

CS540

Midterm Review

Yingyu Liang

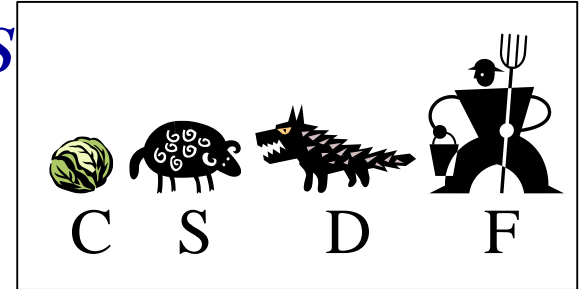
`yliang@cs.wisc.edu`

Computer Sciences Department
University of Wisconsin, Madison

Uninformed Search

The search problem

- **State space** S : all valid configurations
- **Initial states (nodes)** $I = \{(CSDF,)\} \subseteq S$
 - Where's the boat?
- **Goal states** $G = \{(\,,CSDF)\} \subseteq S$
- **Successor function** $succs(s) \subseteq S$: states reachable in one step (one arc) from s
 - $succs((CSDF,)) = \{(CD, SF)\}$
 - $succs((CDF,S)) = \{(CD,FS), (D,CFS), (C, DFS)\}$
- **Cost** $(s,s')=1$ for all arcs. (weighted later)
- The search problem: find a solution path from a state in I to a state in G .
 - Optionally minimize the cost of the solution.



General State-Space Search Algorithm

```
function general-search(problem, QUEUEING-FUNCTION)
;; problem describes the start state, operators, goal test, and
;; operator costs
;; queueing-function is a comparator function that ranks two states
;; general-search returns either a goal node or "failure"

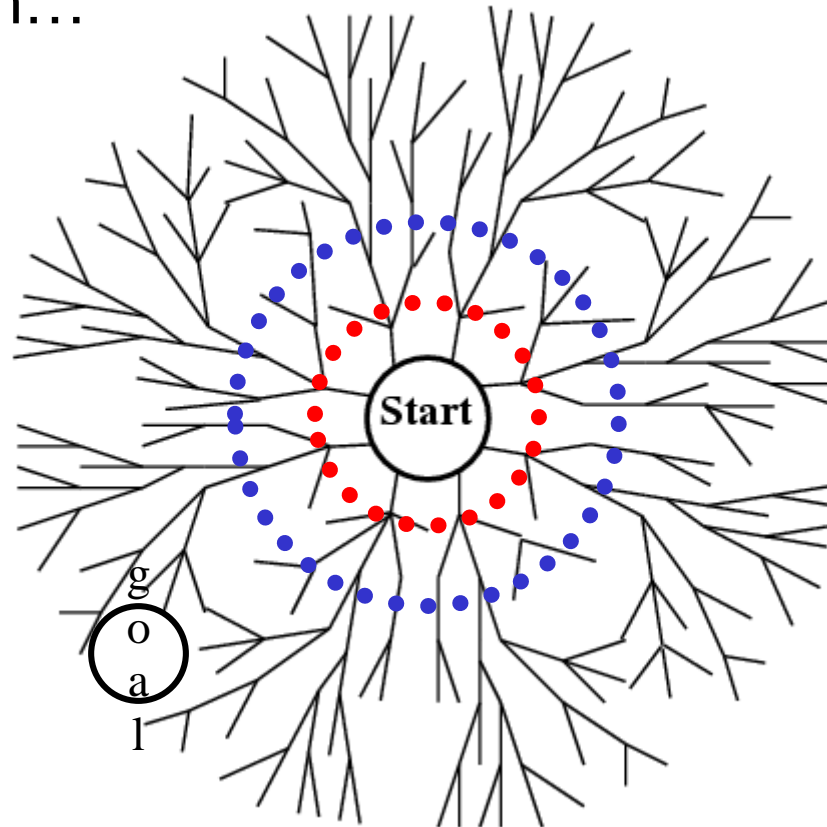
nodes = MAKE-QUEUE(MAKE-NODE(problem.INITIAL-STATE))
loop
  if EMPTY(nodes) then return "failure"
  node = REMOVE-FRONT(nodes)
  if problem.GOAL-TEST(node.STATE) succeeds
then return node
  nodes = QUEUEING-FUNCTION(nodes, EXPAND(node,
                                problem.OPERATORS))
;; succ(s)=EXPAND(s, OPERATORS)
;; Note: The goal test is NOT done when nodes are generated
;; Note: This algorithm does not detect loops
end
```

Search on Trees: Breadth-first search (BFS)

Expand the shallowest node first




- Examine states **one** step away from the initial states
- Examine states **two** steps away from the initial states
- and so on...

ripple

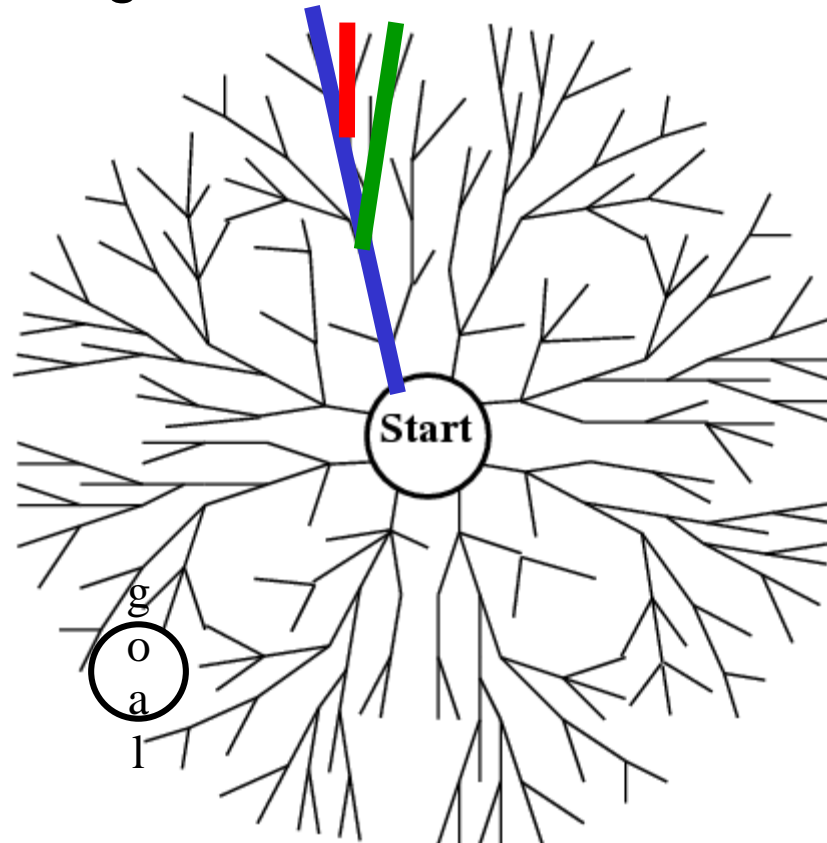


Depth-first search

Expand the deepest node first

1. Select a direction, go deep to the end 
2. Slightly change the end 
3. Slightly change the end some more... 

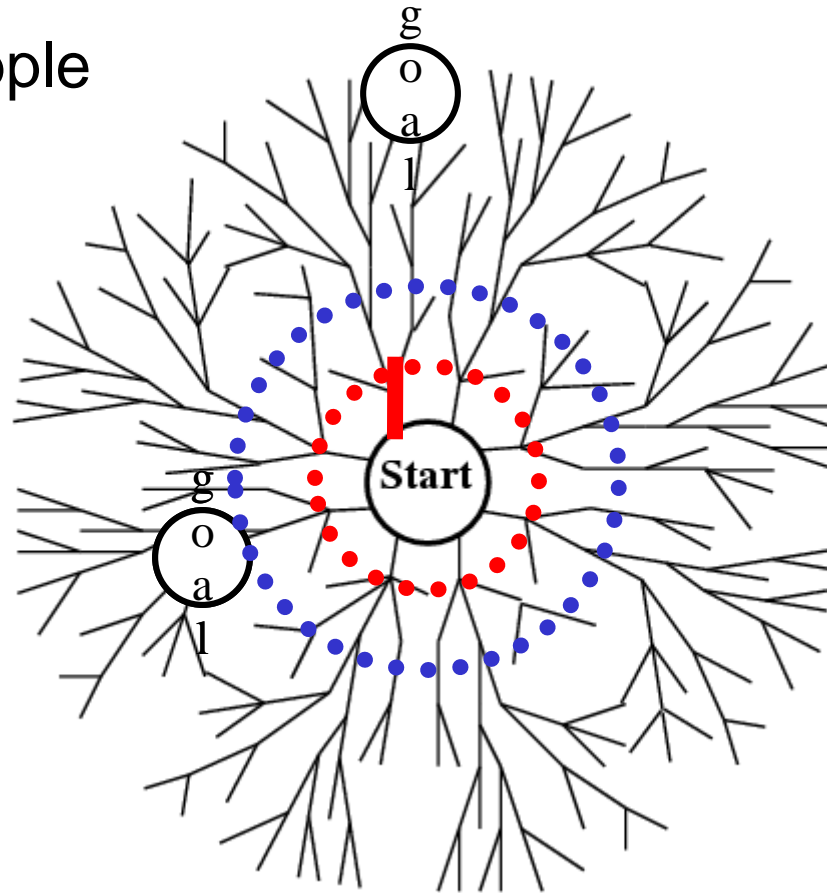
fan



Iterative deepening

1. DFS, but stop if path length > 1 .
2. If goal not found, repeat DFS, stop if path length > 2 .
3. And so on...

fan within ripple



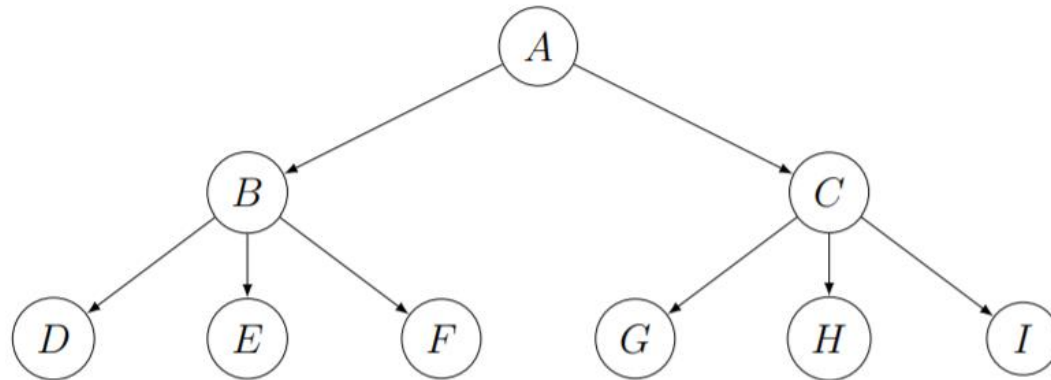
What you should know

- Problem solving as search: state, successors, goal test
- Uninformed search
 - Breadth-first search
 - Uniform-cost search
 - Depth-first search
 - **Iterative deepening** ★
 - Bidirectional search
- Can you unify them (except bidirectional) using the same algorithm, with different priority functions?
- Performance measures
 - Completeness, optimality, time complexity, space complexity



Example

13. Which order of goal check is impossible with Breadth First Search, without specifying the order of successors when putting them in the queue?



(A) H before A

(B) B before G

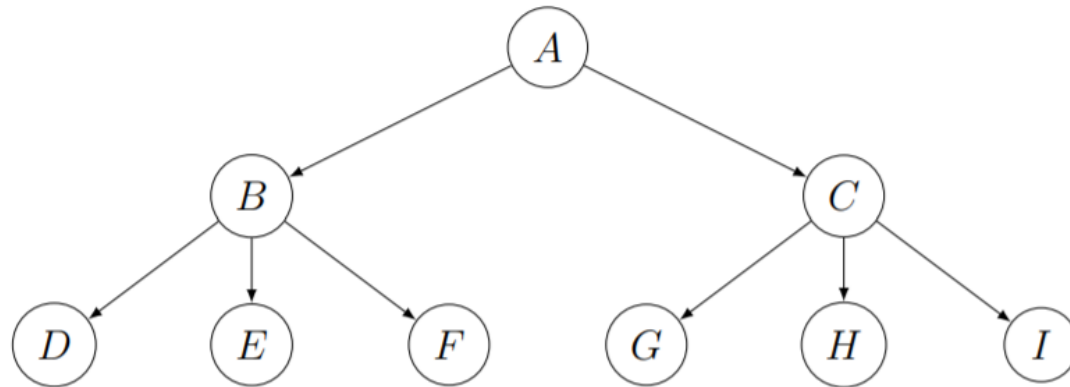
(C) I before D

(D) all of the above

(E) none of the above

Example

13. Which order of goal check is impossible with Breadth First Search, without specifying the order of successors when putting them in the queue?



- (A) H before A (B) B before G (C) I before D
(D) all of the above (E) none of the above

A. For BFS we will always goal-check and expand a parent level before child level. So a grandchild H cannot be goal-checked before the root.

Informed Search

Uninformed vs. informed search

Uninformed search (BFS, uniform-cost, DFS, ID etc.)

Knows the actual path cost $g(s)$ from start to a node s in the fringe, but that's it.



Informed search



also has a heuristic $h(s)$ of the cost from s to goal. ('h'= heuristic, non-negative)

Can be **much faster** than uninformed search.

Third attempt: A* search

- use $g(s)+h(s)$, but the heuristic function $h()$ has to satisfy $h(s) \leq h^*(s)$, where $h^*(s)$ is the true cost from node s to the goal.
- Such heuristic function $h()$ is called **admissible**.
 - An admissible heuristic never over-estimates



It is always
optimistic

- A search with admissible $h()$ is called **A* search**.

What you should know

Know why best-first greedy search is bad.

Thoroughly understand A^*

Trace simple examples of A^* execution.

Understand admissible heuristics.

Example

4. Assume h and h' are any admissible heuristic functions. Consider any real number a and the new heuristic $h''(s) = ah(s) + (1-a)h'(s)$. For what range of a is h'' guaranteed to be a heuristic function? (Clarification: h'' must be admissible.)

Example

4. Assume h and h' are any admissible heuristic functions. Consider any real number a and the new heuristic $h''(s) = ah(s) + (1-a)h'(s)$. For what range of a is h'' guaranteed to be a heuristic function? (Clarification: h'' must be admissible.)

The only a that will guarantee h'' to be admissible for any admissible h and h' is in the interval $[0, 1]$.

Advanced Search: Optimization

Optimization problems

Previously we want a **path** from start to goal

Uninformed search: $g(s)$: Iterative Deepening

Informed search: $g(s)+h(s)$: A^*

Now a different setting:

Each state s has a **score** $f(s)$ that we can compute

The goal is to find the state with the **highest score**, or a **reasonably high score**

Do not care about the path

This is an **optimization problem**

Enumerating the states is intractable

Even previous search algorithms are too expensive

Hill climbing algorithm

1. Pick initial state s
2. Pick t in neighbors(s) with the largest $f(t)$
3. IF $f(t) \leq f(s)$ THEN stop, return s
4. $s = t$. GOTO 2.

- Not the most sophisticated algorithm in the world.
- Very greedy.
- Easily stuck.

your enemy:

**local
optima**

Repeated hill climbing with random restarts

Very simple modification

1. When stuck, pick a random new start, run basic hill climbing from there.
2. Repeat this k times.
3. Return the best of the k local optima.

- Can be very effective
- Should be tried whenever hill climbing is used

Example

8. Consider a state space where the states are all positive integers. State i has two neighbors $i - 1$ and $i + 1$ (except for $i = 1$ which only has one neighbor $i = 2$). State i has score $\frac{(-1)^i}{i}$. If one runs the hill climbing algorithm, how many initial states can reach the global maximum?
- (A) 0 (B) 1 (C) 2 (D) 3 (E) none of the above

Example

8. Consider a state space where the states are all positive integers. State i has two neighbors $i - 1$ and $i + 1$ (except for $i = 1$ which only has one neighbor $i = 2$). State i has score $\frac{(-1)^i}{i}$. If one runs the hill climbing algorithm, how many initial states can reach the global maximum?
- (A) 0 (B) 1 (C) 2 (D) 3 (E) none of the above

D. The scores for states 1,2,3,... are -1, 1/2, -1/3, 1/4, The only states that can reach the global maximum 1/2 is 1,2,3.

Simulated Annealing

1. Pick initial state s
2. Randomly pick t in neighbors(s)
3. IF $f(t)$ better THEN accept $s \leftarrow t$.
4. ELSE /* t is worse than s */
5. accept $s \leftarrow t$ with a small probability
6. GOTO 2 until bored.

How to choose the small probability?

idea: p decreases with time, also as the 'badness'
 $|f(s) - f(t)|$ increases

Typical choice:

$$\exp\left(-\frac{|f(s) - f(t)|}{Temp}\right)$$

Boltzmann
distribution

Example

11. In simulated annealing we move from s to an inferior neighbor t with probability $\exp\left(-\frac{|f(s)-f(t)|}{T}\right)$ where T is the temperature. What is the probability we stay at s instead of moving to t ?
- (A) $\exp\left(-\frac{|f(t)-f(s)|}{T}\right)$ (B) $\exp\left(\frac{|f(s)-f(t)|}{T}\right)$ (C) $\exp\left(1 - \frac{|f(s)-f(t)|}{T}\right)$ (D) $1 - \exp\left(-\frac{|f(s)-f(t)|}{T}\right)$ (E) none of the above

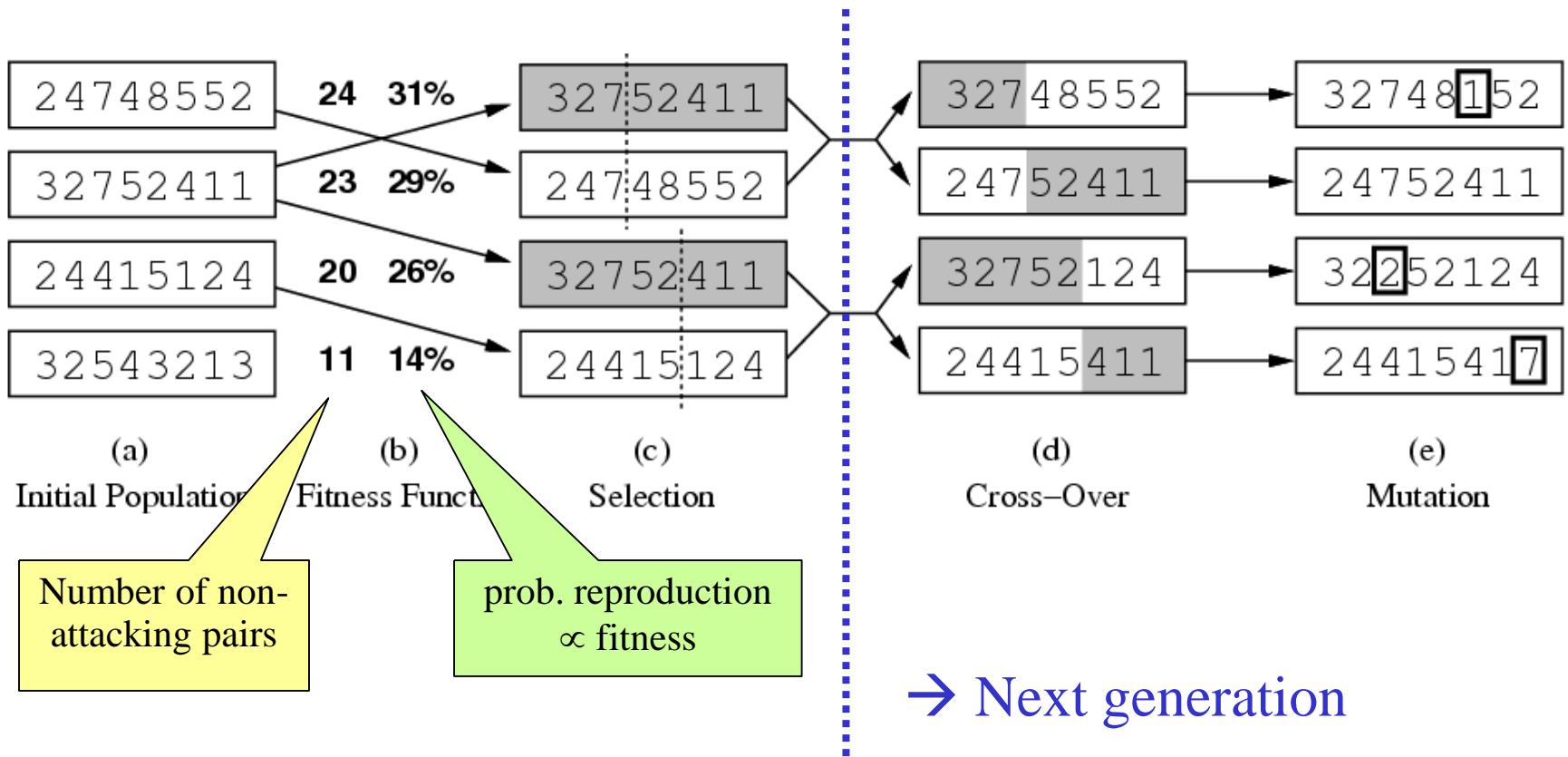
Example

11. In simulated annealing we move from s to an inferior neighbor t with probability $\exp\left(-\frac{|f(s)-f(t)|}{T}\right)$ where T is the temperature. What is the probability we stay at s instead of moving to t ?
- (A) $\exp\left(-\frac{|f(t)-f(s)|}{T}\right)$ (B) $\exp\left(\frac{|f(s)-f(t)|}{T}\right)$ (C) $\exp\left(1 - \frac{|f(s)-f(t)|}{T}\right)$ (D) $1 - \exp\left(-\frac{|f(s)-f(t)|}{T}\right)$ (E) none of the above

D. It is just normalization.

Genetic algorithm

Genetic algorithm: a special way to generate neighbors, using the analogy of **cross-over**, **mutation**, and **natural selection**.



Game Playing

Two-player zero-sum discrete finite deterministic games of perfect information

Definitions:

Zero-sum: one player's gain is the other player's loss.
Does not mean *fair*.

Discrete: states and decisions have discrete values

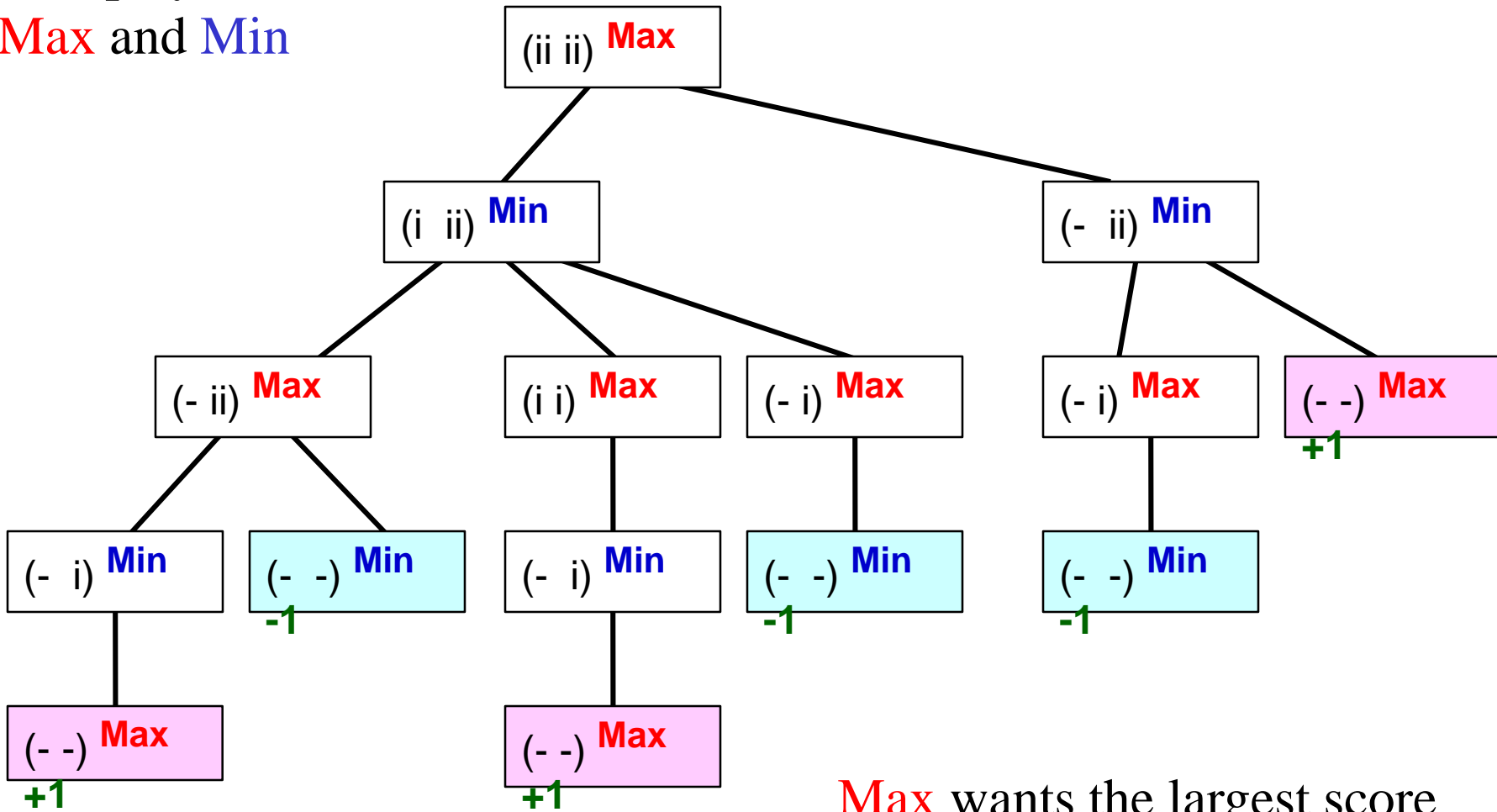
Finite: finite number of states and decisions

Deterministic: no coin flips, die rolls – no chance

Perfect information: each player can see the complete game state. No simultaneous decisions.

The game tree for II-Nim

Two players:
Max and **Min**



Max wants the largest score
Min wants the smallest score

Game theoretic value

Game theoretic value (a.k.a. minimax value) of a node =
the score of the terminal node that will be reached if
both players play optimally.

= The numbers we filled in.

Computed bottom up

In Max's turn, take the max of the children (Max will
pick that maximizing action)

In Min's turn, take the min of the children (Min will
pick that minimizing action)

Implemented as a modified version of DFS: **minimax
algorithm**

Minimax algorithm

function **Max-Value**(s)

inputs:

s: current state in game, Max about to play

output: *best-score (for Max) available from s*

if (s is a terminal state)

then return (terminal value of s)

else

$\alpha := -\infty$

 for each s' in Succ(s)

$\alpha := \max(\alpha, \text{Min-value}(s'))$

return α

function **Min-Value**(s)

output: *best-score (for Min) available from s*

if (s is a terminal state)

then return (terminal value of s)

else

$\beta := \infty$

 for each s' in Succs(s)

$\beta := \min(\beta, \text{Max-value}(s'))$

return β

- Time complexity?
 $O(b^m) \leftarrow \text{bad}$
- Space complexity?
 $O(bm)$

Example

3. Consider a game board consisting of two bits initially at 00. Each player can simultaneously flip 1 or 2 bits in a move, but needs to pay the other player one dollar for each bit flipped. The player who achieves 11 wins and collects 10 dollars from the other player. What is the game theoretic value of this game for the first player?
- (A) 8 (B) 10 (C) -8 (D) -10 (E) none of the above

Example

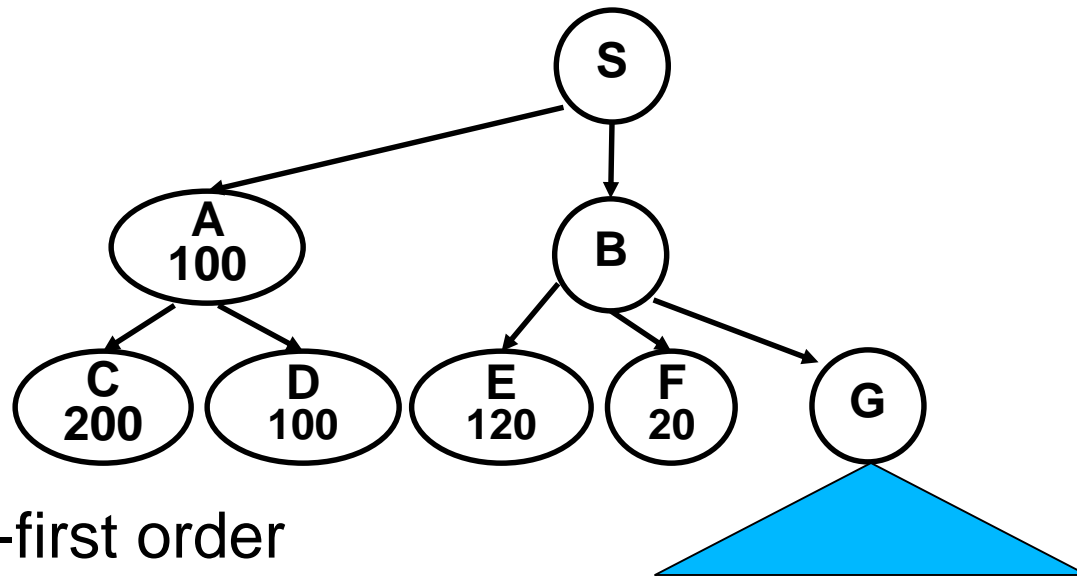
3. Consider a game board consisting of two bits initially at 00. Each player can simultaneously flip 1 or 2 bits in a move, but needs to pay the other player one dollar for each bit flipped. The player who achieves 11 wins and collects 10 dollars from the other player. What is the game theoretic value of this game for the first player?
- (A) 8 (B) 10 (C) -8 (D) -10 (E) none of the above

A. Draw the game tree. It is clear that the first player should immediately flip both bits and force a win, at a cost of 2 dollars but wins 10 dollars. Thus $10-2=8$.

Alpha-Beta Motivation

max

min



Depth-first order

After returning from A, Max can get at least 100 at S

After returning from F, Max can get at most 20 at B

At this point, Max loses interest in B

There is no need to explore G. The subtree at G is pruned. Saves time.

Alpha-beta pruning

function **Max-Value** (s, α , β)

inputs:

s: current state in game, Max about to play

α : best score (highest) for Max along path to s

β : best score (lowest) for Min along path to s

output: $\min(\beta, \text{best-score (for Max) available from s})$

if (s is a terminal state)

then return (terminal value of s)

else for each s' in Succ(s)

$\alpha := \max(\alpha, \text{Min-value}(s', \alpha, \beta))$

if ($\alpha \geq \beta$) then return β /* alpha pruning */

return α

function **Min-Value**(s, α , β)

output: $\max(\alpha, \text{best-score (for Min) available from s})$

if (s is a terminal state)

then return (terminal value of s)

else for each s' in Succs(s)

$\beta := \min(\beta, \text{Max-value}(s', \alpha, \beta))$

if ($\alpha \geq \beta$) then return α /* beta pruning */

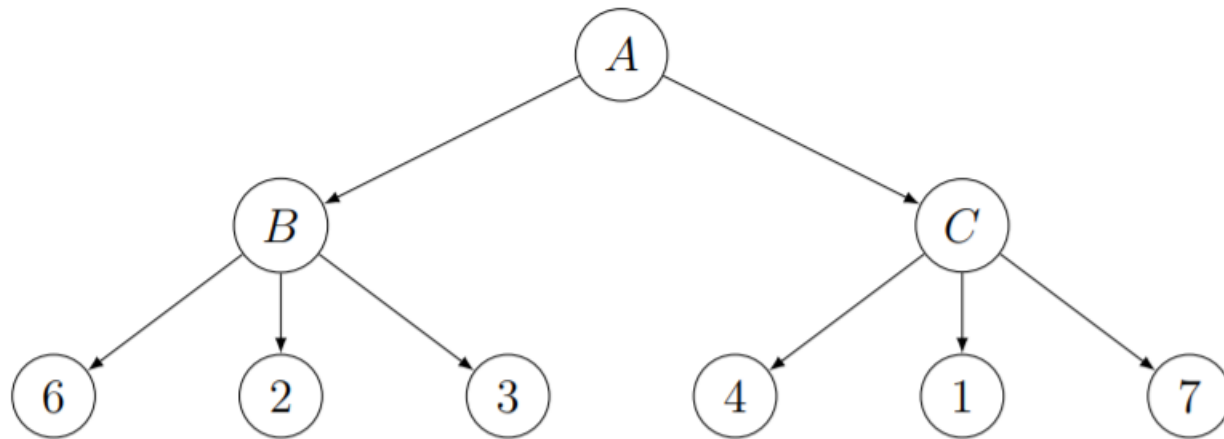
return β

Starting from the root:

Max-Value(root, $-\infty, +\infty$)

Example

12. Which nodes are pruned by alpha-beta pruning? The max player moves first.



(A) 4,1,7

(B) 1,7

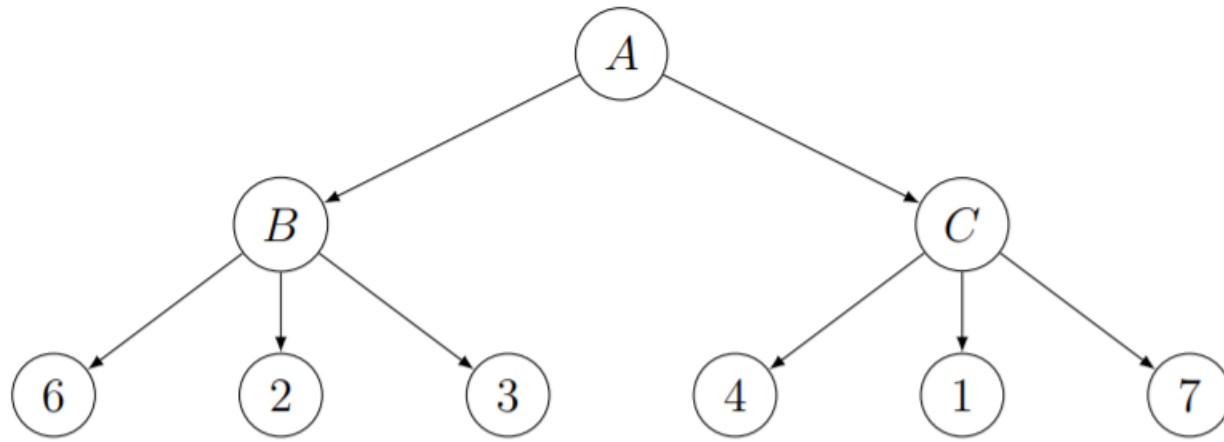
(C) 7

(D) C,4,1,7

(E) none of the above

Example

12. Which nodes are pruned by alpha-beta pruning? The max player moves first.



(A) 4,1,7

(B) 1,7

(C) 7

(D) C,4,1,7

(E) none of the above

C. Run alpha-beta. It will prune 7 after seeing 1.

Math Basics

Probability

Axioms:

- $P(A) \in [0,1]$
- $P(\text{true})=1, P(\text{false})=0$
- $P(A \vee B) = P(A) + P(B) - P(A \wedge B)$

Properties:

- $P(\neg A) = 1 - P(A)$
- If A can take k different values $a_1 \dots a_k$:
$$P(A=a_1) + \dots P(A=a_k) = 1$$
- $P(B) = \sum_{i=1 \dots k} P(B \wedge A=a_i)$, if A can take k values

Probability

- Joint/marginal/conditional probability

- Chain rule:

$$\begin{aligned} P(A_1, A_2, \dots, A_n) \\ = P(A_1) * P(A_2|A_1) * P(A_3|A_2, A_1) * \dots * P(A_n|A_1, A_2, \dots, A_{n-1}) \end{aligned}$$

- Bayes' rule:

$$P(F|H) = \frac{P(F, H)}{P(H)} = \frac{P(H|F)P(F)}{P(H)}$$

- Independence/conditional independence
- Expectation

Example

2. Let $A \in \{1, 2, 3, 4\}$ and $B \in \{1, 2, 3\}$. To fully specify $P(B | A)$ how many numbers are needed?
- (A) 7 (B) 8 (C) 9 (D) 12 (E) none of the above

Example

2. Let $A \in \{1, 2, 3, 4\}$ and $B \in \{1, 2, 3\}$. To fully specify $P(B | A)$ how many numbers are needed?
- (A) 7 (B) 8 (C) 9 (D) 12 (E) none of the above

B. For a fixed A , we need two numbers (the third value of B is given by normalization). Thus $4 \cdot (3-1) = 8$.

Principal Component Analysis

- **Motivation:** keep only important directions
- **Definition:** the direction where the projections of the data have largest variance
- **Equivalent definition:** the direction where the projections of the data have least reconstruction error
- **Math formulation:** Assume data has zero mean.

$$\max_{\mathbf{v}} \sum_{i=1}^n (\mathbf{v}^\top \mathbf{x}_i)^2 \quad \text{s.t.} \quad \|\mathbf{v}\|_2 = 1$$

- **Computation:** reduces to eigen-decomposition of the covariance, and further reduces to SVD of the data matrix

Example

1. What is the projection of $(0.6, 0.8)$ along the direction of $(1, 0)$?
2. What is the projection of $(1, 0)$ along the direction of $(0.6, 0.8)$?
3. Given four data points in R^2 : $x_1 = (-1, 0)$, $x_2 = (1, 0)$, $x_3 = (0, -0.1)$, $x_4 = (0, 0.1)$. Which of the following directions is the first principal component?
(A) $(1, 0)$ (B) $(1, 1)$ (C) $(0, 1)$ (D) $(-1, 1)$

NLP basics

- Bag-of-Words representation
- N-gram model
- Estimate the N-gram model
- Smoothing: Laplace add-one smoothing

Example

7. In a corpus with n word tokens, the phrase “san francisco” appeared m times. If we estimate probability by frequency (the maximum likelihood estimate), what is the estimated probability $\hat{P}(\text{francisco} \mid \text{san})$?
- (A) $\frac{m}{n}$ (B) $\frac{1}{m}$ (C) $\frac{1}{n}$ (D) $\frac{m+1}{n+v}$ where v is the vocabulary size (E) none of the above

Example

7. In a corpus with n word tokens, the phrase “san francisco” appeared m times. If we estimate probability by frequency (the maximum likelihood estimate), what is the estimated probability $\hat{P}(\text{francisco} \mid \text{san})$?
- (A) $\frac{m}{n}$ (B) $\frac{1}{m}$ (C) $\frac{1}{n}$ (D) $\frac{m+1}{n+v}$ where v is the vocabulary size (E) none of the above

Example

7. In a corpus with n word tokens, the phrase “san francisco” appeared m times. If we estimate probability by frequency (the maximum likelihood estimate), what is the estimated probability $\hat{P}(\text{francisco} \mid \text{san})$?
- (A) $\frac{m}{n}$ (B) $\frac{1}{m}$ (C) $\frac{1}{n}$ (D) $\frac{m+1}{n+v}$ where v is the vocabulary size (E) none of the above

E. We don't have enough information. We need to know how many times “san” appears in the corpus.