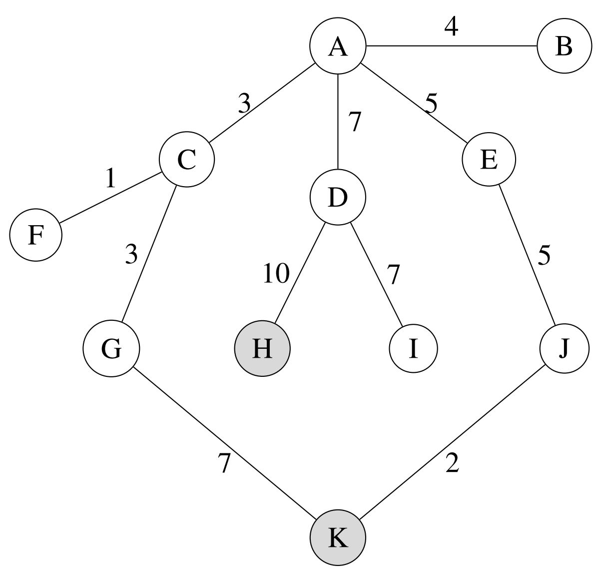
Einführung in Artificial Intelligence SS 2024, 4.0 VU, 192.027

Exercise Sheet 1 – Agents and Search I

For the discussion part of this exercise, mark and upload your solved exercises in **TUWEL** until Wednesday, June 5, 23:55 CEST. The registration for a solution discussion ends on Friday, June 7, 23:55 CEST. Be sure that you tick only those exercises that you can solve and explain!

In the discussion, students will be asked questions about their solutions of examples they checked. The discussion will be evaluated with 0–25 points, which are weighted with the fraction of checked examples and rounded to the next integer. There is *no minimum number of points* needed for a positive grade (i.e., you do not need to participate for a positive grade, but you can get at most ≈80% without doing exercises).
Note, however, that *your registration is binding*. Thus, *if* you register for a solution discussion, then it is *mandatory* to show up. Not coming to the discussion after registration will lead to a reduction of examination attempts from 4 to 2.
Please ask questions in the **TUWEL** forum or visit our tutors during the tutor hours (see **TUWEL**).

Exercise 1.1: Consider the following graph (the grey nodes are target nodes):



Determine for the following search strategies the order in which the nodes are expanded and the corresponding goal node. In case you can expand several nodes and the search strategy does not specify the order, choose the nodes in alphabetic sequence. In addition, compute for each search strategy the set of nodes that is actually kept in memory when the goal node is found (node A has depth zero).

- Breadth First Search
- Uniform-Cost Search
- Depth First Search
- Depth-Limited Search (use a limit of 2)

- Iterative Deepening Search

Exercise 1.2: Look again at the graph of Exercise 1.1. Assume that A is the start node and I is the sole goal node. Which problem do you encounter when applying *depth first search*? Discuss, how the algorithm can be changed such that the problem is solved, and explain why your modification has no impact on the properties of the DFS which were discussed in the lecture.

Exercise 1.3: Decide and explain which of the following statements are true and which are false? Back up your answers with proofs or counterexamples.

(1) If we consider an *arbitrary* search space, then there exists a graph on which neither breadth-first search nor depth-first search would be complete.

(2) Assume an agent makes completely random moves. Then, there is an environment with more than one state, in which this agent would not differ from a rational agent.

Exercise 1.4: In this exercise, we will see that some agents which have rich enough capabilities of *self-consciousness* cannot exist in principle. To this end, we define the notion of a *Gödelian agent*[1] as follows. Imagine such an agent to be a device which is able to tell us statements of a specific form. The statements the agent can tell us are build up using the following symbols:

$$\neg, T, N, (,)$$

We call the set of all statements which we can form using these symbols the *language* of our agent. For example, the statement $\neg T(T)$ is in the agent's language. The *norm* of a statement X is the statement $X(X)$. Not all statements in this language are meaningful. A *sentence* is a statement if it is of one of the following forms:

1. $T(X)$,
2. $\neg T(X)$,
3. $TN(X)$,
4. $\neg TN(X)$.

(Here, X is an arbitrary statement.) We now assign *truth values* to sentences as follows:

1. $T(X)$ is true iff X can be told by the agent;
2. $\neg T(X)$ is true iff X cannot be told by the agent;
3. $TN(X)$ is true iff the norm of X can be told by the agent;
4. $\neg TN(X)$ is true iff the norm of X cannot be told by the agent.

We assume our agent to be trustworthy, i.e., we assume that whenever the agent tells us a sentence, then it is true. Now your task is to show that, under this assumption, the opposite does not hold, i.e., prove that there is a true statement which cannot be told by our trustworthy agent. *Hint:* find a statement which is true iff *the statement itself* cannot be told by the agent. Use then the assumption of trustworthiness to conclude that your statement cannot be told. Think about why your statement cannot be told and discuss the reason(s).

[1] The self-referential character of this agent is very similar to the self-referential techniques used on the proofs of Gödel's seminal incompleteness results. Do not get scared, to solve this exercise you do not need to know anything about these.

Exercise 1.5: Let $b > 1$ be the maximal branching degree in the search tree and let d be its depth. Estimate the number of nodes, $n_{bbfs}(d)$, generated during a bidirectional breadth-first search with depth d. Show in details that $n_{bbfs}(d)$ is $O(b^{\frac{d}{2}})$ and estimate the constant c_{bbfs}.

Exercise 1.6: Describe the application types (PEAS) and the task environments of each of the following intelligent agents. Be sure to explain your reasoning and assumptions.

- Self-driving vehicle,
- An AI for the game Minesweeper,
- Automated Mars exploration rover.

1.1

Breadth first search:

Iteration: 1
Depth = 0
Visited = {A}
FIFO Queue = BCDE
(take from front)

Iteration: 2
Depth = 1
Visited = {AB}
FIFO Queue = CDE

Iteration: 3
Depth = 1
Visited = {ABC}
FIFO Queue = DEFG

Iteration: 4
Depth = 1
Visited = {ABCD}
FIFO Queue = EFGHI
STOP if goal check at generation time: H found

If goal check is at expansion time:

Iteration: 5
Depth = 1
Visited = {ABCDE}
FIFO Queue = FGHIJ

Iteration: 6
Depth = 2
Visited = {ABCDEF}
FIFO Queue = GHIJ

Iteration: 7
Depth = 2
Visited = {ABCDEFG}
FIFO Queue = HIJK

Iteration: 8
Depth = 2
Visited = {ABCDEFGH}
FIFO Queue = IJK
Goal node H reached

Uniform-Cost Search

Iteration: 1
Visited = {A}

Priority Queue =	C	B	E	D
(cost)	3	4	5	7

Iteration: 2
Visited = {AC}

Priority Queue =	B	F	E	G	D
(cost)	4	4	5	6	7

Iteration: 3
Visited = {ACB}

Priority Queue =	F	E	G	D
(cost)	4	5	6	7

Iteration: 4
Visited = {ACB}

Priority Queue =	F	E	G	D
(cost)	4	5	6	7

Iteration: 5
Visited = {ACBF}

Priority Queue =	E	G	D
(cost)	5	6	7

Iteration: 6
Visited = {ACBFE}

Priority Queue =	G	D	J
(cost)	6	7	10

Iteration: 7
Visited = {A C B F E G}

Priority Queue = D J K
(cost) 7 10 13

Iteration: 8
Visited = {A C B F E G D}

Priority Queue = J K I H
(cost) 10 13 14 17

Iteration: 9
Visited = {A C B F E G D J}

Priority Queue = K I H
(cost) 12 14 17

Iteration: 10
Visited = {A C B F E G D J K}

Priority Queue = I H
(cost) 14 17

Goal node K reached

Depth first search:

Iteration 1:
Depth = 0
Visited = {A}
Stack = E D C B
(pop from back)

Iteration 2:
Depth = 1
Visited = {A B}
Stack = E D C
(pop from back)

Iteration 3:
Depth = 1
Visited = {A B C}
Stack = E D G F
(pop from back)

Iteration 4:
Depth = 2
Visited = {A B C F}
Stack = E D G
(pop from back)

Iteration 5:
Depth = 2
Visited = {A B C F G}
Stack = E D K
(pop from back)

STOP if goal check is at generation time!
Goal node K found

If goal test at expansion time →

Iteration 6:
Depth = 3
Visited = {A B C F G K}
Stack = E D
(pop from back)

Goal node K found

Depth Limited Search (Limit = 2, goal test at expansion time)

Iteration 1:
Depth = 0
Visited = {A}
Stack = E D C B
(pop from back)

Iteration 2:
Depth = 1
Visited = {A B}
Stack = E D C
(pop from back)

Iteration 3:
Depth = 1
Visited = {A B C}
Stack = E D G F
(pop from back)

Iteration 4:
Depth = 2
Visited = {A B C F}
Stack = E D G
(pop from back)

Iteration 4:
Depth= 2
Visited = {A B C F G}
Stack = E D
(pop from back)

Iteration 5:
Depth= 1
Visited = {A B C F G D}
Stack = E I H
(pop from back)

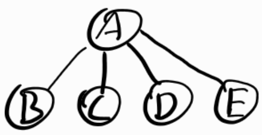
Iteration 6:
Depth=
Visited = {A B C F G D H}
Stack = E I
(pop from back)

Goal node H reached

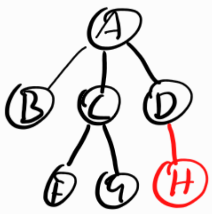
Iterative Deepening Search (goal test at expansion time)

depth-limit = 0

depth-limit = 1



depth-limit = 2



Iteration 1:
Depth=0
Visited = {A}
Stack = E D C B
(pop from back)

Iteration 2:
Depth= 1
Visited = {A B}
Stack = E D C
(pop from back)

Iteration 3:
Depth= 1
Visited = {A B C}
Stack = E D G F
(pop from back)

Iteration 4:
Depth= 2
Visited = {A B C F}
Stack = E D G
(pop from back)

Iteration 4:
Depth= 2
Visited = {A B C F G}
Stack = E D
(pop from back)

Iteration 5:
Depth= 1
Visited = {A B C F G D}
Stack = E I H
(pop from back)

Iteration 6:
Depth=
Visited = {A B C F G D H}
Stack = E I
(pop from back)

Goal node H reached

Exercise 1.2: Look again at the graph of Exercise 1.1. Assume that A is the start node and I is the sole goal node. Which problem do you encounter when applying *depth first search*? Discuss, how the algorithm can be changed such that the problem is solved, and explain why your modification has no impact on the properties of the DFS which were discussed in the lecture.

Iteration 1:
visited = {A}
Stack = EDCB
(take from back)

Iteration 2:
visited = {A B}
Stack = EDCB
(take from back)

Iteration 3:
visited = {A B C}
Stack = EDGF
(take from back)

Iteration 4:
visited = {A B C F}
Stack = EDG
(take from back)

Iteration 5:
visited = {A B C F G}
Stack = EDK
(take from back)

Iteration 6:
visited = {A B C F G K}
Stack = EDJ
(take from back)

Iteration 7
visited = {A B C F G K J}
Stack = EDE
(take from back)

Iteration 8
visited = {A B C F G K J E}
Stack = EDA
(take from back)

Problem:

Normal DFS does not check for loops and could get stuck in one. (As seen in this example)

Solution:

Only expand a node if it is not in the set of visited nodes

OR

Perform DFS multiple times with depth limit (Iterative Deepening search)

Exercise 1.3: Decide and explain which of the following statements are true and which are false? Back up your answers with proofs or counterexamples.

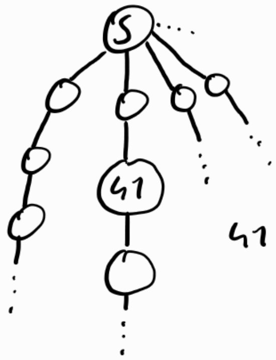
(1) If we consider an *arbitrary* search space, then there exists a graph on which neither breadth-first search nor depth-first search would be complete.

<u>Completeness</u>: An algorithm is complete if it always finds a solution, should one exist

TRUE: Consider a graph where every node has an infinite number of successors ($b = n, d = n, n \to \infty$)

DFS will explore a branch infinitely, never finding a solution

BFS will be stuck exploring depth=1, due to the infinite branching factor



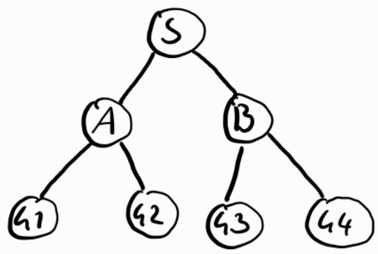
G1 is never reached

(2) Assume an agent makes completely random moves. Then, there is an environment with more than one state, in which this agent would not differ from a rational agent.

2

TRUE: We can create an environment, where all choices of the rational agent are of equal value. Thus the rational agent cannot find a "best choice" effectively becoming a random agent

Consider the following search tree. The oal of the rational Agent is to find the shortest path to any goal node G.



Here the rational agent would not differ from a random agent.

Exercise 1.4: In this exercise, we will see that some agents which have rich enough capabilities of *self-consciousness* cannot exist in principle. To this end, we define the notion of a *Gödelian agent*[1] as follows. Imagine such an agent to be a device which is able to tell us statements of a specific form. The statements the agent can tell us are build up using the following symbols:

$$\neg, T, N, (,)$$

We call the set of all statements which we can form using these symbols the *language* of our agent. For example, the statement $\neg T(T)$ is in the agent's language. The *norm* of a statement X is the statement $X(X)$. Not all statements in this language are meaningful. A *sentence* is a statement if it is of one of the following forms:

1. $T(X)$,
2. $\neg T(X)$,
3. $TN(X)$,
4. $\neg TN(X)$.

(Here, X is an arbitrary statement.) We now assign *truth values* to sentences as follows:

1. $T(X)$ is true iff X can be told by the agent;
2. $\neg T(X)$ is true iff X cannot be told by the agent;
3. $TN(X)$ is true iff the norm of X can be told by the agent;
4. $\neg TN(X)$ is true iff the norm of X cannot be told by the agent.

We assume our agent to be trustworthy, i.e., we assume that whenever the agent tells us a sentence, then it is true. Now your task is to show that, under this assumption, the opposite does not hold, i.e., prove that there is a true statement which cannot be told by our trustworthy agent. *Hint:* find a statement which is true iff *the statement itself* cannot be told by the agent. Use then the assumption of trustworthiness to conclude that your statement cannot be told. Think about why your statement cannot be told and discuss the reason(s).

$$\neg TN(\neg TN)$$

if we consider X to be true and assume:

$X = \neg TN(X)$

if X is true and the agent cannot be told the norm of X (4.) But if you are unable to tell the norm of X then X itself cannot be told by the agent.

So if X is true it cannot be told by a trustworthy agent.

And X can be true as you can substitute it with $\neg TN$ resulting in:

$$\neg TN(\neg TN)$$

As stated above the norm of X ($X = \neg TN$) cannot be told by an agent, but here the statement itself is the norm of $\neg TN$.

IF $\neg TN(\neg TN)$ cannot be told it is true, which is not posible as the agent cannot tell the norm of $\neg TN$. Meaning $\neg TN(\neg TN)$ must be false, which would make the agent untrustworthy

Exercise 1.5: Let $b > 1$ be the maximal branching degree in the search tree and let d be its depth. Estimate the number of nodes, $n_{bbfs}(d)$, generated during a bidirectional breadth-first search with depth d. Show in details that $n_{bbfs}(d)$ is $O(b^{\frac{d}{2}})$ and estimate the constant c_{bbfs}.

First we look at how many nodes regular BFS generates. (Assuming goal test is done at generation time)

For each depth-level of the search tree we generate b^i nodes, where i is the current depth-level.

$$n_{BFS} = 1 + b + b^2 + b^3 + \ldots + b^d = \sum_{i=0}^{d} b^i$$

Now we take into account that the two search trees of bidirectional meet in the middle. Meaning that both search trees must not explore maximum depth d, but rather only half of it $\frac{d}{2}$

$$n_{BFS\text{-}Start} = n_{BFS\text{-}Goal} = 1 + b + b^2 + \ldots + b^{\frac{d}{2}} = \sum_{i=0}^{\frac{d}{2}} b^i$$

$$n_{BBFS} = n_{BFS\text{-}Start} + n_{BFS\text{-}Goal} = 2 \sum_{i=0}^{\frac{d}{2}} b^i$$

Now we show that $n_{BBFS} \in O(b^{\frac{d}{2}})$ by using the following rule of big-O(s) notation

$$T(n) \in O(f(n)) \text{ if } \frac{T(n)}{f(n)} \leq c \text{ , where } c > 0$$

$$T(n) = 2 \sum_{i=0}^{\frac{d}{2}} b^i, \quad f(n) = b^{\frac{d}{2}}$$

$$T(n) = 2 \sum_{i=0}^{\frac{d}{2}} b^i = 2 \cdot \frac{b^{\frac{d}{2}+1} - 1}{b - 1}$$

Geometrische Reihe

$$\sum_{k=0}^{n} q^k = \frac{1 - q^{n+1}}{1 - q}$$

<u>Use the formula:</u>

$$\frac{T(n)}{f(n)} \leq c_{BBFS}$$

$$\frac{2 \cdot \frac{b^{\frac{d}{2}+1} - 1}{b - 1}}{b^{\frac{d}{2}}} \leq c_{BBFS}$$

$$2 \cdot \frac{1 - b^{\frac{d}{2}+1}}{1 - b} \cdot \frac{1}{b^{\frac{d}{2}}} \leq c$$

$$2 \cdot (1 - b^{\frac{d}{2}+1}) \cdot \frac{1}{b^{\frac{d}{2}}} \leq c \cdot (1-b)$$

$$\frac{1 - b^{\frac{d}{2}+1}}{b^{\frac{d}{2}}} \leq \frac{c \cdot (1-b)}{2}$$

$$\frac{1}{b^{\frac{d}{2}}} - \frac{b^{\frac{d}{2}+1}}{b^{\frac{d}{2}}} \leq \frac{c \cdot (1-b)}{2}$$

$$\frac{1}{b^{\frac{d}{2}}} - \frac{b^{\frac{d}{2}} \cdot b}{b^{\frac{d}{2}}} \leq \frac{c(1-b)}{2}$$

$$\frac{1}{b^{\frac{d}{2}}} - b \leq \frac{c(1-b)}{2}$$

$$2b^{-\frac{d}{2}} - 2b \leq c(1-b)$$

$$\frac{2b^{-\frac{d}{2}} - 2b}{1-b} \leq c$$

$$c \geq -\frac{2b - 2b^{-\frac{d}{2}}}{1-b}$$

$$\lim_{d \to \infty} -\frac{2b - 2^{-\frac{d}{2}}}{1-b} = -\frac{2b}{1-b}$$

($2^{-\frac{d}{2}} \to 0$)

$$\lim_{b \to \infty} -\frac{2b}{1-b} \Rightarrow \text{L'Hospital} \Rightarrow -\frac{2}{-1} = 2$$

($2b \to \infty$, $1-b \to -\infty$)

$$c_{BBFS} \leq 2$$

Exercise 1.6: Describe the application types (PEAS) and the task environments of each of the following intelligent agents. Be sure to explain your reasoning and assumptions.

- Self-driving vehicle,
- An AI for the game Minesweeper,
- Automated Mars exploration rover.

Performance Measure: How the success of the agent is measured

Environment: The world the agent operates in. Includes everything that could affect the agent behavior

Actuators: "Outputs" of the agent. They are used to manipulate the environment.

Sensors: The mechanism of how the agent percieves the environment.

Task environment: The "problems" to which rational agents are solutions

Self driving vehicle:

Performance:

- Time from point A -> B: Travel time should be as low as possible
- Adherene to traffic laws: laws should only be broken in extreme scenarios (avoiding a crash)
- safety: Minimize damage to vehicle and passanger
- fuel efficiency
- passenger comfort

Environment:

- Roads
- Pedestrians
- Other vehicles
- Weather
- Traffic sign (Traffic lights, street signs, ...)

Actuators:

- Steering
- Acceleration
- Braking
- Light Control
- A/C

Sensors:

- Cameras
- GPS + Navigation System
- LiDAR: For distance and speed measurements

An AI for the game Minesweeper

Performane:

- Number of actions taken to successfullyc omplete the game: This is the only Performance metric the AI shald strive for. Any other potential metrics alredy solved by minimizing this metric.

Environment:

- Grid of cells: field of integers and bombs. Only visability changes

Actuators:

- Virtual clicks on cells of the grid

Sensors:

- reading of the game borol state: check if cell is a bomb, undiscovered or how many bombs are nearby

Automated Mars exploration rover

Performane Measure:

- Quality of data collected
- energy efficiency
- Route-planning/Path navigation: It is crucial that the rover does not get stuck on terrain

Environment:

- nearly-static, partially observable terrain:
  only changes are made by the rover itself
  rover is unable to see beyond its surroundings

Actuators:

- Wheels (Accelleration, Braking)
- robotic arms
- scientific instruments for data collection

Sensors:

- Cameras
- Thermomethers
- Spectrometer
- gyroscope and accelerometers: Used to measure tilt angle