Search 2

Shivaram Kalyanakrishnan

Department of Computer Science and Engineering
Indian Institute of Technology Bombay

February 2023

This Lecture

- Informed search (a.k.a. heuristic search)
- Application in game-playing

This Lecture

- Informed search (a.k.a. heuristic search)
- Application in game-playing

Generic Search Template: Pseudocode

- Primary data element is a Node, which a tuple of the form

$$(\mathit{state}, \mathit{pathFromStartState}, \mathit{pathCost}).$$

Generic Search Template: Pseudocode

- Primary data element is a Node, which a tuple of the form

$$(\mathit{state}, \mathit{pathFromStartState}, \mathit{pathCost}).$$

- At every stage of the search,
- some states have been explored
- some states remain unexplored, and
- The *Frontier* is a set of nodes due for imminent expansion.

Generic Search Template: Pseudocode

- Primary data element is a Node, which a tuple of the form

$$(state, pathFromStartState, pathCost).$$

- At every stage of the search,
- some states have been explored
- some states remain unexplored, and
- The *Frontier* is a set of nodes due for imminent expansion.

Frontier $\leftarrow \{Node(startState, (startState), 0)\}$.
Repeat for ever:
  Select a node n from *Frontier*.
  //Expand n.
  If $isGoal(n.state)$:
    Return n.
  For each action a available from $n.state$:
    $s \leftarrow NextState(n.state, a)$.
    $c \leftarrow Cost(n.state, a)$.
    $n' \leftarrow Node(s, n.path + (a, s), n.pathCost + c)$.
    Merge n' with *Frontier*.//Typically insertion; might allow deletions.

Generic Search Template: Pseudocode

- Primary data element is a Node, which a tuple of the form

$$(state, pathFromStartState, pathCost).$$

- At every stage of the search,
- some states have been explored
- some states remain unexplored, and
- The *Frontier* is a set of nodes due for imminent expansion.

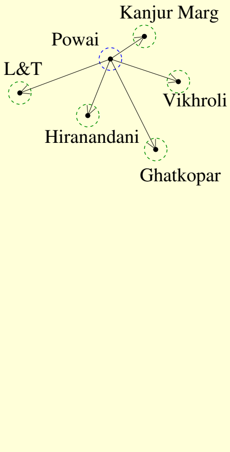
```
Frontier ← {Node(startState, (startState), 0)}.
Repeat for ever:
    Select a node n from Frontier.//How is this selection made?
    //Expand n.
    If isGoal(n.state):
        Return n.
    For each action a available from n.state:
        s ← NextState(n.state, a).
        c ← Cost(n.state, a).
        n′ ← Node(s, n.path + (a, s), n.pathCost + c).
        Merge n′ with Frontier.//Typically insertion; might allow deletions.
```

Incorporating Domain Knowledge into Search

- Have to travel from Powai to Mahim.

Incorporating Domain Knowledge into Search

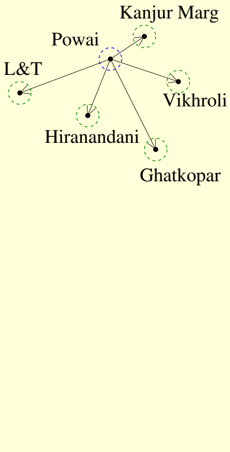
- Have to travel from Powai to Mahim.



- First you expand the Powai node.

Incorporating Domain Knowledge into Search

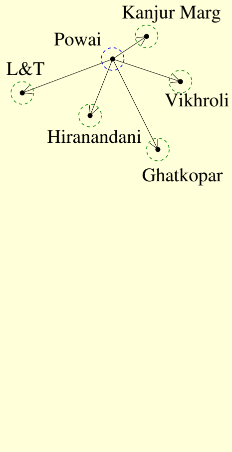
- Have to travel from Powai to Mahim.



- First you expand the Powai node.
- Which node will you expand next?

Incorporating Domain Knowledge into Search

- Have to travel from Powai to Mahim.



- First you expand the Powai node.
- Which node will you expand next?
- L&T and Hiranandani are geographically closer to Mahim: should that count?

Heuristic Functions and A* Search Algorithm

- A heuristic function $h(n)$ is a guess of $c^\star(n)$, the optimal path-cost-to-goal of (the state in) node n.

Heuristic Functions and A* Search Algorithm

- A heuristic function $h(n)$ is a guess of $c^\star(n)$, the optimal path-cost-to-goal of (the state in) node n.

- $h(n)$ is usually easy to compute. On the previous slide, we implicitly used straight line distance:

$$h(n) = \sqrt{(n.state.x - Mahim.x)^2 + (n.state.y - Mahim.y)^2}.$$

Heuristic Functions and A* Search Algorithm

- A heuristic function $h(n)$ is a guess of $c^\star(n)$, the optimal path-cost-to-goal of (the state in) node n.

- $h(n)$ is usually easy to compute. On the previous slide, we implicitly used straight line distance:

$$h(n) = \sqrt{(n.state.x - Mahim.x)^2 + (n.state.y - Mahim.y)^2}.$$

- Recall that in LCFS, we expand

$$\operatorname*{argmin}_{n \in \text{Frontier}} g(n).$$

Heuristic Functions and A* Search Algorithm

- A heuristic function $h(n)$ is a guess of $c^\star(n)$, the optimal path-cost-to-goal of (the state in) node n.

- $h(n)$ is usually easy to compute. On the previous slide, we implicitly used straight line distance:

$$h(n) = \sqrt{(n.state.x - Mahim.x)^2 + (n.state.y - Mahim.y)^2}.$$

- Recall that in LCFS, we expand

$$\operatorname*{argmin}_{n \in \text{Frontier}} g(n).$$

- In A* search, we expand

$$\operatorname*{argmin}_{n \in \text{Frontier}} \left(g(n) + h(n)\right).$$

Heuristic Functions and A* Search Algorithm

- A heuristic function $h(n)$ is a guess of $c^\star(n)$, the optimal path-cost-to-goal of (the state in) node n.

- $h(n)$ is usually easy to compute. On the previous slide, we implicitly used straight line distance:

$$h(n) = \sqrt{(n.state.x - Mahim.x)^2 + (n.state.y - Mahim.y)^2}.$$

- Recall that in LCFS, we expand

$$\operatorname*{argmin}_{n \in \text{Frontier}} g(n).$$

- In A* search, we expand

$$\operatorname*{argmin}_{n \in \text{Frontier}} \left(g(n) + h(n) \right).$$

- $g(n)$ summarises the past (known); $h(n)$ anticipates the future (unknown).

Heuristic Functions and A* Search Algorithm

- A heuristic function $h(n)$ is a guess of $c^\star(n)$, the optimal path-cost-to-goal of (the state in) node n.

- $h(n)$ is usually easy to compute. On the previous slide, we implicitly used straight line distance:

$$h(n) = \sqrt{(n.state.x - Mahim.x)^2 + (n.state.y - Mahim.y)^2}.$$

- Recall that in LCFS, we expand

$$\underset{n \in \text{Frontier}}{\operatorname{argmin}}\; g(n).$$

- In A* search, we expand

$$\underset{n \in \text{Frontier}}{\operatorname{argmin}}\; \left(g(n) + h(n)\right).$$

- $g(n)$ summarises the past (known); $h(n)$ anticipates the future (unknown).
- The addition of $h(n)$ makes A* an informed or heuristic search algorithm.

Heuristic Functions and A* Search Algorithm

- A heuristic function $h(n)$ is a guess of $c^\star(n)$, the optimal path-cost-to-goal of (the state in) node n.

- $h(n)$ is usually easy to compute. On the previous slide, we implicitly used straight line distance:

$$h(n) = \sqrt{(n.state.x - Mahim.x)^2 + (n.state.y - Mahim.y)^2}.$$

- Recall that in LCFS, we expand

$$\operatorname*{argmin}_{n \in \text{Frontier}} g(n).$$

- In A* search, we expand

$$\operatorname*{argmin}_{n \in \text{Frontier}} \left(g(n) + h(n)\right).$$

- $g(n)$ summarises the past (known); $h(n)$ anticipates the future (unknown).
- The addition of $h(n)$ makes A* an informed or heuristic search algorithm.

- A* search was originally conceived for robotic path planning.

Admissible Heuristics

- A heuristic h is admissible if for all nodes n,

$$0 \leq h(n) \leq c^\star(n),$$

where $c^\star(n)$ is the optimal cost-to-goal of *n.state*.

Admissible Heuristics

- A heuristic h is admissible if for all nodes n,

$$0 \leq h(n) \leq c^{\star}(n),$$

  where $c^{\star}(n)$ is the optimal cost-to-goal of $n.state$.

- **Key result**. If A* search is run using an admissible heuristic (and some minor technical conditions hold), then the first goal node it expands will have optimal path-cost from the start state (and the algorithm can terminate).

Admissible Heuristics

- A heuristic h is admissible if for all nodes n,

$$0 \leq h(n) \leq c^{\star}(n),$$

  where $c^{\star}(n)$ is the optimal cost-to-goal of *n.state*.

- **Key result**. If $A^{\star}$ search is run using an admissible heuristic (and some minor technical conditions hold), then the first goal node it expands will have optimal path-cost from the start state (and the algorithm can terminate).

- Is straight line distance an admissible heuristic for navigation?

Admissible Heuristics

- A heuristic h is admissible if for all nodes n,

$$0 \leq h(n) \leq c^\star(n),$$

where $c^\star(n)$ is the optimal cost-to-goal of *n.state*.

- **Key result**. If A* search is run using an admissible heuristic (and some minor technical conditions hold), then the first goal node it expands will have optimal path-cost from the start state (and the algorithm can terminate).

- Is straight line distance an admissible heuristic for navigation?

  Yes.

Admissible Heuristics

- A heuristic h is admissible if for all nodes n,

$$0 \leq h(n) \leq c^{\star}(n),$$

where $c^{\star}(n)$ is the optimal cost-to-goal of $n.state$.

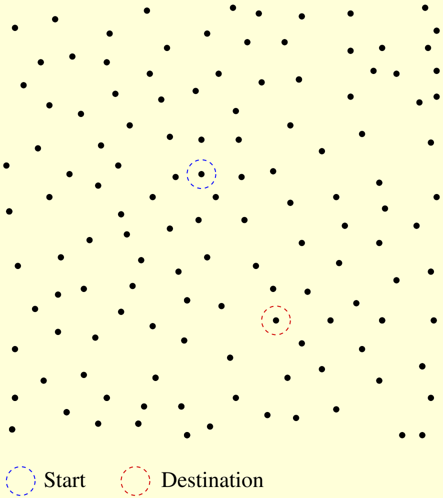
- **Key result**. If A* search is run using an admissible heuristic (and some minor technical conditions hold), then the first goal node it expands will have optimal path-cost from the start state (and the algorithm can terminate).

- Is straight line distance an admissible heuristic for navigation?

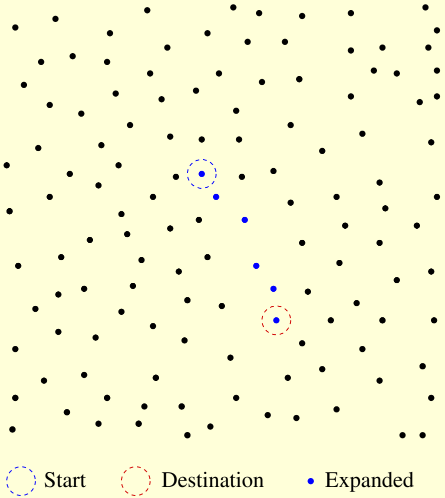
  Yes.

- For a given task, which is the best heuristic function to use?

Effect of Heuristic

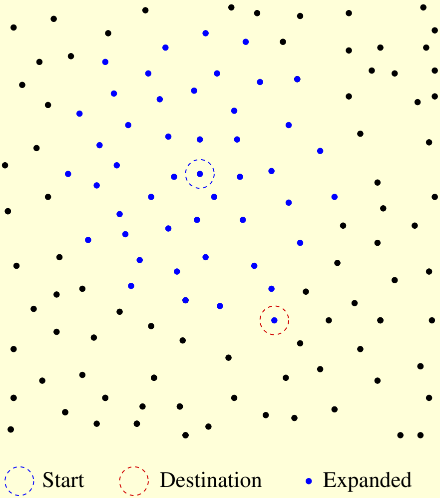


Effect of Heuristic



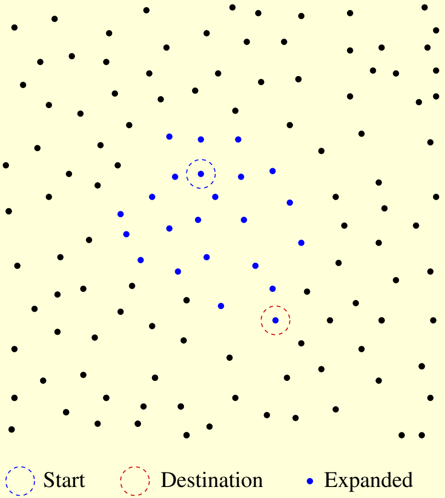
$h(n) = c^\star(n)$. Will only expand nodes along optimal path! But $c^\star(n)$ not known!

Effect of Heuristic



$h(n) = 0$. Identical to LCFS.

Effect of Heuristic



Intermediate/typical $h(n)$ expands fewer nodes than LCFS.

Questions

- How to come up with an effective admissible heuristic for a task?

Questions

- How to come up with an effective admissible heuristic for a task?

  For many tasks people have already done so. A general strategy is to solve the task with relaxed constraints.

Questions

- How to come up with an effective admissible heuristic for a task?

  For many tasks people have already done so. A general strategy is to solve the task with relaxed constraints.

- What's a good heuristic for 15-puzzle?

Start state

13	2	3	12
9	11	1	10
●	5	4	14
15	8	7	6

1	2	3	4
5	6	7	8
9	10	11	12
13	14	15	●

Goal state

Questions

- How to come up with an effective admissible heuristic for a task?

  For many tasks people have already done so. A general strategy is to solve the task with relaxed constraints.

- What's a good heuristic for 15-puzzle?

Start state

13	2	3	12
9	11	1	10
●	5	4	14
15	8	7	6

$\dashrightarrow$

1	2	3	4
5	6	7	8
9	10	11	12
13	14	15	●

Goal state

Sum of Manhattan distances between each number's position in start state and its position in goal state.

Questions

- How to come up with an effective admissible heuristic for a task?

  For many tasks people have already done so. A general strategy is to solve the task with relaxed constraints.

- What's a good heuristic for 15-puzzle?

Start state

13	2	3	12
9	11	1	10
●	5	4	14
15	8	7	6

Goal state

1	2	3	4
5	6	7	8
9	10	11	12
13	14	15	●

  Sum of Manhattan distances between each number's position in start state and its position in goal state.

- Can we make do with inadmissible heuristics?

Questions

- How to come up with an effective admissible heuristic for a task?

  For many tasks people have already done so. A general strategy is to solve the task with relaxed constraints.

- What's a good heuristic for 15-puzzle?

Start state

13	2	3	12
9	11	1	10
●	5	4	14
15	8	7	6

Goal state

1	2	3	4
5	6	7	8
9	10	11	12
13	14	15	●

  Sum of Manhattan distances between each number's position in start state and its position in goal state.

- Can we make do with inadmissible heuristics?

  Yes—example coming up in next section. But try to avoid.

Questions

- How to come up with an effective admissible heuristic for a task?

  For many tasks people have already done so. A general strategy is to solve the task with relaxed constraints.

- What's a good heuristic for 15-puzzle?

Start state

13	2	3	12
9	11	1	10
●	5	4	14
15	8	7	6

$\dashrightarrow$

Goal state

1	2	3	4
5	6	7	8
9	10	11	12
13	14	15	●

  Sum of Manhattan distances between each number's position in start state and its position in goal state.

- Can we make do with inadmissible heuristics?

  Yes—example coming up in next section. But try to avoid.

- Is A* search used widely in practice?

Questions

- How to come up with an effective admissible heuristic for a task?

  For many tasks people have already done so. A general strategy is to solve the task with relaxed constraints.

- What's a good heuristic for 15-puzzle?

Start state

13	2	3	12
9	11	1	10
•	5	4	14
15	8	7	6

$\dashrightarrow$

1	2	3	4
5	6	7	8
9	10	11	12
13	14	15	•

Goal state

  Sum of Manhattan distances between each number's position in start state and its position in goal state.

- Can we make do with inadmissible heuristics?

  Yes—example coming up in next section. But try to avoid.

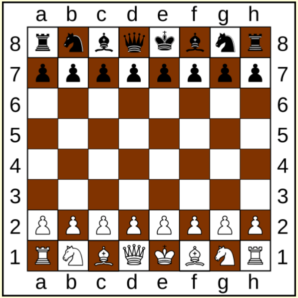
- Is $A^\star$ search used widely in practice?

  Yes. Along with variants such as $IDA^\star$.

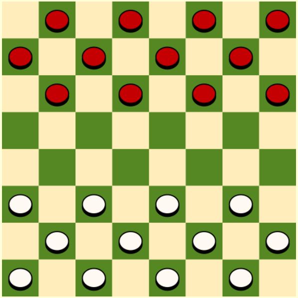
This Lecture

- Informed search (a.k.a. heuristic search)
- Application in game-playing

Search and Games



[1]

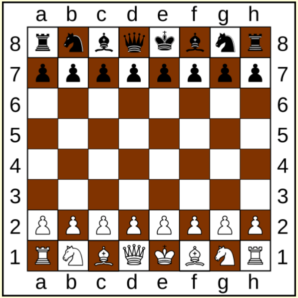


[2]

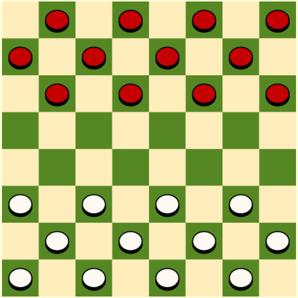
[1] https://commons.wikimedia.org/wiki/File:AAA_SVG_Chessboard_and_chess_pieces_02.svg. CC image courtesy of ILA-boy on WikiMedia Commons licensed under CC-BY-SA-3.0.

[2] https://commons.wikimedia.org/wiki/File:Draughts.svg.

Search and Games



Chess [1]

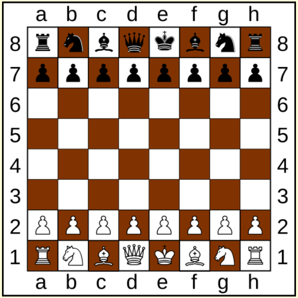


Checkers/Draughts [2]

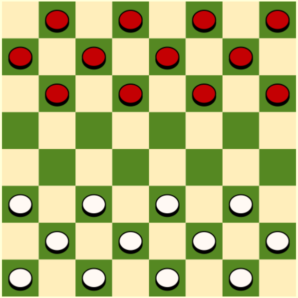
[1] https://commons.wikimedia.org/wiki/File:AAA_SVG_Chessboard_and_chess_pieces_02.svg. CC image courtesy of ILA-boy on WikiMedia Commons licensed under CC-BY-SA-3.0.

[2] https://commons.wikimedia.org/wiki/File:Draughts.svg.

Search and Games



Chess [1]



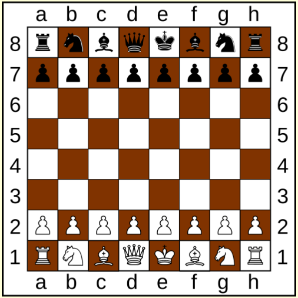
Checkers/Draughts [2]

- Can winning at chess/checkers be posed as a search problem?

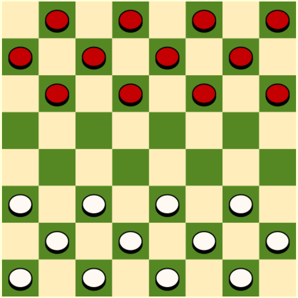
[1] https://commons.wikimedia.org/wiki/File:AAA_SVG_Chessboard_and_chess_pieces_02.svg. CC image courtesy of ILA-boy on WikiMedia Commons licensed under CC-BY-SA-3.0.

[2] https://commons.wikimedia.org/wiki/File:Draughts.svg.

Search and Games



Chess [1]



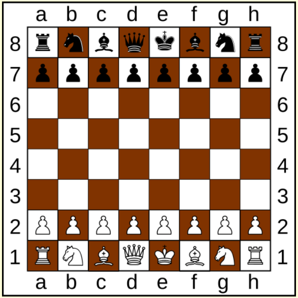
Checkers/Draughts [2]

- Can winning at chess/checkers be posed as a search problem?
- What's the main difference from our previous examples?

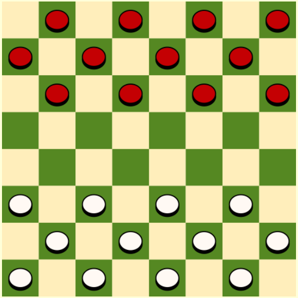
[1] `https://commons.wikimedia.org/wiki/File:AAA_SVG_Chessboard_and_chess_pieces_02.svg`. CC image courtesy of ILA-boy on WikiMedia Commons licensed under CC-BY-SA-3.0.

[2] `https://commons.wikimedia.org/wiki/File:Draughts.svg`.

Search and Games



Chess [1]



Checkers/Draughts [2]

- Can winning at chess/checkers be posed as a search problem?
- What's the main difference from our previous examples? There's another player!

[1] https://commons.wikimedia.org/wiki/File:AAA_SVG_Chessboard_and_chess_pieces_02.svg. CC image courtesy of ILA-boy on WikiMedia Commons licensed under CC-BY-SA-3.0.

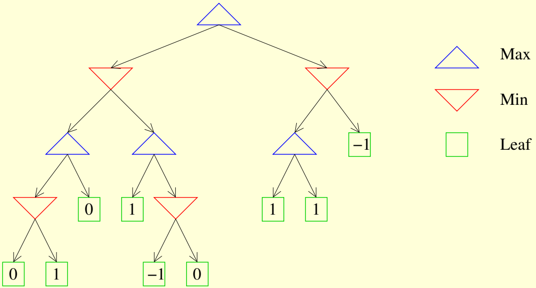
[2] https://commons.wikimedia.org/wiki/File:Draughts.svg.

Game Tree

- Assume turn-taking zero sum game with two players, Max and Min.
- Action costs usually taken as 0, but leaves have value

  -1 (Max loses), 0 (draw), 1 (Max wins).
- Value of Max node is maximum of values of children.

  Value of Min node is minimum of values of children.



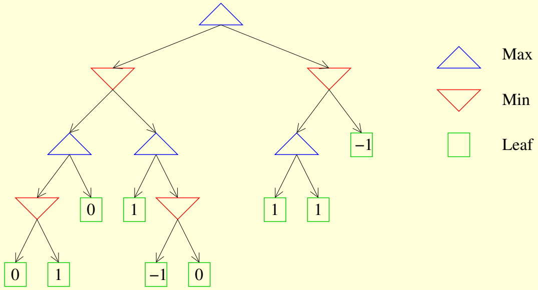
Game Tree

- Assume turn-taking zero sum game with two players, Max and Min.
- Action costs usually taken as 0, but leaves have value

$$-1 \text{ (Max loses)}, 0 \text{ (draw)}, 1 \text{ (Max wins)}.$$

- Value of Max node is maximum of values of children.

  Value of Min node is minimum of values of children.
- What is the value of the root node?

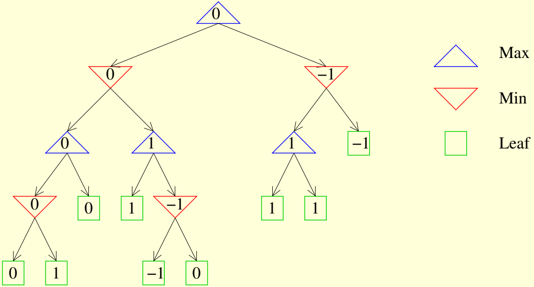


Game Tree

- Assume turn-taking zero sum game with two players, Max and Min.
- Action costs usually taken as 0, but leaves have value

  -1 (Max loses), 0 (draw), 1 (Max wins).
- Value of Max node is maximum of values of children.

  Value of Min node is minimum of values of children.
- What is the value of the root node?

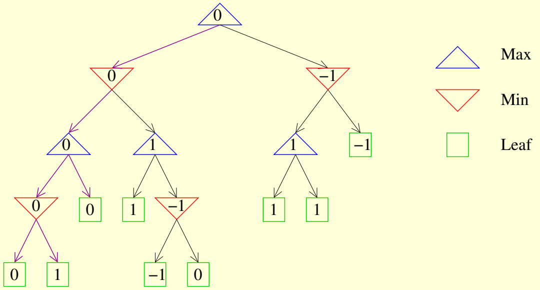


Game Tree

- Assume turn-taking zero sum game with two players, Max and Min.
- Action costs usually taken as 0, but leaves have value

  -1 (Max loses), 0 (draw), 1 (Max wins).
- Value of Max node is maximum of values of children.

  Value of Min node is minimum of values of children.
- What is the value of the root node?



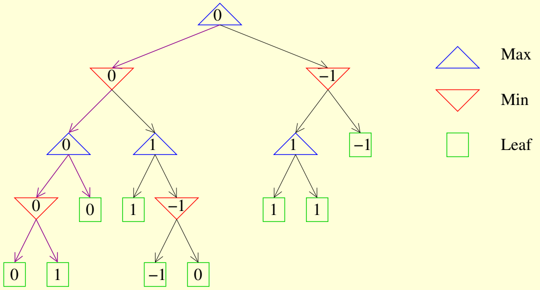
Game Tree

- Assume turn-taking zero sum game with two players, Max and Min.
- Action costs usually taken as 0, but leaves have value

$$-1 \text{ (Max loses)}, 0 \text{ (draw)}, 1 \text{ (Max wins)}.$$

- Value of Max node is maximum of values of children.

  Value of Min node is minimum of values of children.
- What is the value of the root node?



- In 2007, a massive, long-running computation concluded that the value of the root node for Checkers is 0 (draw).

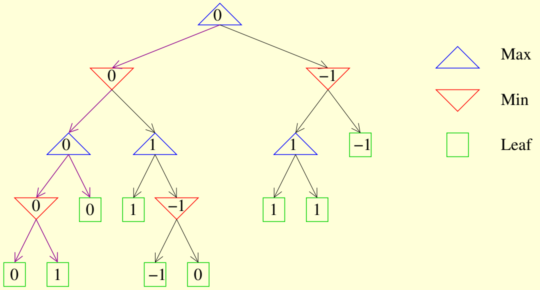
Game Tree

- Assume turn-taking zero sum game with two players, Max and Min.
- Action costs usually taken as 0, but leaves have value

$$-1 \text{ (Max loses)}, 0 \text{ (draw)}, 1 \text{ (Max wins)}.$$

- Value of Max node is maximum of values of children.

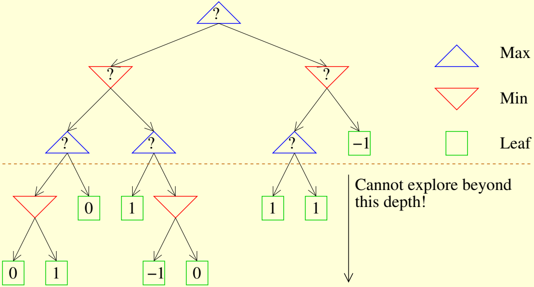
  Value of Min node is minimum of values of children.
- What is the value of the root node?



- In 2007, a massive, long-running computation concluded that the value of the root node for Checkers is 0 (draw).
- The Checkers game tree has $\approx 10^{40}$ nodes; Chess has $\approx 10^{120}$.

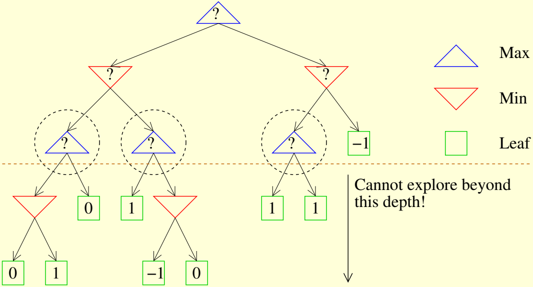
Evaluation Function

- What if game tree depth/size makes it infeasible to solve?



Evaluation Function

- What if game tree depth/size makes it infeasible to solve?



- At some depth d from current node, estimate node value using features.
- For example, in Chess, set evaluation as

$$w_1 \times \text{Material difference} + w_2 \times \text{King safety} + w_3 \times \text{pawn strength} + \dots .$$

- Weights $w_1, w_2, w_3, \dots$ are tuned or learned from experience.

References

- Section 3.6, **Artificial Intelligence: Foundations of Computational Agents**, David Poole and Alan Mackworth, Cambridge University Press, 2010. Available on-line at `https://artint.info/html/ArtInt.html`.