EXAM DAY COMPLETE GUIDE - Every Problem Type Explained

Your Exam: Thursday, December 5, 2025 at 2:00 PM
Goal: 49/70 points (70%) to pass
Format: 4 problems, 90 minutes, open notes

==

TABLE OF CONTENTS

==

==

PROBLEM TYPE 1: TM DESIGN (20 POINTS)

==

What You'll See

Example: "Design a TM that accepts A = {0, 00, 010}"

Variations:
- "Design TM for strings ending with pattern X"
- "Design TM that accepts exactly these strings"
- "Design TM that accepts strings containing pattern X"

STEP-BY-STEP WALKTHROUGH

MINUTE 0-1: Write Examples

DO THIS FIRST! Write 3-4 test cases:

✓ "0" → ACCEPT (in the set)
✓ "00" → ACCEPT (in the set)
✓ "010" → ACCEPT (in the set)
✗ "1" → REJECT (not in set)
✗ "000" → REJECT (not in set)
✗ "0100" → REJECT (not in set)

Why? Catches edge cases immediately!

MINUTE 1-2: Identify Problem Type

Ask yourself: "What kind of problem is this?"

TYPE A: Finite Specific Strings (like {0, 00, 010})
- Accepts EXACTLY these strings, nothing else
- States = one per string + intermediate steps
- Strategy: Build a "tree" of states that branch based on what you see

TYPE B: Pattern at End (like "ends with 010")
- Accepts ANY string ending with pattern
- States = remember last N symbols
- Strategy: States track "progress toward pattern"

TYPE C: Contains Pattern (like "contains 101")
- Once you see pattern, stay accepted
- Strategy: Similar to Type B but once pattern found, stay in accept zone

WALKTHROUGH EXAMPLE 1: Finite Strings A = {0, 00}

STEP 1: Plan the logic (1 minute)

If string is "0":
  Read 0 → see blank → ACCEPT

If string is "00":
  Read 0 → read 0 → see blank → ACCEPT

Anything else:
  REJECT

STEP 2: Design states (2 minutes)

q_0 = START (haven't read anything)
q_1 = SEEN-ONE-0 (read one 0)
q_2 = SEEN-TWO-0s (read two 0s)
qaccept = Accept
qreject = Reject

Key insight: States remember "how far into a valid string we are"

STEP 3: Work out transitions (5 minutes)

Work through ONE STATE AT A TIME:

From q_0 (START):
- See 0? → Could be "0" or start of "00" → go to q_1
- See 1? → Not in our language → qreject
- See blank? → Empty string, not valid → qreject

q_0:
  0 → 0, R, q_1
  1 → 1, R, qreject

  ⊔ → ⊔, R, qreject

From q_1 (SEEN-ONE-0):
- See 0? → Could be "00" → go to q_2
- See 1? → "01" not in language → qreject
- See blank? → String is "0" which IS valid → qaccept!

q_1:
  0 → 0, R, q_2
  1 → 1, R, qreject
  ⊔ → ⊔, R, qaccept ← Accepts "0"!

From q_2 (SEEN-TWO-0s):
- See 0? → "000" not in language → qreject
- See 1? → "001" not in language → qreject
- See blank? → String is "00" which IS valid → qaccept!

q_2:
  0 → 0, R, qreject
  1 → 1, R, qreject
  ⊔ → ⊔, R, qaccept ← Accepts "00"!

STEP 4: Draw diagram (3 minutes)

```
      0→0,R         0→0,R
  →  [q₀] ─────────→ [q₁] ─────────→ [q₂]
       |          |          |
       | 1 or ⊔     | ⊔          | ⊔
      ↓          ↓          ↓
    [qrej]     [qacc]◎     [qacc]◎
```

STEP 5: Test one example (2 minutes)

Pick "00" (hardest case):

```
[0][0][⊔]
 ↑
```
q_0 → see 0 → go to q_1, move R

```
[0][0][⊔]
    ↑
```
q_1 → see 0 → go to q_2, move R

```
[0][0][⊔]
       ↑
```
q_2 → see ⊔ → ACCEPT ✓

Total time: ~12 minutes

WALKTHROUGH EXAMPLE 2: Ends with Pattern "101"

STEP 1: Examples (1 minute)

✓ "101" → ACCEPT
✓ "0101" → ACCEPT
✓ "11101" → ACCEPT
✗ "1010" → REJECT (ends with 010)
✗ "10" → REJECT (too short)

STEP 2: Key insight (30 seconds)

"I need to REMEMBER the last 3 symbols I've seen!"

STEP 3: Design states (2 minutes)

States track "what's at the end so far":

q_0 = No pattern progress (or broke pattern)
q_1 = Last symbol was "1"
q_2 = Last 2 symbols were "10"
q_3 = Last 3 symbols were "101" ← WINNING STATE!

STEP 4: Figure out transitions (5 minutes)

From q_0 (no pattern):
- See 1? → Start of "101" → q_1
- See 0? → Can't start "101" with 0 → stay q_0
- See blank? → Didn't end with "101" → reject

q_0:
  1 → 1, R, q_1 (starting pattern)
  0 → 0, R, q_0 (no progress)
  ⊔ → ⊔, R, qreject

From q_1 (last was "1"):
- See 0? → Now have "10" → q_2
- See 1? → Still just "1" → stay q_1
- See blank? → Only had "1" → reject

q_1:
  0 → 0, R, q_2 (building pattern)
  1 → 1, R, q_1 (restart with new 1)
  ⊔ → ⊔, R, qreject

From q_2 (last 2 were "10"):
- See 1? → Now have "101"! → q_3 ✓
- See 0? → Now "100", broke pattern → q_0
- See blank? → Only "10" → reject

q_2:

1 → 1, R, q_3 (PATTERN COMPLETE!)
0 → 0, R, q_0 (broke pattern)
⊔ → ⊔, R, qreject

From q_3 (have "101" at end!):
- See 1? → New "1" could start new pattern → q_1
- See 0? → Broke pattern → q_0
- See blank? → STRING ENDS WITH "101" → ACCEPT!

q_3:
1 → 1, R, q_1 (continue reading)
0 → 0, R, q_0 (broke pattern)
⊔ → ⊔, R, qaccept ← THE WINNER!

STEP 5: Draw diagram (3 minutes)

```
   1(loop)
    ┌───────┐
   ↓   |
→ [q₀]───┴─→ [q₁] ───→ [q₂] ───→ [q₃]
   | 0     1→1,R   0→0,R   1→1,R
   └─────┘                  | ⊔
                     ↓
                  [qacc]◎
```

STEP 6: Test example (2 minutes)

Test "0101" (tricky case):

```
[0][1][0][1][⊔]
 ↑
```
q_0 → see 0 → stay q_0

```
[0][1][0][1][⊔]
    ↑
```
q_0 → see 1 → go q_1

```
[0][1][0][1][⊔]
       ↑
```
q_1 → see 0 → go q_2 (have "10")

```
[0][1][0][1][⊔]
          ↑
```
q_2 → see 1 → go q_3 (have "101"!)

```
[0][1][0][1][⊔]
             ↑
```
q_3 → see ⊔ → ACCEPT! ✓

Total time: ~12 minutes

==
EDGE CASES & PIVOT STRATEGIES
==

Edge Case 1: Empty String

Problem: "What if input is empty (just blank)?"

Pivot:
- If empty string IS in the language → accept from q_0 on blank
- If empty string NOT in language → reject from q_0 on blank

Example: A = {ε, 0} includes empty string

```
q₀:
  0 → 0, R, q₁
  ⊔ → ⊔, R, qaccept ← Accepts empty!
```

Edge Case 2: Single Symbol

Problem: "What if string is just one symbol?"

Test it! Write "0" and "1" as test cases first thing.

Example: If A = {0, 00}, then "0" should accept but "1" should reject.

Edge Case 3: Pattern Can Overlap

Problem: What if string is "01010"? Do we see "010" twice?

Answer: For "ends with" problems, you only care about THE END!

Strategy: When you complete a pattern (reach q_3), keep reading! You might break the pattern later.

Example: "01010"
- Read "010" → reach q_3 (winning state)
- Read another "1" → break pattern, go back to q_1
- Read "0" → go to q_2
- End → in q_2, not q_3 → REJECT ✗

Edge Case 4: Very Long Strings

Don't worry! TM doesn't care about length. Your states work the same for "101" as for "11111111101".

PIVOT STRATEGIES

If you're stuck on transitions:

1. Pick ONE state
2. Ask for EACH possible symbol: "What should happen if I see 0? 1? blank?"
3. Move to next state

If diagram is messy:

Just write the transitions as a list! Diagram is optional (but helpful for YOU to check).

If running out of time:

Priority:
1. Write all states (1 point)
2. Write all transitions (7-8 points)
3. Skip diagram if needed (2 points)
4. Skip testing (0 points, but helps YOU catch errors)

==
PROBLEM TYPE 2: BINARY MODIFICATION (15 POINTS)
==

What You'll See

Example: "Design TM that takes binary number w and outputs w-1 (decrement)"

Variations:
- Increment (w+1)
- Double (2w)
- Check if even/odd
- Reverse the string
- Copy the string

STEP-BY-STEP WALKTHROUGH

EXAMPLE: Binary Increment (w + 1)

MINUTE 0-1: Write Examples

"1" + 1 = "10" (1→2) ✓
"10" + 1 = "11" (2→3) ✓
"11" + 1 = "100" (3→4) ← Carrying!

"111" + 1 = "1000" (7→8) ← Lots of carrying!
"101" + 1 = "110" (5→6) ✓

Notice the pattern: Carrying happens with consecutive 1s!

MINUTE 1-3: Figure Out Algorithm (Scratch Paper)

Think out loud:

"Adding 1 in binary... I start at the RIGHT (like regular addition)"

"If rightmost is 0 → change to 1, done! Easy!"

"If rightmost is 1 → change to 0, carry left"

"Keep carrying: change 1s to 0s until I find a 0"

"If all 1s, need to add new 1 at start!"

Write it out:

Algorithm:
1. Scan right to find end
2. Move left to rightmost digit
3. If 0 → change to 1, ACCEPT
4. If 1 → change to 0, carry left
5. Keep carrying: 1→0, move left
6. When find 0 → change to 1, ACCEPT
7. If reach start → write 1 (overflow), ACCEPT

MINUTE 3-8: Write Part (a) - English Algorithm (5 POINTS!)

THIS IS FREE POINTS - DON'T SKIP!

What you write on exam:

PART (a): English algorithm for binary increment

ALGORITHM:

1. Scan right across the tape until we reach the blank symbol (⊔), marking the end of the number.

2. Move left to position ourselves on the rightmost digit.

3. Check the rightmost digit:
   - If it is 0: Change it to 1, move left, and ACCEPT (increment complete, no carry)
   - If it is 1: Change it to 0 (need to carry), move left, and proceed to step 4

4. Continue carrying (moving left):
   - For each 1 encountered: change it to 0, continue moving left (still carrying)
   - When we find a 0: change it to 1, move left, and ACCEPT (carry absorbed)
   - If we reach the blank (␣) at the start: we've carried off the left end, so write 1 and ACCEPT (overflow - all 1s became all 0s, add leading 1)

5. Edge case: If input is empty, write 1 and ACCEPT (0 + 1 = 1)

EXAMPLES:
- Input "10" (2): rightmost is 0 → change to 1 → output "11" (3) ✓
- Input "11" (3): rightmost is 1 → change to 0, carry left, find 1, change to 0, carry left, hit start, write 1 → output "100" (4) ✓
- Input "1011" (11): rightmost is 1 → change to 0, carry left, find 1, change to 0, carry left, find 0, change to 1 → output "1100" (12) ✓

✅ You just got 5 points in 5 minutes!

MINUTE 8-10: Identify States

Every binary modification has 3 PHASES:

PHASE 1: POSITIONING (find where to work)
  → qstart

PHASE 2: MODIFICATION (do the work)
  → qinc (at rightmost, decide what to do)
  → qcarry (keep carrying left)

PHASE 3: DECISION (accept/reject)
  → qaccept

Mental note: 4 states total for increment

MINUTE 10-15: Design Transitions (Part b - 10 points)

Work through each state:

qstart (PHASE 1 - scan right):

"Keep moving right until I hit blank..."

```
qstart:
  0 → 0, R, qstart (keep scanning)
  1 → 1, R, qstart (keep scanning)
  ␣ → ␣, L, qinc (found end! move back to last digit)
```

qinc (PHASE 2 - at rightmost digit):

"Am I on 0 or 1?"

qinc:
  0 → 1, L, qaccept (easy case! 0→1, done)
  1 → 0, L, qcarry (need to carry! 1→0, keep going)
  ␣ → 1, L, qaccept (empty string → write 1)

qcarry (PHASE 2 - carrying left):

"Keep changing 1s to 0s until I find a 0..."

qcarry:
  1 → 0, L, qcarry (still carrying, keep going)
  0 → 1, L, qaccept (found 0! change to 1, done)
  ␣ → 1, L, qaccept (overflow! write new 1 at start)

qaccept: HALT ◎

MINUTE 15-18: Write It Formally on Exam

States: Q = {qstart, qinc, qcarry, qaccept}

Transition function δ:

qstart:
- δ(qstart, 0) = (qstart, 0, R)
- δ(qstart, 1) = (qstart, 1, R)
- δ(qstart, ␣) = (qinc, ␣, L)

qinc:
- δ(qinc, 0) = (qaccept, 1, L)
- δ(qinc, 1) = (qcarry, 0, L)
- δ(qinc, ␣) = (qaccept, 1, L)

qcarry:
- δ(qcarry, 1) = (qcarry, 0, L)
- δ(qcarry, 0) = (qaccept, 1, L)
- δ(qcarry, ␣) = (qaccept, 1, L)

Diagram:

```
PHASE 1: SCAN
   0,1 loop
    ┌──────────┐
   ↓     |
→ [qstart]┘
    | ␣→␣,L
   ↓
PHASE 2: INCREMENT
  [qinc]
```

```
   |    |
   |    └→ 1→0,L → [qcarry]
   |                |   |
   |                |   └→ 1→0,L (loop)
   |                |
   |                └→ 0→1,L or ⊔→1,L
   |                      ↓
   └→ 0→1,L → [qaccept]◎
```

MINUTE 18-20: Test Example

Pick "11" (the carry case):

```
Initial: [1][1][⊔]
           ↑
         qstart
```

After scanning:

```
[1][1][⊔]
      ↑
qstart → see ⊔ → move L, go qinc

[1][1][⊔]
   ↑
qinc → see 1 → write 0, move L, go qcarry

[1][0][⊔]
 ↑
qcarry → see 1 → write 0, move L, stay qcarry

[0][0][⊔]
↑
qcarry → see ⊔ → write 1, move L, go qaccept

[1][0][0][⊔]
↑
qaccept ✓
```

Result: "100" = 4 in decimal ✓ CORRECT!

Total time: ~20 minutes for 15 points!

==
EDGE CASES & PIVOT STRATEGIES
==

Edge Case 1: Empty Input

Problem: What if input is blank (empty)?

For increment: Empty → should become "1"

Solution:

```
qinc:
  ⊔ → 1, L, qaccept
```

For decrement: Empty → can't decrement nothing → REJECT

Edge Case 2: Input is "0"

For increment: "0" + 1 = "1" ✓ (works fine)

For decrement: "0" - 1 = ??? (can't go negative!) → REJECT

Solution:

```
qborrow:
  ⊔ → ⊔, R, qreject (reached start while borrowing = was "0")
```

Edge Case 3: All 1s (Overflow)

Problem: "111" + 1 = "1000" (need to add digit!)

Solution: When qcarry hits blank (⊔), write new 1:

```
qcarry:
  ⊔ → 1, L, qaccept (write new leading 1)
```

Edge Case 4: All 0s

For decrement: "1000" - 1 = "111" (borrow through all 0s)

Trace carefully! Make sure qborrow keeps moving left through 0s without changing them.

PIVOT STRATEGIES

If problem is DIFFERENT operation:

Decrement (w - 1):
- Start at right
- If 1 → change to 0, done (easy)
- If 0 → change to 1, borrow left

- Borrow: keep moving left until find 1, change to 0
- Edge: if hit start while borrowing → reject (was "0")

Double (2w):
- Scan left to find start
- Mark first symbol, copy it to end
- Repeat for all symbols
- More complex! 6-8 states

Check if Even:
- Scan right to last digit
- If last is 0 → ACCEPT (even)
- If last is 1 → REJECT (odd)
- Super simple! 3 states

Reverse:
- Mark symbols from left, copy to right in reverse
- Very complex! Save for last if running out of time

If stuck on algorithm:

Think like you're doing it by hand!

1. Write out "111" + 1 = ? on paper
2. Show your work with carries
3. That's your algorithm!

If running out of time:

PRIORITY:
1. Part (a) English algorithm (5 points - EASY!)
2. States list (1 point)
3. Key transitions (partial credit - 3-4 points)
4. Skip diagram if needed
5. Skip testing

==
PROBLEM TYPE 3: COUNTABILITY (10 POINTS)
==

What You'll See

Example: "Is Q = {(i,j) | i,j ∈ ℕ} countable?"

Translation: "Can you list ALL pairs of natural numbers?"

Variations:
- Pairs: ℕ×ℕ, ℤ×ℕ, etc.
- Triples: ℕ×ℕ×ℕ
- Sets of strings: {0,1}* (all binary strings)

- Subsets: P(ℕ) (power set)

QUICK DECISION TREE (30 seconds)

```
Is it...
├── Finite set? → COUNTABLE ✓
├── ℕ, ℤ, ℚ? → COUNTABLE ✓
├── Pairs/triples of countable sets? → COUNTABLE ✓
├── Finite strings over finite alphabet? → COUNTABLE ✓
├── ℝ, intervals like [0,1]? → NOT COUNTABLE ✗
└── P(ℕ) (power set of ℕ)? → NOT COUNTABLE ✗
```

STEP-BY-STEP WALKTHROUGH

EXAMPLE: Is ℕ×ℕ countable?

MINUTE 0-1: Understand the Question

Translation: "Can I list ALL pairs (i,j) where i,j are natural numbers?"

Examples: (0,0), (0,1), (1,0), (1,1), (2,3), (100,200), ...

Seems like a LOT! But let's try...

MINUTE 1-2: Try to Enumerate

Bad attempt:

(0,0), (0,1), (0,2), (0,3), ...

Problem: This NEVER gets to (1,0)! ✗

Good attempt - Enumerate by SUM:

Sum = 0: (0,0) → 1 pair
Sum = 1: (0,1), (1,0) → 2 pairs
Sum = 2: (0,2), (1,1), (2,0) → 3 pairs
Sum = 3: (0,3), (1,2), (2,1), (3,0) → 4 pairs
Sum = 4: (0,4), (1,3), (2,2), (3,1), (4,0) → 5 pairs
...

This works! Every pair (i,j) appears at sum level i+j!

MINUTE 2-7: Write Your Answer

QUESTION: Is Q = {(i,j) | i,j ∈ ℕ} countable?

ANSWER: YES, Q is countable. ✅

JUSTIFICATION:

A set is countable if we can enumerate all its elements systematically
(establish a bijection with ℕ).

For the set Q, we can enumerate all pairs by their sum (i + j):

Sum = 0: (0,0)
Sum = 1: (0,1), (1,0)
Sum = 2: (0,2), (1,1), (2,0)
Sum = 3: (0,3), (1,2), (2,1), (3,0)
Sum = 4: (0,4), (1,3), (2,2), (3,1), (4,0)
...and so on

This enumeration is systematic and covers every possible pair exactly once:
- For any pair (i,j), it appears at sum level i+j
- Each sum level has finitely many pairs
- We can list them in order: sum 0, then sum 1, then sum 2, etc.

Therefore, Q is countable. ✓

ALTERNATIVE PROOF: We can define a bijection f: ℕ×ℕ → ℕ using
f(i,j) = 2^i × 3^j, which maps each pair to a unique natural number.

Total time: ~7 minutes for 10 points!

==
EDGE CASES & PIVOT STRATEGIES
==

Edge Case 1: It's NOT Countable

Example: "Is ℝ countable?"

ANSWER: NO ✗

JUSTIFICATION:

By Cantor's diagonalization argument, ℝ is uncountable.

Proof sketch:
1. Assume ℝ is countable (for contradiction)
2. List all real numbers: r_1, r_2, r_3, ...

3. Construct new number d that differs from r_n in nth decimal place
4. d is real but NOT in our list! Contradiction.
5. Therefore, ℝ is uncountable. ✗

Other uncountable sets:
- Any interval: [0,1], (0,1), [a,b]
- P(ℕ) (power set of natural numbers)
- Set of all infinite binary sequences

Edge Case 2: Integers (ℤ)

Question: "Is ℤ countable?"

YES! ✓

Enumeration:

0, 1, -1, 2, -2, 3, -3, 4, -4, ...

Pattern: 0, then alternate positive/negative

Formula:
- $f(0) = 0$
- $f(2n) = n$ (for $n > 0$)
- $f(2n+1) = -n$ (for $n > 0$)

Edge Case 3: Rationals (ℚ)

Question: "Is ℚ countable?"

YES! ✓

Strategy: Enumerate by height p+q for fraction p/q

Height 1: none
Height 2: 1/1
Height 3: 1/2, 2/1
Height 4: 1/3, 2/2 (skip dup), 3/1
Height 5: 1/4, 2/3, 3/2, 4/1
...

Skip duplicates (like 2/2 = 1/1)

Edge Case 4: Triples (ℕ×ℕ×ℕ)

Question: "Is ℕ×ℕ×ℕ countable?"

YES! ✓

Strategy 1: Enumerate by sum i+j+k

Strategy 2: Use pairing twice
- First pair (i,j) → natural number n
- Then pair n with k → natural number m

Key principle: Countable × Countable = Countable!

Edge Case 5: Strings ({0,1}*)

Question: "Is the set of all binary strings countable?"

YES! ✓

Enumeration by length:

Length 0: ε
Length 1: 0, 1
Length 2: 00, 01, 10, 11
Length 3: 000, 001, 010, 011, 100, 101, 110, 111
...

Each length has finitely many strings!

PIVOT STRATEGIES

If you're not sure:

Ask yourself:
1. "Can I group elements somehow?" (by sum, by length, by height)
2. "Is each group finite?"
3. "Can I order the groups?"

If YES to all 3 → COUNTABLE!

Quick templates:

For COUNTABLE:

ANSWER: YES, the set is countable.

JUSTIFICATION:
We can enumerate all elements systematically by [YOUR GROUPING]:

[Show first few groups]

This enumeration covers all elements because:
- [Explain why every element appears]
- [Explain why each group is finite]
- [Explain why we can order groups]

Therefore, the set is countable.

For UNCOUNTABLE:

ANSWER: NO, the set is NOT countable.

JUSTIFICATION:
[If it's ℝ or interval]: By Cantor's diagonalization, this set is uncountable.

[If it's power set]: By Cantor's theorem, the power set of any set is strictly larger than the set itself, so P(ℕ) is uncountable.

Therefore, the set is not countable.

If running out of time:

Just write:
1. Answer (YES or NO) - 2 points
2. Enumeration method (by sum, by length, etc.) - 4 points
3. Brief explanation - 4 points

Skip: Formal bijection proof (save 3 minutes!)

==

PROBLEM TYPE 4: DECIDABILITY (20 POINTS)

==

What You'll See

Format: "Is [PROBLEM] decidable? Is it Turing-recognizable?"

Examples:
- ALL_DFA = Does this DFA accept everything?
- EQ_CFG = Do these CFGs generate same language?
- HALT_TM = Does this TM halt on this input?

QUICK LOOKUP TABLE (MEMORIZE THIS!)

Problem Type	Decidable?	Recognizable?	Why?
DFA problems	YES ✅	YES ✅	Can check in finite time
E_DFA	YES ✅	YES ✅	Graph reachability
EQ_DFA	YES ✅	YES ✅	Minimize & compare
ALL_DFA	YES ✅	YES ✅	Check complement
CFG problems	Mixed	Mixed	Depends on problem
A_CFG	YES ✅	YES ✅	Parse finite derivations

E_CFG	YES ✅	YES ✅	Check if generates anything
EQ_CFG	NO ❌	NO ❌	Can't compare infinite langs
TM problems	NO ❌	Mixed	Can't predict behavior
A_TM	NO ❌	YES ✅	Can recognize, can't decide
HALT_TM	NO ❌	NO ❌	Classic undecidable
E_TM	NO ❌	NO ❌	Can't tell if accepts nothing

STEP-BY-STEP WALKTHROUGH

EXAMPLE 1: ALL_DFA

Problem: ALL_DFA = {⟨A⟩ | A is a DFA and L(A) = Σ*}

Translation: "Does this DFA accept EVERY possible string?"

MINUTE 0-1: Identify Type

Keywords: "DFA"

Quick check: DFA problems → almost always DECIDABLE!

MINUTE 1-3: Part (a) - Is it decidable?

Think: "How would I check if DFA accepts everything?"

Bad idea: Test every string → infinite strings! ✗

Good idea: Use complement!
- If L(A) = Σ*, then complement is empty
- E_DFA (emptiness) is decidable
- So check if complement is empty!

Answer: YES, decidable ✓

MINUTE 3-8: Write Part (a) Answer

QUESTION: Is ALL_DFA decidable?

ANSWER: YES, ALL_DFA is decidable. ✅

JUSTIFICATION:

To decide whether a DFA A accepts all strings (L(A) = Σ*), we can construct a Turing Machine M that:

1. Takes input ⟨A⟩ (encoding of DFA A)

2. Constructs the complement DFA A' where:
   - $L(A') = \Sigma^* - L(A)$
   - (Easy: flip accept and non-accept states)

3. Checks if $L(A') = \emptyset$ using E_DFA decision procedure:
   - Run graph search from start state
   - Check if ANY accept state is reachable
   - If no accept state reachable → L(A') is empty
   - If accept state reachable → L(A') is not empty

4. Decide:
   - If $L(A') = \emptyset$, then $L(A) = \Sigma^*$, so ACCEPT
   - If $L(A') \neq \emptyset$, then $L(A) \neq \Sigma^*$, so REJECT

Since E_DFA is decidable (graph reachability always terminates) and
DFAs are closed under complement, this procedure always halts with
the correct answer.

Therefore, ALL_DFA is decidable. ✓

MINUTE 8-10: Part (b) - Is it recognizable?

Key theorem: If decidable → recognizable (always!)

Reasoning: If TM always halts with answer, then it certainly halts
and accepts when answer is YES.

MINUTE 10-12: Write Part (b) Answer

QUESTION: Is ALL_DFA Turing-recognizable?

ANSWER: YES, ALL_DFA is Turing-recognizable. ✅

JUSTIFICATION:

Since ALL_DFA is decidable (as shown in part a), it is also
Turing-recognizable.

KEY THEOREM: Every decidable language is Turing-recognizable.

REASONING:
- Decidable means: TM always halts (accept or reject)
- Recognizable means: TM halts and accepts when $w \in L$ (may loop when $w \notin L$)
- If we have a TM that ALWAYS halts, then it certainly halts on YES cases
- Therefore: decidable → recognizable (always true)

Since ALL_DFA is decidable, it must be Turing-recognizable. ✓

Total time: ~12 minutes for 20 points!

EXAMPLE 2: EQ_CFG (THE TRICKY ONE!)

Problem: EQ_CFG = {⟨G,H⟩ | G,H are CFGs and L(G) = L(H)}

Translation: "Do these two CFGs generate the same language?"

MINUTE 0-1: Identify Type

Keywords: "CFG" + "equality"

Quick check: CFG equality → NOT DECIDABLE!

MINUTE 1-3: Part (a) - Is it decidable?

Think: "How would I check if two CFGs are equal?"

Bad idea: Test every string → infinite! ✗

Reality: CFGs can't be minimized, can't be fully compared.

Answer: NO, not decidable ✗

MINUTE 3-8: Write Part (a) Answer

QUESTION: Is EQ_CFG decidable?

ANSWER: NO, EQ_CFG is NOT decidable. ❌

JUSTIFICATION:

To decide whether two CFGs generate the same language, we would need to compare potentially infinite languages.

WHY IT'S NOT DECIDABLE:

1. CFGs can generate infinite languages
2. We cannot exhaustively test all strings (infinite!)
3. CFGs are NOT closed under intersection or complement
   (unlike DFAs, which are)
4. We cannot minimize CFGs to a canonical form
   (unlike DFAs, which can be minimized)

Since there is no algorithmic way to completely compare two potentially infinite languages generated by CFGs, no Turing Machine can always decide this problem.

Therefore, EQ_CFG is NOT decidable. ✗

MINUTE 8-11: Part (b) - Is it recognizable?

THIS IS TRICKY! Most things are at least recognizable, but EQ_CFG is NOT!

Key insight: The COMPLEMENT is recognizable!

Think about complement: "CFGs are NOT equal"
- To show L(G) ≠ L(H), find ONE string that differs
- Enumerate all strings: ε, 0, 1, 00, 01, 10, 11, ...
- Test each in both languages
- If find difference → accept!
- This IS recognizable! ✓

Key theorem: If both L and complement are recognizable → L is decidable

Since EQ_CFG NOT decidable but complement IS recognizable → EQ_CFG NOT recognizable!

MINUTE 11-15: Write Part (b) Answer

QUESTION: Is EQ_CFG Turing-recognizable?

ANSWER: NO, EQ_CFG is NOT Turing-recognizable. ❌

JUSTIFICATION:

This is a special case! Let me explain using the complement:

THE COMPLEMENT IS RECOGNIZABLE:

EQ̄_CFG = {⟨G,H⟩ | L(G) ≠ L(H) or invalid encoding}

To recognize the complement (inequality):
- Enumerate all strings in Σ*: ε, 0, 1, 00, 01, ...
- For each string w, test membership in both languages:
  * Check if w ∈ L(G) using A_CFG (decidable)
  * Check if w ∈ L(H) using A_CFG (decidable)
- If we find w ∈ L(G) but w ∉ L(H), or vice versa: ACCEPT (languages different!)
- This procedure WILL find a difference if one exists
- Therefore, EQ̄_CFG IS recognizable ✓

KEY THEOREM: If both L and L̄ (complement) are Turing-recognizable, then L is decidable.

APPLYING THE THEOREM:
- We proved EQ̄_CFG IS recognizable
- We know EQ_CFG is NOT decidable (from part a)
- By the theorem: if EQ_CFG were also recognizable, it would be decidable

- But it's NOT decidable!
- Therefore, EQ_CFG must NOT be recognizable

CONCLUSION: EQ_CFG is NOT Turing-recognizable. ✗

This is one of the few languages that is not even recognizable!

Total time: ~15 minutes for 20 points!

==
EDGE CASES & PIVOT STRATEGIES
==

Edge Case 1: DFA vs CFG vs TM

RULE OF THUMB:
- DFA problems → DECIDABLE ✓
- CFG problems → MIXED (depends on problem)
- TM problems → NOT DECIDABLE ✗

Edge Case 2: Complement Questions

If problem asks about complement:

Example: "Is Ē_DFA (non-emptiness) decidable?"

Answer: YES (if E_DFA decidable, so is complement)

Key: Decidable languages are closed under complement!

Edge Case 3: Both Parts Same Answer

Common pattern:
- If decidable → also recognizable (part b is YES)
- If not decidable BUT recognizable → need to think about complement

Example: A_TM
- Part (a): NO, not decidable
- Part (b): YES, recognizable (just run the TM!)

Edge Case 4: Neither Decidable NOR Recognizable

Only a few:
- EQ_CFG (CFG equality)
- E_TM (TM emptiness)
- HALT̄_TM (non-halting)

Strategy: Show complement IS recognizable, use theorem.

PIVOT STRATEGIES

If you forget whether it's decidable:

Use this flowchart:

```
Is it DFA problem?
├── YES → Decidable ✓
└── NO → Is it CFG problem?
    ├── YES → Is it A_CFG or E_CFG?
    |   ├── YES → Decidable ✓
    |   └── NO (equality) → NOT decidable ✗
    └── NO (TM problem) → NOT decidable ✗
```

If you can't explain WHY:

For DECIDABLE:
1. Describe high-level algorithm
2. Say "this always terminates"
3. Say "so it's decidable"

For NOT DECIDABLE:
1. Say "we'd need to check infinite cases"
2. OR say "this would solve halting problem"
3. Say "so it's not decidable"

Part (b) quick template:

If part (a) was YES (decidable):

ANSWER: YES, it's recognizable.

Since [PROBLEM] is decidable, it is also recognizable.
Every decidable language is Turing-recognizable because if a TM always halts, it certainly halts on accept cases.

If part (a) was NO (not decidable):

Need to think more carefully...
- Can we recognize YES cases? (run algorithm, accept if find proof)
- Is complement recognizable?
- Use theorem if both recognizable → decidable

If running out of time:

PRIORITY:
1. Part (a) answer + brief reason (10 points)
2. Part (b) answer (5 points)

3. Skip detailed justification if needed

Quick answers:
- DFA → YES, YES
- CFG membership → YES, YES
- CFG equality → NO, NO
- TM halting → NO, NO
- TM membership → NO, YES

==
FINAL EXAM STRATEGY
==

Time Allocation (90 minutes)

Problem 1 (TM Design):	15 minutes
Problem 2 (Binary Mod):	20 minutes
Problem 3 (Countability):	10 minutes
Problem 4a (Decidable):	12 minutes
Problem 4b (Decidable):	13 minutes
Review:	10 minutes
Buffer:	10 minutes

Order to Tackle Problems

RECOMMENDED ORDER:

1. Problem 3 (Countability) - 10 min - EASIEST, builds confidence
2. Problem 4 (Decidability) - 25 min - Use lookup table
3. Problem 2 (Binary Mod) - 20 min - Write English algorithm first!
4. Problem 1 (TM Design) - 15 min - Test examples!
5. Review - 10 min - Check for errors

What to Write First

On scratch paper:
1. Time budget (write end time for each problem)
2. Quick lookup table for decidability
3. "TEMPLATES: Finite=DFA, Pattern=Track, Binary=3Phase"

For each problem:
1. Write examples FIRST (catches edge cases!)
2. Plan approach (30 seconds)
3. Start writing formal answer

How to Get 49/70 Points (70%)

Target:
- Problem 1: 14/20 (miss diagram, still good!)
- Problem 2: 13/15 (English + states + most transitions)
- Problem 3: 10/10 (FULL POINTS - easiest!)
- Problem 4: 12/20 (both parts partial credit)

Total: 49 points = PASS! ✓

Emergency Strategies

If Running Out of Time

Problem 1 (TM):
- Write states (1 pt)
- Write key transitions (5-6 pts)
- Skip diagram

Problem 2 (Binary):
- WRITE ENGLISH ALGORITHM (5 pts - FREE!)
- Write states (1 pt)
- Write some transitions (3-4 pts)
- Skip testing

Problem 3:
- Don't skip! 10 minutes for 10 points!

Problem 4:
- Write answers (YES/NO) (4 pts)
- Write brief reason (4-6 pts)
- Skip detailed justification

If Stuck on a Problem

DON'T PANIC!

1. Skip to next problem (come back later)
2. Write what you know (partial credit!)
3. Draw examples (often shows the answer)
4. Use templates from this guide

If You Don't Know the Answer

Write SOMETHING!

For TM design:
- Write states (even if transitions wrong)
- Draw diagram (even if incomplete)
- Test example (shows your thinking)

For binary modification:
- Write English algorithm (describes what you're trying to do)
- List states (shows structure)

For countability:
- Write "YES" or "NO" (50% chance!)
- Show some enumeration attempt
- Explain your reasoning

For decidability:
- Use lookup table
- Write high-level idea
- State relevant theorems

Partial credit is real! 5-7 points per problem adds up!

The Night Before (Tonight!)

DON'T:
- ❌ Stay up all night
- ❌ Try to memorize everything
- ❌ Stress about problems you can't solve

DO:
- ✅ Review this guide (1-2 hours)
- ✅ Review your cheat sheet
- ✅ Review the worked examples
- ✅ Get 7-8 hours sleep
- ✅ Eat good breakfast tomorrow

Exam Day Checklist

Bring:
- ✅ Pencils (2-3)
- ✅ Eraser
- ✅ Your printed materials (cheat sheet, walkthroughs)
- ✅ Water bottle
- ✅ Student ID
- ✅ Confidence! 💪

10 minutes before exam:
- Breathe deeply
- Review template list
- Tell yourself: "I've got this!"

==

YOU'RE READY!
==

You have:
- ✅ Complete walkthrough of all 4 problem types
- ✅ Edge cases and pivot strategies
- ✅ Time management plan
- ✅ Emergency strategies
- ✅ Confidence boosters

Remember:
- Write examples first
- Use the templates
- Get partial credit
- 49/70 = PASS!

Tomorrow at 2:00 PM, you're going to walk in and crush this exam!

Good luck! 🚀