**Homework 2 – Report**

**University:** Shiraz University  **Course:** Artificial intelligence – Spring 2025 **Instructor:** Prof. Zohreh Azimifar **Student:** Salar Rahnama – Amirreza Baghban  **Student ID:** 40131850 - 40131840 **Assignment Title:** Informed search, CSP, Adversarial Search  **Due Date:** God and TAs know

**Q1:**

**a)**

**Problem Setup**

* **Classes and Times**:
  1. *C1* Class 1: Computer Fundamentals (8:00 AM–9:00 AM)
  2. *C2* Class 2: Artificial Intelligence (8:30 AM–9:30 AM)
  3. *C3* Class 3: Natural Language Processing (9:00 AM–10:00 AM)
  4. *C4* Class 4: Machine Vision (9:00 AM–10:00 AM)
  5. *C5* Class 5: Machine Learning (9:30 AM–10:30 AM)
* **Professors and Capabilities**:
  1. Professor A: Can teach Class 3, Class 4
  2. Professor B: Can teach Class 2, Class 3, Class 4, Class 5
  3. Professor C: Can teach all (Class 1, Class 2, Class 3, Class 4, Class 5)
* **Days**: Saturday, Sunday, Monday (we need to schedule all five classes across these days).
* **Constraints**:

The constraints ensure that the assignments are valid, focusing on professor availability and time conflicts. There are two types of constraints:

* **Unary Constraints (Implicit in Domains)**:
  1. Each class must be assigned a professor who is qualified to teach it. This is already enforced by the domains defined above (e.g., C1 C\_1 C1​ can only be assigned to Professor C).
* **Binary Constraints (No Overlap)**:
  1. No professor can teach two classes that overlap in time on the same day.

**Step 1: Define Domains**

* Class 1 (8:00–9:00 AM): Only Professor C can teach.  
   *D(C1)*: {(C, Sat), (C, Sun), (C, Mon)}.
* Class 2 (8:30–9:30 AM): Professors B, C.  
   *D(C2)*: {(B, Sat), (B, Sun), (B, Mon), (C, Sat), (C, Sun), (C, Mon)}.
* Class 3 (9:00–10:00 AM): Professors A, B, C.  
   *D(C3)*: {(A, Sat), (A, Sun), (A, Mon), (B, Sat), (B, Sun), (B, Mon), (C, Sat), (C, Sun), (C, Mon)}.
* Class 4 (9:00–10:00 AM): Professors A, B, C.  
   *D(C4)*: {(A, Sat), (A, Sun), (A, Mon), (B, Sat), (B, Sun), (B, Mon), (C, Sat), (C, Sun), (C, Mon)}. Same domain as Class 3.
* Class 5 (9:30–10:30 AM): Professors B, C.  
   *D(C5)*: {(B, Sat), (B, Sun), (B, Mon), (C, Sat), (C, Sun), (C, Mon)}.

**Step 2: Identify Overlapping Classes**

Classes overlap if their time intervals intersect on the same day. Let’s check the time ranges:

* Class 1 (8:00–9:00) overlaps with Class 2 (8:30–9:30) because 8:30–9:00 intersects.
* Class 2 (8:30–9:30) overlaps with Class 3 (9:00–10:00) and Class 4 (9:00–10:00) because 9:00–9:30 intersects.
* Class 3 (9:00–10:00) overlaps with Class 4 (9:00–10:00) (same time) and Class 5 (9:30–10:30) because 9:30–10:00 intersects.
* Class 4 (9:00–10:00) overlaps with Class 5 (9:30–10:30) because 9:30–10:00 intersects.
* Class 5 (9:30–10:30) overlaps with Class 3 and Class 4.

Overlap pairs (same day constraint):

* Class 1 and Class 2
* Class 2 and Class 3
* Class 2 and Class 4
* Class 3 and Class 4
* Class 3 and Class 5
* Class 4 and Class 5

If any of these pairs are scheduled on the same day, they cannot be assigned to the same professor.

Binary Constraints: For each pair of classes that overlap, if they are scheduled on the same day, they must be assigned different professors. Formally, for each pair (𝐶𝑖, 𝐶𝑗) where 𝐶𝑖 and 𝐶𝑗 overlap, the constraint is:

* If *Ci = (Pi, Di)* and *Cj = (Pj, Dj)*, then:
  + *Di ≠ Dj* (different days) **OR** *Pi ≠ Pj* (different professors).

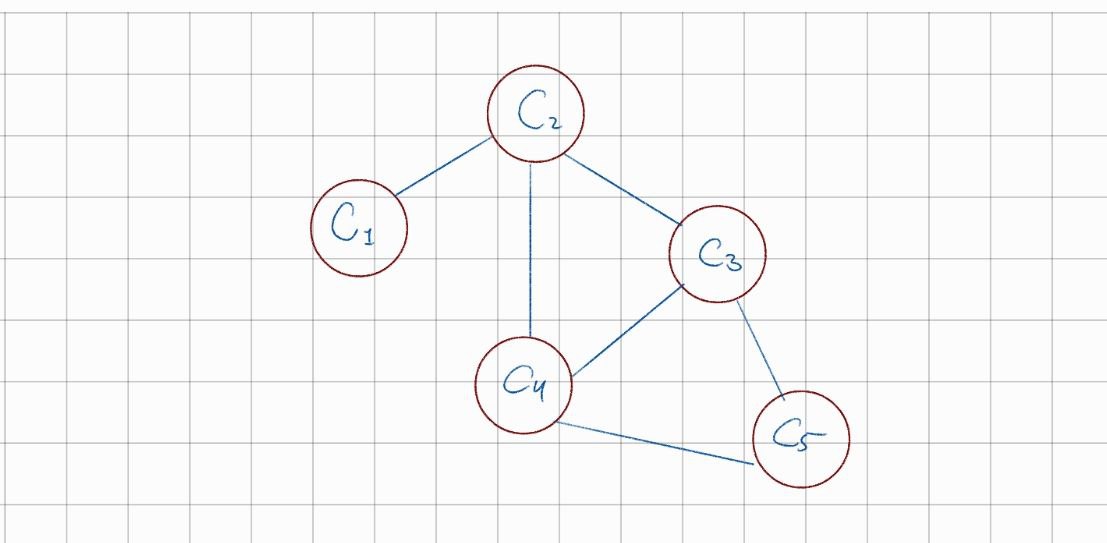
The overlapping pairs and their constraints are:

* (C1​, C2​): If C1 = (P1, D1), C2 = (P2, D2), then D1 ≠ D2 ∨ P1 ≠ P2​.
* (C2, C3): If C2 = (P2, D2), C3 = (P3, D3), then D2 ≠ D3 ∨ P2 ≠ P3.
* (C2, C4): If C2 = (P2, D2), C4 = (P4, D4), then D2 ≠ D4 ∨ P2 ≠ P4.
* (C3, C4): If C3 = (P3, D3), C4 = (P4, D4), then D3 ≠ D4 ∨ P3 ≠ P4.
* (C3, C5): If C3 = (P3, D3), C5 = (P5, D5), then D3 ≠ D5 ∨ P3 ≠ P5​.
* (C4, C5): If C4 = (P4, D4), C5 = (P5, D5), then D4 ≠ D5 ∨ P4 ≠ P5​.

**Implicit Constraint**:

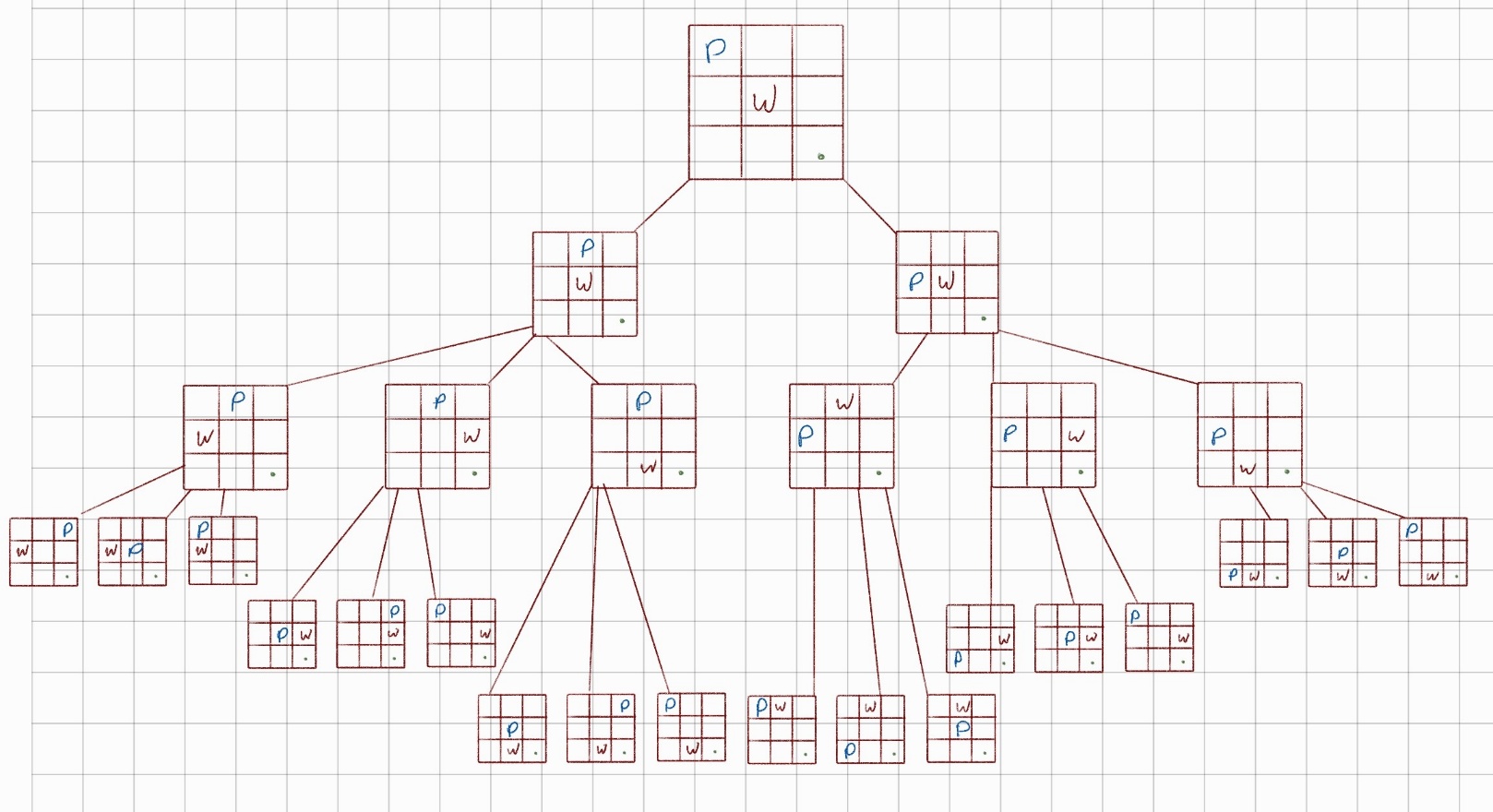
* All classes must be scheduled (i.e., each variable *Ci​* must be assigned a value from its domain). This is ensured by the CSP solving process, which seeks a complete assignment.

**b)**

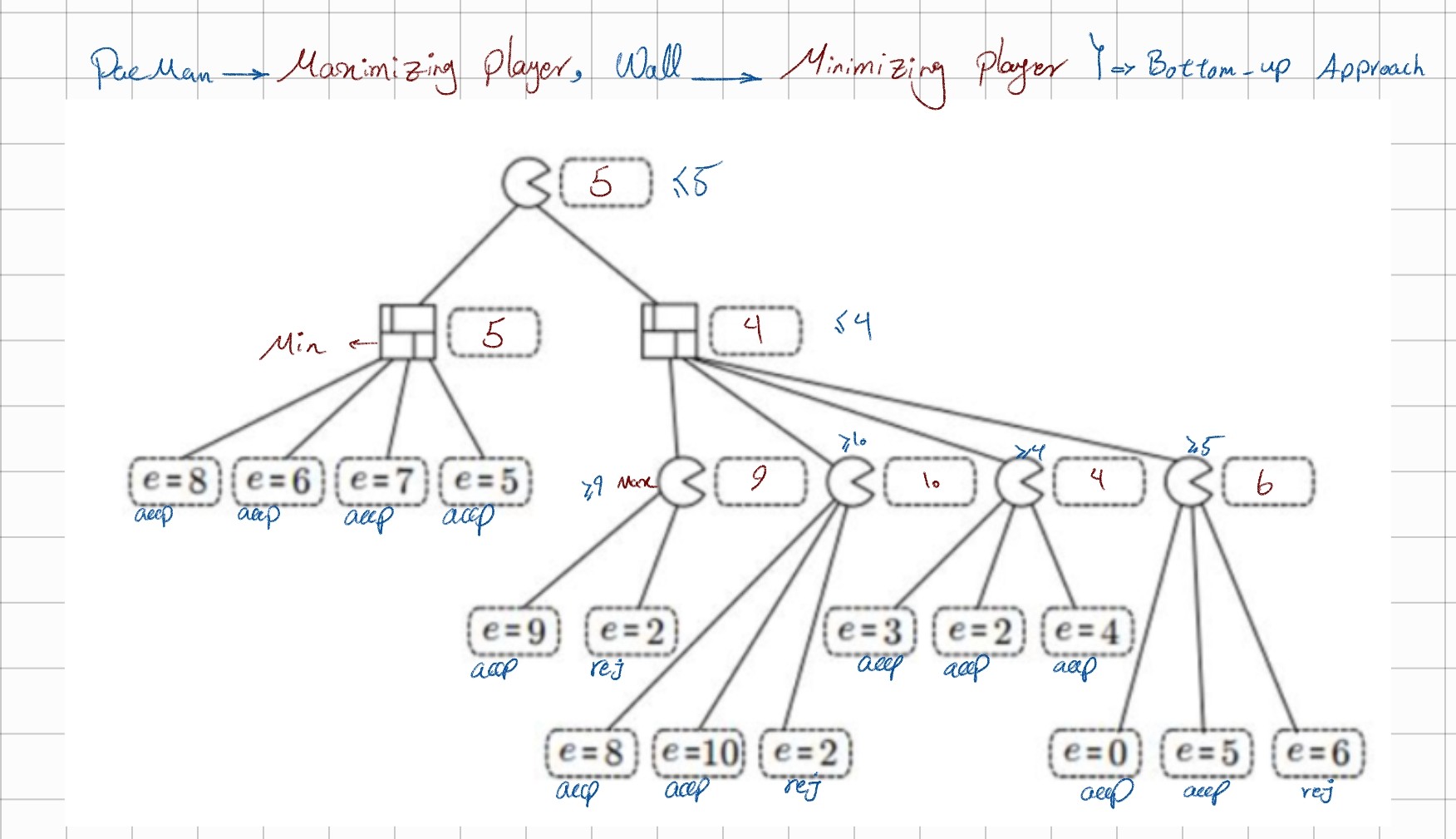
****

**Q2:**

**a)**



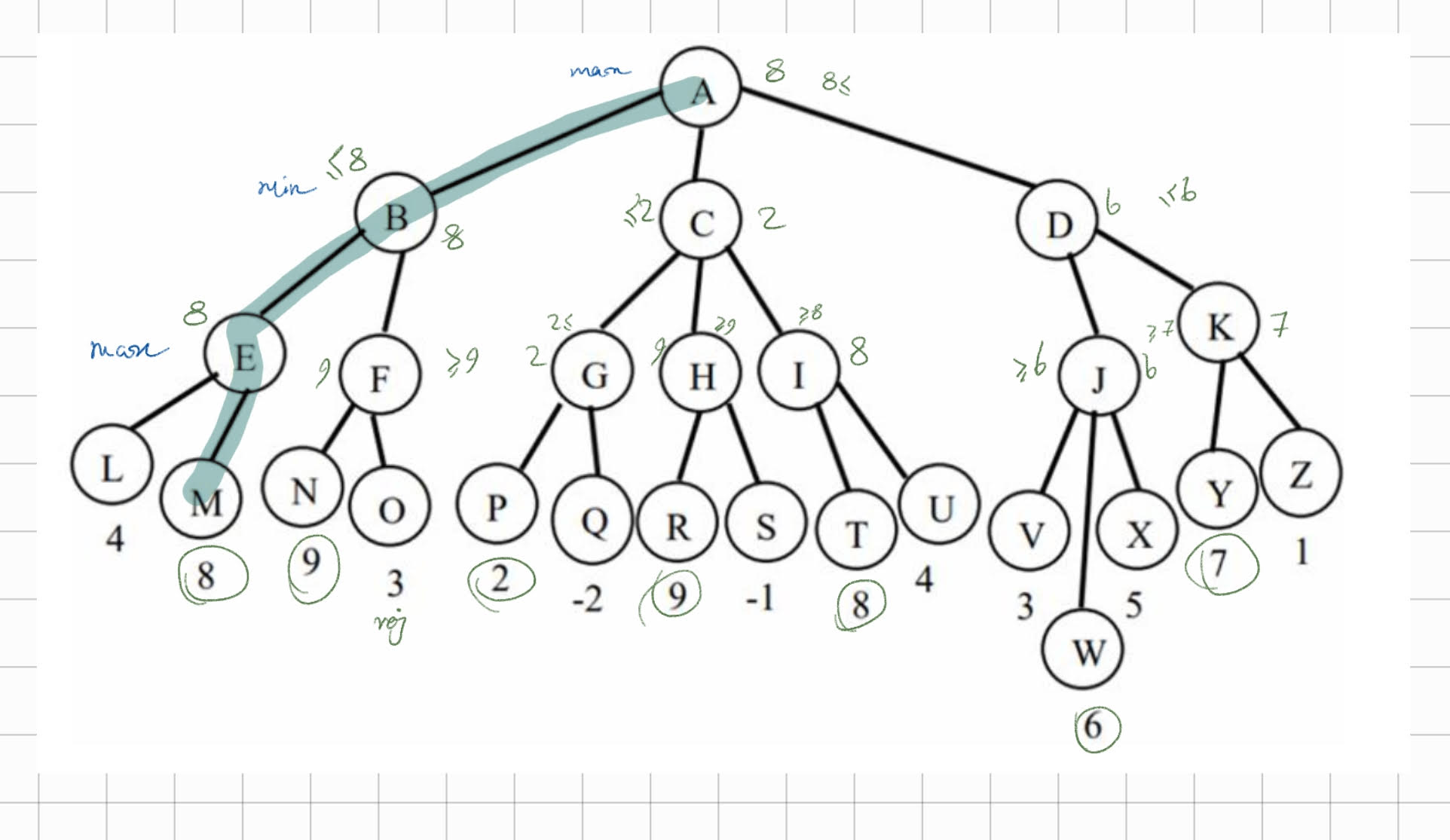
**Game Tree Depth:** **3**, **Game Value:** **0** (Pacman Only moved twice and couldn’t reach the goal)

**b)**

*e* = 2, *e* = 6 are not examined by the Alpha-Beta Pruning

**Q5:**

**a)**

**b)**

First move of max will be **A** to **B**

**c)**

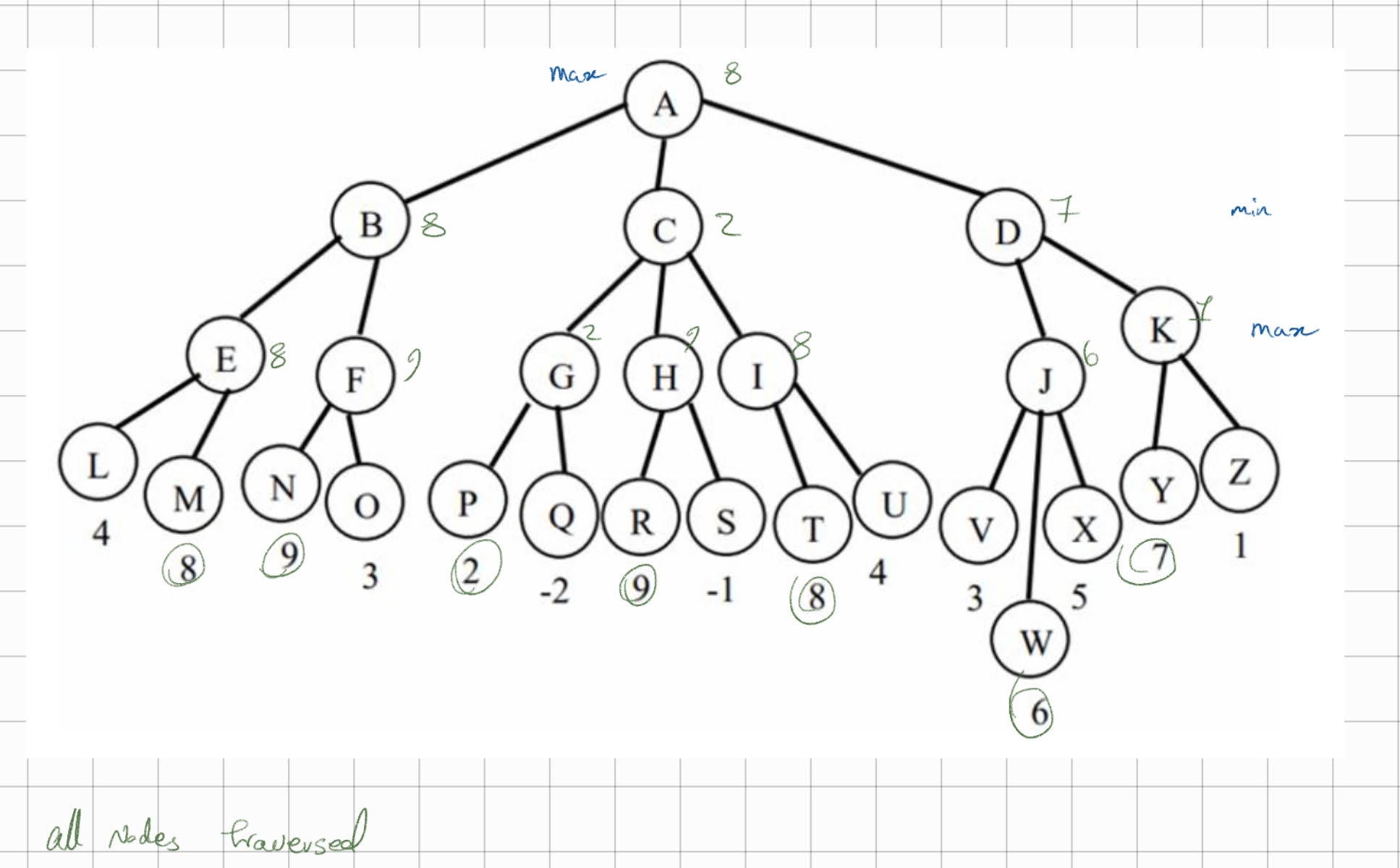
In F sees the N and it matches with condition so O is pruned. With this logic all the pruned nodes will be: {O, H, I, R, S, T, U, X, K, Y, Z}

**d)**

1. **Minimax value at the root** does *not* change. Alpha-beta always computes the same minimax value, regardless of child-visit order.

2. **Number of prunings can change.** A better (or worse) move ordering can lead to more (or fewer) cut-offs.

**e)**

****