1. Main Techniques

1.1 Diagonalization

The most fundamental technique for proving functions non-computable. Key steps:

1. Assume the function is computable
2. Use it to construct a function that differs from every computable function
3. Reach a contradiction

Example:

```
To prove f(x) = {
    ϕ_x(x) + 1  if ϕ_x(x)↓
    0           otherwise
} is not computable:

1. Assume f is computable
2. Then ∃e such that f = ϕ_e
3. Consider f(e):
   - If ϕ_e(e)↓ then f(e) = ϕ_e(e) + 1 ≠ ϕ_e(e)
   - If ϕ_e(e)↑ then f(e) = 0 ≠ ϕ_e(e)
4. Contradiction: f(e) ≠ ϕ_e(e) = f(e)
```

1.2 Reduction from Known Non-Computable Functions

Common non-computable functions to reduce from:

- Halting function h(x,y) = "φ_x(y)↓"
- Diagonal function K(x) = "x ∈ W_x"
- Total function t(x) = "φ_x is total"

Example:

```
To prove g(x) = "ϕ_x is total" is not computable:
1. Assume g is computable
2. Define h(x) = {
       ϕ_x(x) + 1  if g(x) = 1
       0           otherwise
   }
3. Show h would be computable (contradiction)
```

2. Special Cases

2.1 Total Non-Computable Functions

To prove a total function is not computable:

1. Show it's total
2. Prove non-computability by diagonalization or reduction

Example:

```
f(n) = {
    χ_K(n)  if n ∈ K
    0       otherwise
}

1. Total because output always defined
2. Not computable as it would solve halting problem
```

2.2 Functions with Special Properties

Some functions might be non-computable due to special properties:

- Growth rate exceeding all computable functions
- Having undecidable properties in their domain/range
- Encoding solutions to undecidable problems

3. Common Proof Patterns

3.1 Contradiction via Halting Problem

```
1. Assume f is computable
2. Use f to decide membership in K
3. Reach contradiction since K is not decidable
```

3.2 Construction of Uncomputable Function

```
1. Start with a computable function g
2. Modify g to create f that differs from every ϕ_e
3. Show f cannot be computable
```

3.3 Rice's Theorem Application

```
1. Show function defines a non-trivial property
   of computable functions
2. Apply Rice's Theorem to prove non-computability
```

4. Common Mistakes to Avoid

1. Not proving totality when claiming a function is total
2. Confusing partial and total functions
3. Incorrect application of diagonalization
4. Assuming composition preserves non-computability

5. Exercise Strategy

1. Identify which technique might work:
    - Does it look like diagonalization would work?
    - Can you reduce from a known non-computable function?
    - Does it involve properties of computable functions?
2. Set up the proof structure:
    - Assume computability
    - Plan contradiction
    - Build helper functions if needed
3. Execute the proof:
    - Be precise about function definitions
    - Track where computability is used
    - Verify contradiction is reached
4. Verify:
    - Check all cases are covered
    - Verify all functions used are computable
    - Confirm contradiction is valid