraphs for formal proofs
- break the proof down into steps
- each step should make clear what its conclusion is

Indirect Proof:

Same as direct proof but now you are making an argument that either the premise or something that is known to be true is contradicted by assuming that the conclusion you are trying to prove is true is false. That is you reason from assuming that the conclusion is false to reasoning from this that either the premise or something known to be true is false.

The Principle of Mathematical Induction

When to use in computing: when trying to prove that a proposition is true for each of a sequence of steps. In computing each proposition is a usually a statement about a part of an algorithm. It can be used both for iterative and recursive algorithms,

Principle of Mathematical Induction. Let P be a property of positive integers such that:

1. Base Step: P(1) is true, and
2. Inductive Step: if P(k) is true, then P(k+1) is true.

Then P(n) is true for all positive integers.

Strong Form of Mathematical Induction. Let P be a property of positive integers such that:

1. Base Step: P(1) is true, and
2. Inductive Step: if P(i) is true for all $1 \le i \le k$ then P(k+1) is true.

Then P(n) is true for all positive integers.

The key difference is in strong mathematical induction you to prove the inductive step you can use some or all of the previous propositions to prove the next proposition.

Steps in applying

1. State base step: State what P(1) means in the context of the problem
2. Prove the base step: Prove that P(1) is true
3. State the inductive step: State $P(k) \rightarrow P(k+1)$ means in the context of the problem
4. Prove the inductive step: Prove that $P(k) \rightarrow P(k+1)$
5. Invoke the PMI – Since we have shown that P(1) is true and that $P(k) \rightarrow P(k+1)$ is true, by the Principle of Mathematical Induction P(n) is true for all $n \in Z^+$. Again P(n) is in the context of the problem

Counter Example: Proving a proposition is False

Find a **concrete** example that contradicts the proposition.

E.g. Proposition: the formula n^2-n+41 generates prime numbers for all $n \ge 1$.

This is proposition is false since if n = 41 then the result of this formula is 41^2 .

Interval Scheduling:

Proof of correctness: Earliest finish time algorithm finds a maximum subset of mutually compatible jobs

State the setup clearly: Assume $\mathbf{\Sigma^*} = \{ (s_1^*, f_1^*), ..(s_n^*, f_n^*)\}$ is an optimal solution to an interval scheduling problem with n compatible jobs specified by their start and end times and let $\mathbf{\Sigma} = \{ (s_1, f_1), ..(s_m, f_m)\}$ be the set of jobs determined by the earliest finish time algorithm for the same problem. Both sets are sorted by finish time.

Proposition: Let P(k) be the proposition $f_k \le f^*_k$: Want to show this is true for all k≥1 up to n.

(Then we will use this fact to prove that Σ is also optimal (that is has the same number of jobs as the optimal solution.)

Proof by induction:

Let P(k) be the proposition $f_k \le f^*_k$: Want to show this is true for all k≥1 up to m.

Base Case: To show P(1) is true. That is $f_1 \le f^*_1$.

Proof of Base Case: $f_1 \le f^*_1$ is true since f_1 has the earliest finish time of all the jobs by definition of the algorithm

Inductive case: Show that P(k) is true → P(k+1) is true for any 1≤k<m. Thus we need to show that $f_k \le f^*_k :\rightarrow f_{k+1} \le f^*_{k+1}$:

Proof of Inductive case:

(s_{k+1}, f_{k+1}) is the job with the earliest finish chosen from the set of jobs that start later than (s_k, f_k)

Claim: job (s^*_{k+1}, f^*_{k+1}) must start after job (s_k, f_k) finishes. Since

- $s^*_{k+1} \ge f^*_k$ since (s^*_{k+1}, f^*_{k+1}) is compatible with (s^*_k, f^*_k) and
- $f^*_k \ge f_k$ by the inductive hypothesis $f_k \le f^*_k$
- thus $s^*_{k+1} \ge f_k$

- Thus (s^*_{k+1}, f^*_{k+1}) is compatible with (s_k, f_k)
- Therefore (s_{k+1}, f_{k+1}) will finish no later than (s^*_{k+1}, f^*_{k+1}) (since it is chosen to be the compatible job with the earliest finish time that starts after f_k) or in other words $f_{k+1} \le f^*_{k+1}$

Thus by the Principle of Mathematical Induction: P(k) is true for k = 1..m

Final Step in the proof: Show that m=n. Suppose that m < n. But this is impossible since know by the above proposition that $f_m \le f^*_m$ Thus the $m+1^{st}$ job in **$\mathbf{\Sigma^*}$ must be compatible with the set Σ. and the greedy algorithm would choose it or some job that is compatible with Σ.** This contradicts that the number of jobs found by greedy is <.m. Thus **Σ has the same number of jobs as the optimal solution and is itself optimal.**