1. Many-One Reducibility (≤_m)

1.1 Definition

For sets $A, B \subseteq \mathbb{N}$, $A \leq_m B$ if there exists a total computable function f such that:

```
∀x: x ∈ A ⟺ f(x) ∈ B
```

1.2 Properties

1. Transitivity: If $A \leq_m B$ and $B \leq_m C$ then $A \leq_m C$
2. Properties preserved:

```
- If B recursive ⟹ A recursive
- If B r.e. ⟹ A r.e.
- If A not recursive ⟹ B not recursive
- If A not r.e. ⟹ B not r.e.
```

1.3 Template for Reductions

```
To prove A ≤_m B:

1. Define g(x,y) function:
   g(x,y) = {
     [something] if x ∈ A
     [something else] otherwise
   }

2. Apply s-m-n theorem to get s where:
   ϕ_s(x)(y) = g(x,y)

3. Prove s is reduction function:
   x ∈ A ⟺ s(x) ∈ B
```

2. Common Reduction Patterns

2.1 From K (Halting Set)

```
To reduce K ≤_m A:
```

```
1. Define g(x,y) = {
     1 if x ∈ K
     ↑ otherwise
   }

2. Get s where ϕ_s(x)(y) = g(x,y)

3. Prove x ∈ K ⟺ s(x) ∈ A
```

2.2 From K̄ (Complement of Halting Set)

```
To reduce K̄ ≤_m A:

1. Define g(x,y) = {
     y if x ∉ K
     ↑ if x ∈ K
   }

2. Get s where ϕ_s(x)(y) = g(x,y)

3. Prove x ∈ K̄ ⟺ s(x) ∈ A
```

3. Problem Types and Solutions

3.1 Non-Recursiveness Proofs

```
To prove A not recursive:

1. Show K ≤_m A or K̄ ≤_m A
2. Since K/K̄ not recursive, A not recursive
```

3.2 Non-R.E. Proofs

```
To prove A not r.e.:

1. Show K̄ ≤_m A
2. Since K̄ not r.e., A not r.e.
```

3.3 Comparing Sets

```
To compare sets A and B:

1. Try to construct reduction A ≤_m B
2. If impossible, prove B ≤_m A
3. If neither possible, prove incomparability
```

4. Exercise Examples

4.1 Input Problem

```
Prove A_n = {x | ϕ_x(n)↓} not recursive:

1. Define g(x,y) = {
     1 if x ∈ K
     ↑ otherwise
   }

2. Get s where ϕ_s(x)(y) = g(x,y)

3. Show x ∈ K ⟺ s(x) ∈ A_n
```

4.2 Output Problem

```
Prove B_n = {x | n ∈ E_x} not recursive:

1. Define g(x,y) = {
     n if x ∈ K
     ↑ otherwise
   }

2. Get s where ϕ_s(x)(y) = g(x,y)

3. Show x ∈ K ⟺ s(x) ∈ B_n
```

5. Common Pitfalls

1. Function not total:

```
Reduction function must be total computable
```

2. Wrong direction:

```
Must prove both directions:
x ∈ A ⟹ f(x) ∈ B
f(x) ∈ B ⟹ x ∈ A
```

3. Not computable:

```
Ensure reduction function is computable
```

6. Strategy for Finding Reductions

1. Look for patterns:

```
- How to encode membership in A into B?
- What structure in B can represent A?
```

2. Common transformations:

```
- Use input as parameter
- Encode multiple values
- Use s-m-n theorem
```

3. Check requirements:

```
- Total function
- Computable
- Preserves membership
```