Minimax Search

CMPUT 355: Games, Puzzles, and Algorithms

Lecture Outline

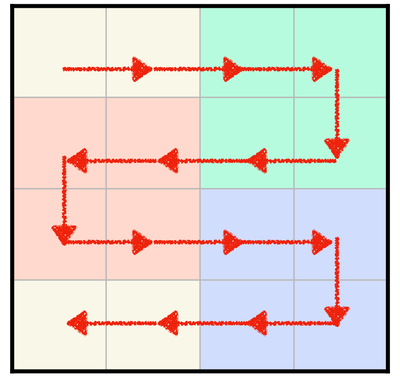
1. Logistics & Recap
2. Game trees & strategies
3. Minimax search

Logistics

- **Slides**
  - Github Pages appears to be broken?
  - Today's slides are available on Canvas
- **Practice quiz questions #2:**
  - Now available
  - Answers will be posted tomorrow (Feb 3)
- **Quiz 1 marks available**
  - Solutions have been posted
  - We will add scans of the quizzes to Canvas in the next day or two
  - Any concerns about grading should be raised in a comment on the Canvas submission
- **Quiz 2:** Friday, **Feb 6**
  - In-class, full 50 minutes
  - **No need to email** if you have to miss it;
    up to 3 missed quizzes replaced by final exam **automatically**
  - Coverage: up to the end of **last Friday's lecture** (Sliding tiles & subgoals)
  - Questions will be very similar to practice questions

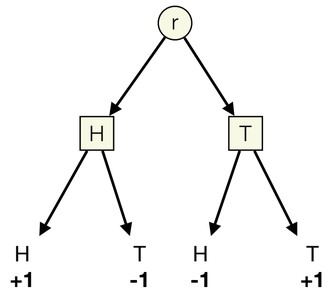
Recap: Puzzles & Search

- Many puzzles can be represented as a **search graph**
  - **Nodes** for **positions**
  - **Edges** between positions that can be reached with a **single move**
- **Search algorithms:** Compute a solution in the graph (a **path** from start to target)
  - **Breadth-first search:** Search all 1-edge paths, then 2-edge paths, etc.
    - guaranteed to find **shortest** solution
    - Potentially very slow, must search **every node** in the worst case
  - **A*:** Search paths with low **total estimated cost** first
    - Estimates from a **heuristic**
    - Guaranteed to find **cheapest** solution if heuristic is **admissible**
    - Can be very slow for unsolvable positions
  - **Subgoal schedule:** Solve **intermediate problems** in order instead of the whole thing at once
    - Each intermediate problem may have a vastly smaller search space
    - Solution may not be optimal



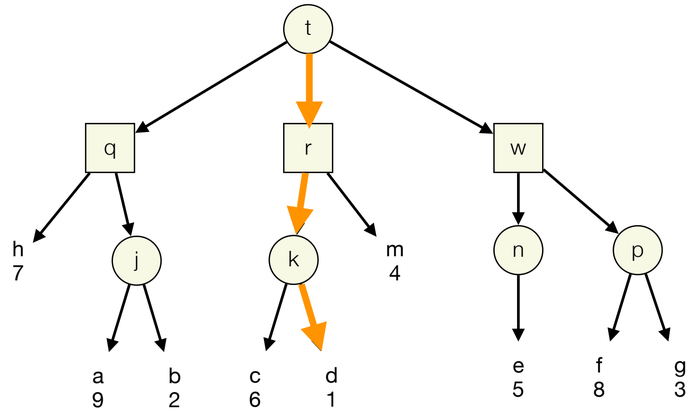
Games vs. Puzzles

- In a **puzzle:**
  - **Single** player
  - Sequence of moves completely determines the outcome
  - **Solution:** sequence of moves from start to target
- In a (two-player zero-sum alternating-move) **game:**
  - **Two** players (with exactly opposed goals)
  - Outcome depends on the moves of **both** players
  - So it's not sufficient to specify a sequence of moves (**why?**)
- **Question:** What is a **solution** in a two-player game?



Game Trees

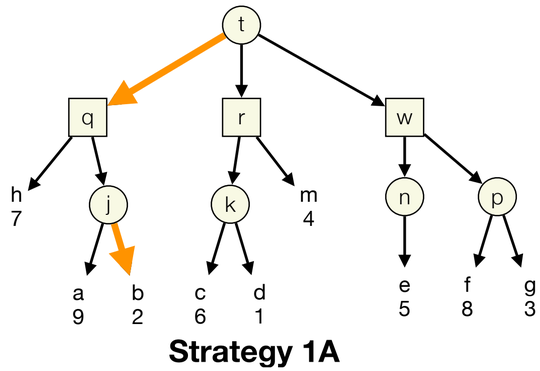
- In a game, the **position** does not completely describe the situation (**why?**)
  - **State:** The position plus the player-to-move
- Game tree:
  - **Nodes** for states
  - **Edges** for states reachable by a single move (typically **directed**)
  - **Leaf nodes** labelled with the outcome for **player 1** (**why not both?**)
- **Player 1** chooses action at the root
  - **Player 2** chooses action at the next state, etc.
- A game **outcome** is determined by the **path** from the root to a leaf node



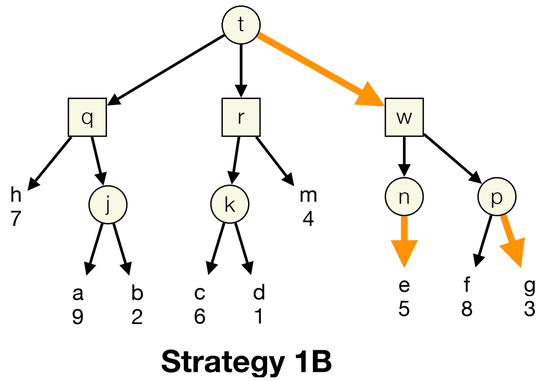
Strategies

A strategy for a given player specifies an action at each state where that player is to move

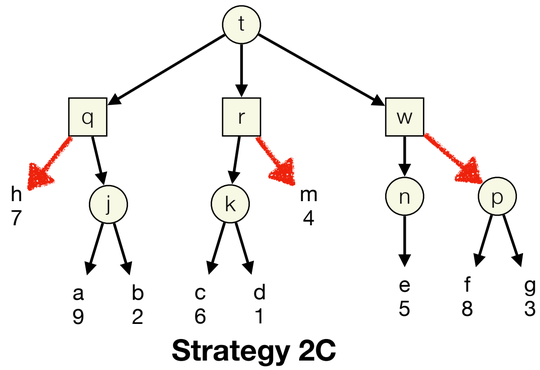
- E.g., for **player 1:**
  - $1A = \{t \rightarrow q, j \rightarrow b\}$
  - $1B : \{t \rightarrow w, n \rightarrow e, p \rightarrow q\}$
- For **player 2:**
  - $2C = \{q \rightarrow h, r \rightarrow m, w \rightarrow p\}$
  - $2D = \{q \rightarrow j, r \rightarrow k, w \rightarrow n\}$



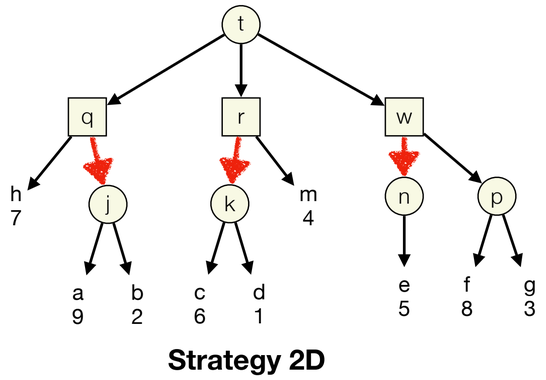
Strategy 1A



Strategy 1B

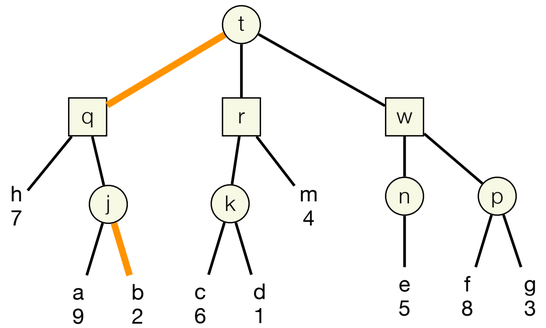


Strategy 2C

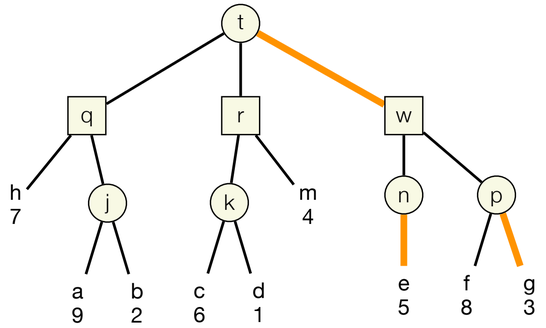


Strategy 2D

Comparing Strategies



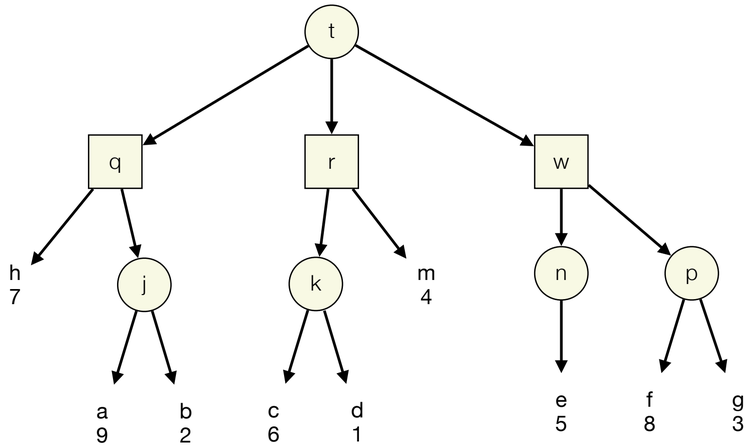
Strategy 1A

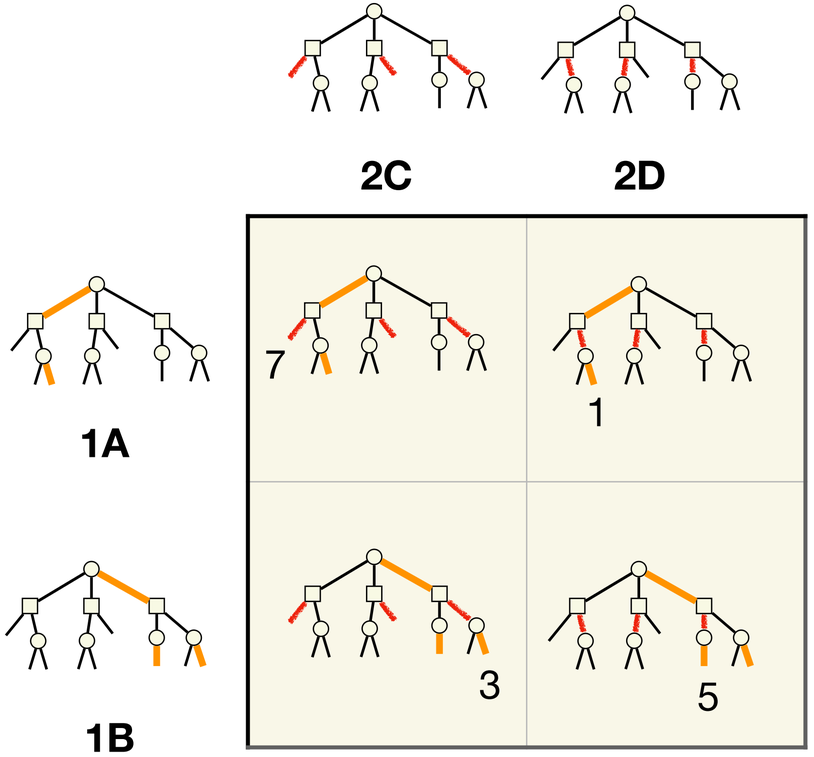


Strategy 1B

- **Question:** Which strategy is better?
- Need to know **both strategies** to determine the outcome!

Induced Normal Form





- Every **combination of strategies** yields a specific outcome
- For a set of strategies, we can arrange them in a table called an (induced) **normal form**

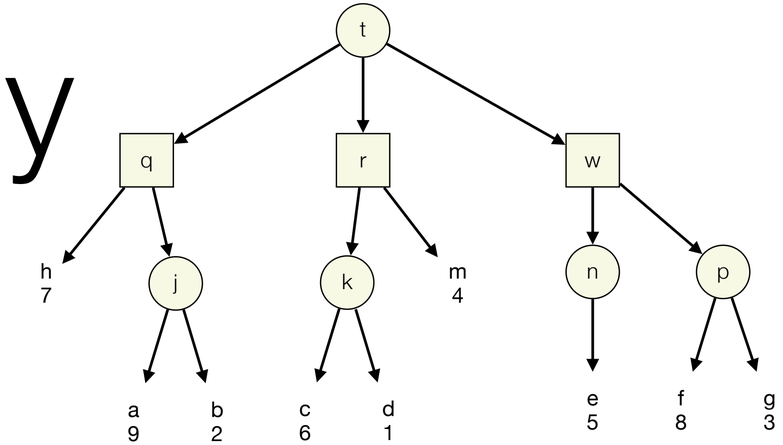
Question: Which strategy should **player 1** choose?

- If P1 picks 1A, **min** score is **1**
- If P2 picks 1B, **min** score is **3**

Question: Which strategy should **player 2** choose?

- If P2 picks 2C, **max** score is **7**
- If P2 picks 2D, **max** score is **5**

Minimax Strategy



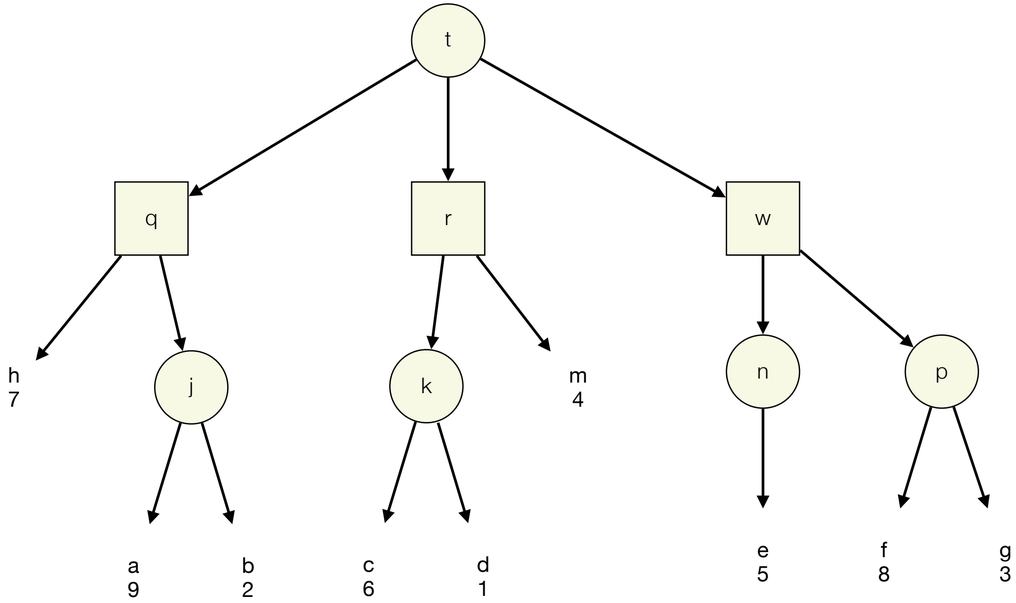
- Player 1's **minimax strategy** is the strategy that has the **best minimum outcome**
- To compute:
  - For every possible strategy of P1, write down the **minimum outcome against every possible strategy of P2**
  - Strategy with the largest minimum outcome is a **minimax strategy**
- The value that P1 gets if both players choose their minimax strategy is the game's **minimax value**

Questions:

1. How many strategies does **P1** have in **this game**?
2. How many strategies does **P2** have in **this game**?
3. How many strategies does **P1** have in a game where:
   - 2 possible moves at each position
   - P1 will get to move exactly n times

Minimax Search

- Fortunately, we don't need to consider every possible strategy
- We can start at the bottom of the tree, and compute minimax value for the **subgame** that starts at that node
    - If we get there, we can assume that we will get the **minimax value** of the subgame (**why?**)

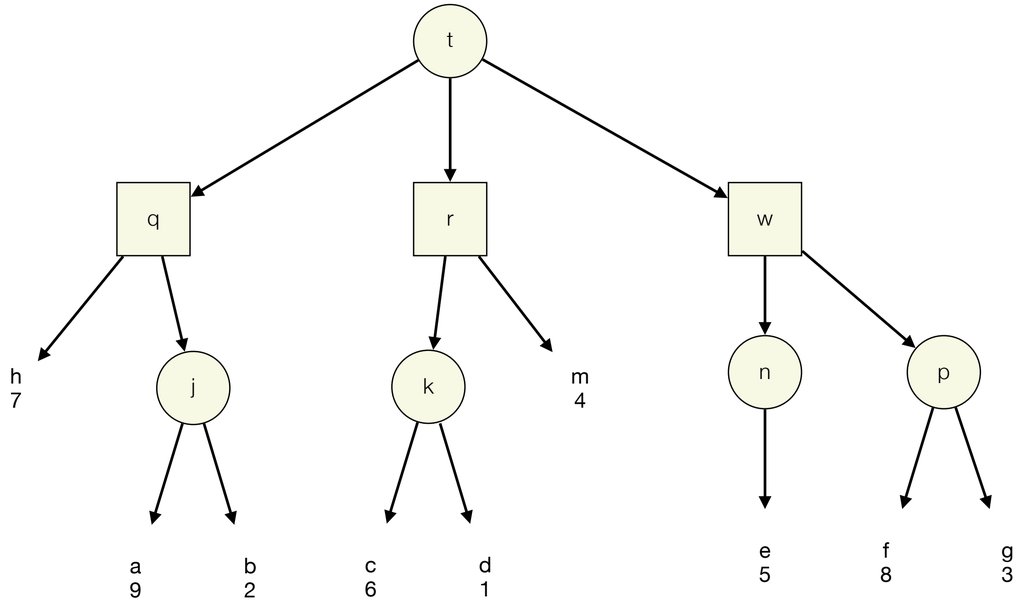


Minimax Pseudocode

assume P1 plays at root
assume players alternate turns

```
def score(s):
    return P1's score at state s

def minimax(s):
    if terminal(s):
        return score(s)
    if player(s) == 1:
        return max{minimax(c) for all c in children(s)}
    if player(s) == 2:
        return min{minimax(c) for all c in children(s)}
```



Questions:

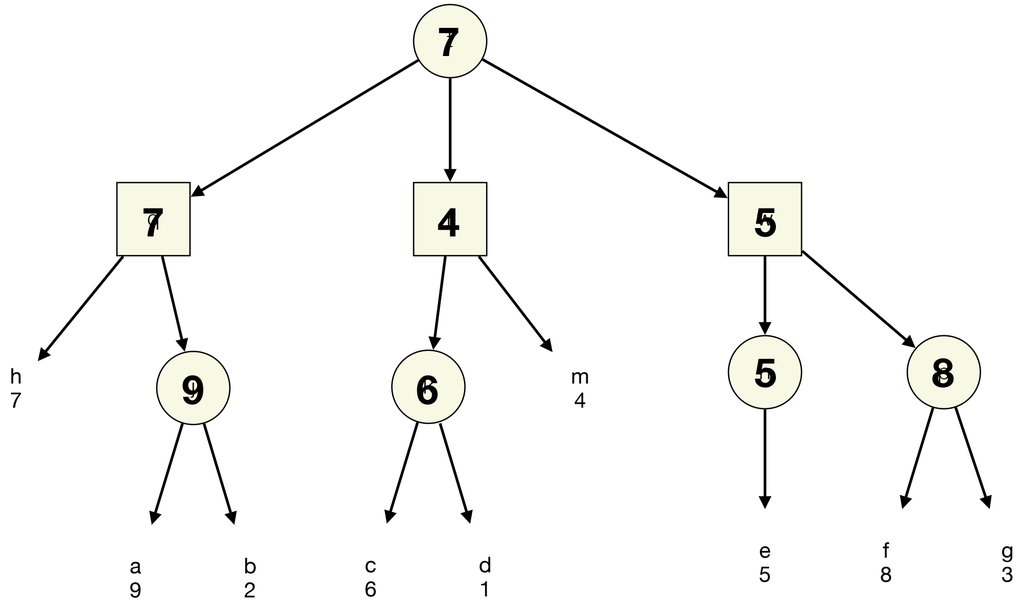
1. Why does **P1** take the **max** over all children?
2. Why does **P2** take the **min** over all children?
3. How do we use minimax values to derive a **strategy**?
4. In what **order** does this algorithm explore the tree?
5. How many **nodes** does this algorithm consider?

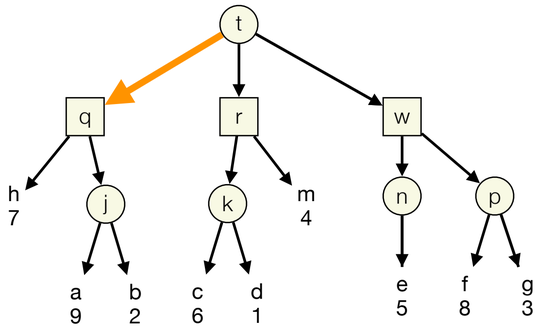
Minimax Example

assume P1 plays at root
assume players alternate turns

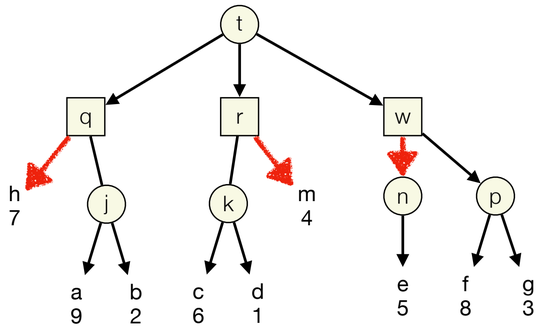
```
def score(s):
    return P1's score at state s

def minimax(s):
    if terminal(s):
        return score(s)
    if player(s) == 1:
        return max{minimax(c) for all c in children(s)}
    if player(s) == 2:
        return min{minimax(c) for all c in children(s)}
```





Minimax strategy for P1



Minimax strategy for P2

Summary

- **Games** have two players, each with opposing goals
  - A **state** represents both the position and the **player-to-move**
  - Can represent a game as a **tree** of states, with edges for each move
  - Label each leaf node with the outcome for P1
- A **strategy** is an assignment of an action to each state at which a player is to move
  - The outcome is determined by **both** player's strategies
- A **minimax strategy** is a strategy with the **best worst-case** over all of the **opponent's** strategies
  - When both players play minimax strategies, the result is the **minimax value**
  - Can use **minimax search** to find the minimax value of the game recursively
- **Minimax search:**
  - At each of P1's states, find the maximum minimax value among all the children
  - At each of P2's states, find the minimum minimax value among all the children