
Lecture 25

— Heuristics and AI development —
Part 1

Introduction to Software Development
Pablo Frank-Bolton

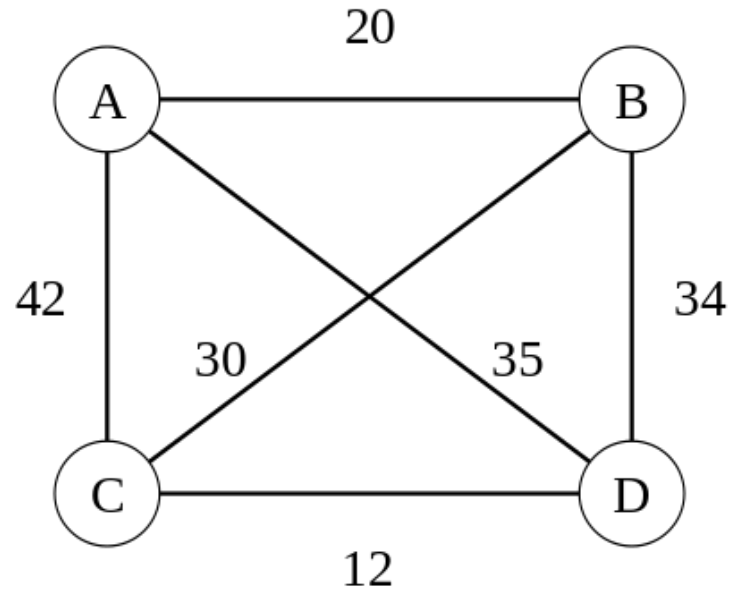
First, a word about Brute Force

Brute Force gets a bad rep. The name doesn't help.

Its nickname is better: Exhaustive Search.

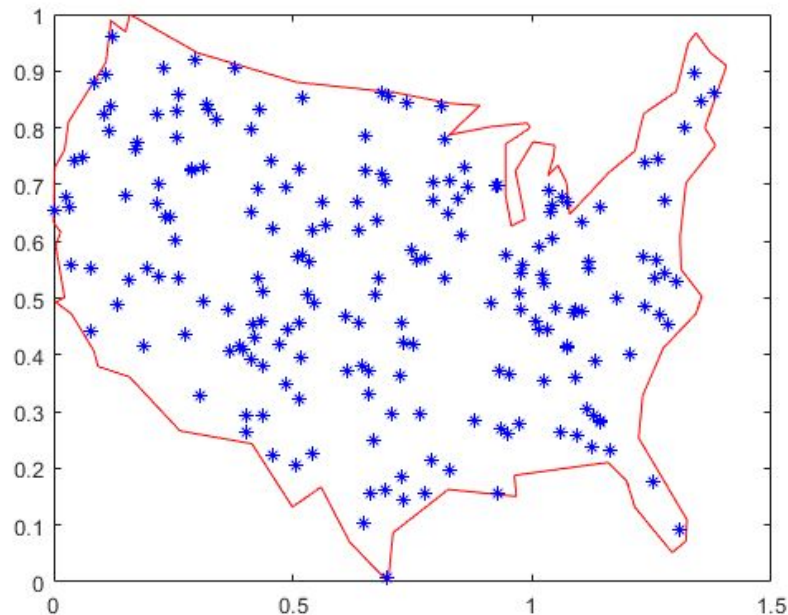
It is, by no means the worst option.

Example: Set of cities and toll roads. Find the closed route (cycle) that passes through all cities and that has the least cost.



First, a word about Brute Force

The issue is: how “exhausted” are you willing to get?



Heuristics

An alternative to Exhaustive Search

What is a Heuristic?

A Heuristic or heuristic function is a rule or set of rules by which a suitable approximation to a solution may be achieved in a reasonable amount of time.

Synonyms and subtypes of heuristics:

- **Informed Search**
- Educated Guess
- Common Sense
- Educated Risk
- Conventional Wisdom
- Rule of Thumb: this might NOT be an appropriate term (look it up)

Why use a Heuristic?

Heuristics are used when a general rule cannot be devised because of lack of information, the difficulty of considering all cases (complexity), or the fact that approximate solutions are sufficient.

After all, you don't actually compute optimum packing arrangements when going on a trip, do you?.

You just arrange stuff in what seems like a suitable manner until you pack everything or resort to pushing.

Examples of “Popular” Heuristics

Heuristics may be developed incrementally through the use of trial and error or the accumulation of years of “lore” or expertise.

- Informed Search: Which street to take if you are going from A to B and you don't know the exact route?
- Educated Guess: If you meet a shy person in a building where only math and business majors take classes, what is the major of the shy person? (sure?)
- Common Sense: Look both sides when crossing the (any) street.
- Educated Risk: If you arrive at a bus stop and there is no bus. You could walk 2 miles or wait. Do you wait? How long?
- Conventional Wisdom: On time is ± 5 minutes

How do you define a Heuristic?

Informally, your gut feeling is informed by previous experience (your own or learned).

Examples:

If you are lost, look for a known landmark and orient yourself.

Does it always work? NO!



Vs



A more “CS” definition

When searching for solutions composed of multiple options, a heuristic is a rule or function that chooses the next **reasonable** option but not necessarily the best one.

Example:



- Simple Chess AI based on a sequence of prioritized rules:

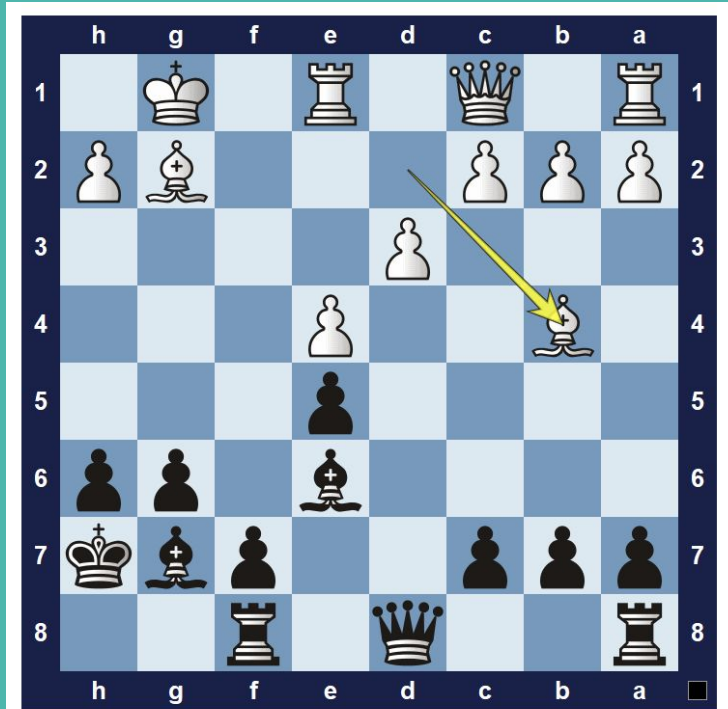
	<u>Conditions</u>
1. If you can mate, mate.	Win/Lose
2. If next move your opponent can mate you, defend against that.	Win/Lose
3. If you can take a piece of more value than your best threatened one, do so.	Attack/Defend
4. If a piece is threatened, move it to safety.	Attack/Defend
5. Move to a square that is not threatened.	Move
6. Move to a square that is defended.	Move

Using our Rules

What is the next best move?
Is it the absolute best move?

1. If you can mate, mate.
2. If next move your opponent can mate you, defend against that.
3. If you can take a piece of more value than your best threatened one, do so.
4. If a piece is threatened, move it to safety.
5. Move to a square that is not threatened.
6. Move to a square that is defended.

King	Queen	Rook	Bishop	Knight	Pawn
∞	9	5	3	3	1
					



White just moved their bishop to b4. How can you take advantage of this mistake?

Heuristics in Computer Science

Rules or functions that return values which help rank alternatives with respect to their potential contribution to the solution of the problem.

- Example 1: 0/1-Knapsack. Packing N different objects with weights: W_1, W_2, \dots, W_N and values V_1, V_2, \dots, V_N in a **limited space**.
- Objective: Maximize the total profit of packed objects.
 - Heuristic 1: Sort objects by weight and add objects from largest to smallest which still fit.
 - Heuristic 2: Sort objects by Value and add most to least valuable which still fit.
 - Heuristic 3: You come up with a better one!

Exercise 1

0/1 Knapsack

Come up with your own rule and compute the best solution.

Max Capacity: 10

W	V
3	30
7	50
4	45
8	10
2	35

Exercise 1b

0/1 Knapsack

How about this case?

Did your Heuristic win both times?

Max Capacity: 10

W	V
4	10
5	50
3	20

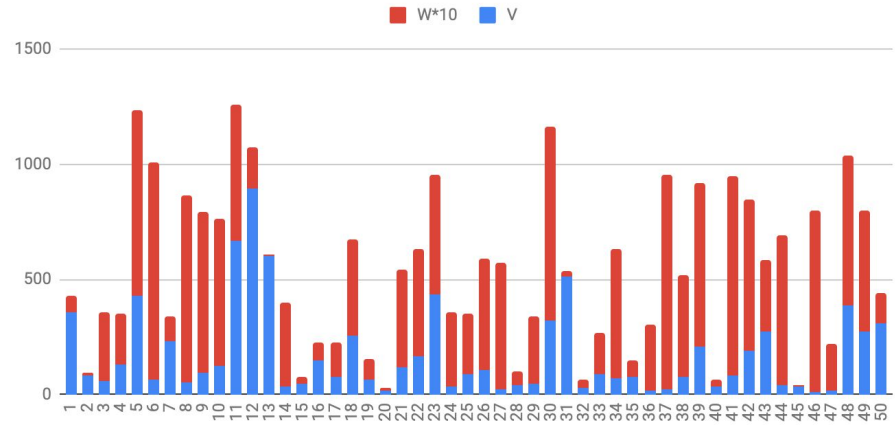
Watch out, though!

0/1 Knapsack

Is the best-ratio heuristic
optimal?

Is the optimal result **perfect?**

V and W



MaxW: 2078/7534

MinW: 5891/7534

MaxV: 7303/7534

MaxR: 7503/7534

Optimal: 7525/7534

Ad-Hoc Rules as Heuristics

One possible way of formally defining a heuristic is to have a set of ad-hoc rules for a problem.

Example: In a Maze, what rule could you follow to get out?

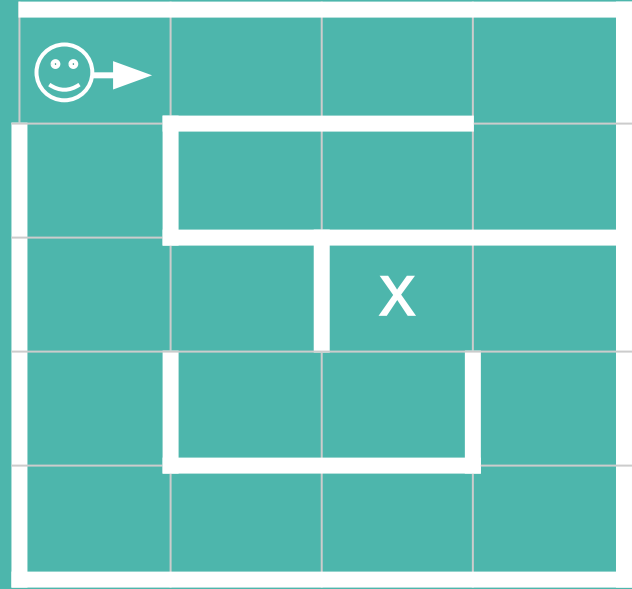
- Possible rule: place your left hand on a wall and keep going until you get out.
 - This could be called a “wall following” method, or
 - a “left-turn always method”.

While this approach might work for some mazes, it would not apply to other problems like the 0/1 knapsack problem

Exercise 2

Simple 2D Maze

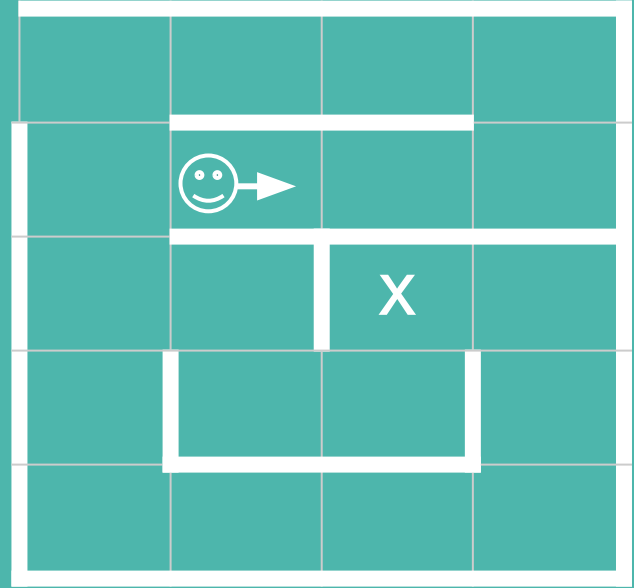
Does the “Left Turn” rule work?
The arrow shows the direction
you are facing



Exercise 2b

Simple 2D Maze

Does the “Left Turn” rule work now?



Formal Heuristics

To try to develop a more general problem solving strategy, we may think of solving a problem as the process of making a sequence of decisions from a set of candidate actions in order to reach a goal. We may define the set of candidate actions as:

$$A = \{a_1, \dots, a_N\}$$

Example: If you try to exit a 2D maze, your options at each grid point might be:

$$A: \{\text{Down, Right, Up, Left}\}$$

Formal Heuristics

Under this problem-solving approach, solving a problem is performing a sequence of “searches” for the next best candidate action.

Next best is : $a_k = \text{getNext}(A)$;

Finally, the result is the sequence: **S**: $\{s_1, \dots, s_M\}$ where each solution step s_i is one action from A. For example: S: $\{a_7, a_4, a_2, a_8\}$. Note that the solution S might have more or less actions than the total set of possible actions A.

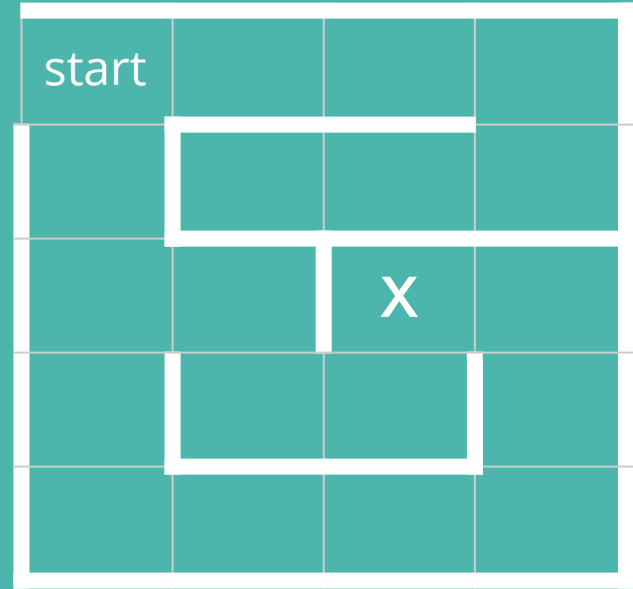
Example Maze Solution: S: {Up, Up, Right, Right, Down, Left, Left}, Or more simply: S: {U, U, R, R, D, L, L}.

We define a partial solution **Ŝ**: as a sequence that has not yet reached the goal

Exercise 3

Simple 2D Maze

Which is the correct sequence?



Formal Heuristics

In general, these decisions have an attached cost and/or an inherent value.

e.g. In the 0/1 Knapsack problem, a_k is really a combination of $\{v_k, w_k\}$

One usually wants to either **minimize the cost** or **maximize the value**. This depends how you phrase the problem. We'll use the cost version but the following applies also to maximizing value.

Formally, we can use a function $f(\hat{S})$ to estimate the cost of going from the current state of the partial solution \hat{S} to the **final state** or **goal**.

Ideal $f(i)$:

$f(i)$: Actual minimum distance
to the goal x

$f(i)$ is given by an oracle

6 start	7	8	9
5	12	11	10
4	3	x	1
5	2	1	2
6	5	4	3

- currently at $[2,0]$ (the circle);
- current partial solution: $\hat{S}:\{D,D\}$;
- $f(\hat{S})$ estimate to reach x is 4.
- $f(\hat{S} \cup D) = 5$
- $f(\hat{S} \cup R) = 3$

Formal Heuristics

The problem arises when we don't have enough information to compute the ideal $f(\hat{S})$ for each partial solution \hat{S} .

In these cases, we need to come up with one or many rules to make a decision.

A typical policy is called: the **Greedy Method**. The idea is:

1. Design a decent approximation of $f(\hat{S})$
2. Sort the set of allowable next actions according to $f(\hat{S})$
3. Pick the best one (minimum cost or maximum value).

Possible Estimate

$f(i)$:

$f(i)$: Manhattan distance to x

$f(i)$ is like the hint in the hot/cold searching game

4	3	2	3
3	2	1	2
2	1	x	1
3	2	1	2
4	3	2	3

- currently at $[2,0]$ (the circle);
- current partial solution: $\hat{S}:\{D,D\}$;
- $f(\hat{S})$ estimate to reach x is 2.
- $f(\hat{S} \cup D) = 3$
- $f(\hat{S} \cup R) = 1 \leftarrow$ Next Greedy Choice

Problems with the Greedy Method

The greedy method is an excellent option for beginners.

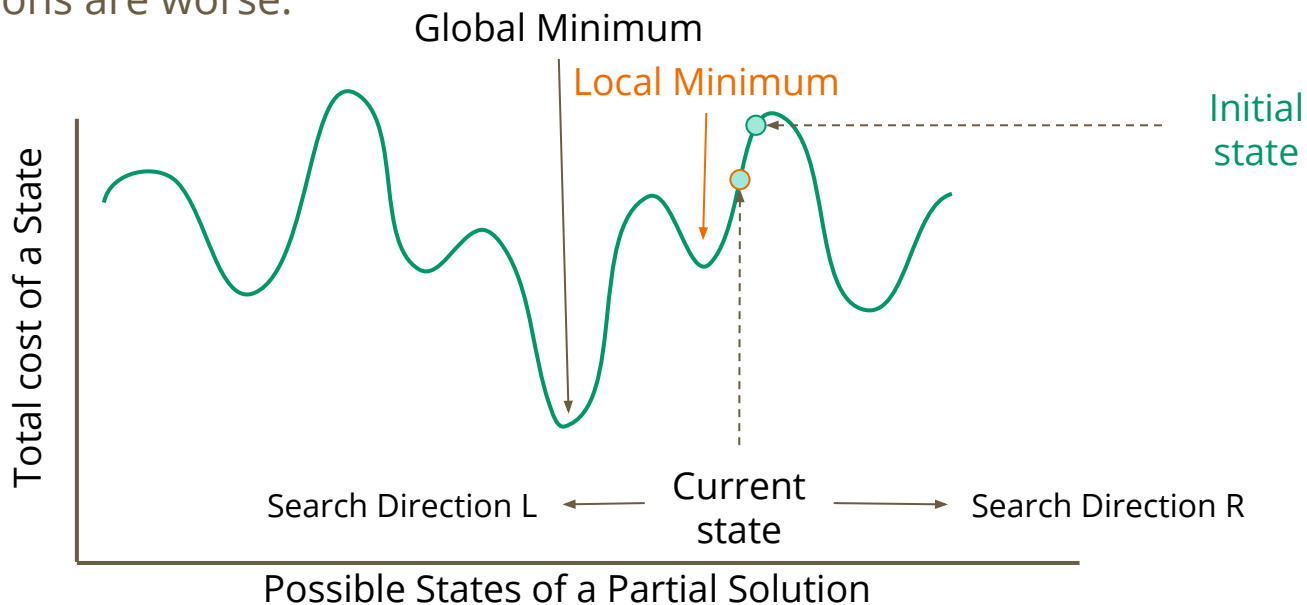
However, it has a weakness: The sequence of “top picks” can arrive at a partial solution that looks good if you don’t look too hard, but is actually not good at all. We call these points:

Local minima: a state with lower cost than all of its close neighbors but not all possible states.

Local maxima: a state with greater value than all of its close neighbors but not all possible states.

Example: Minimization Problem In 1 Dimension

If we start at the **initial state** and build \hat{S} , the next decisions from $A:\{L, R\}$ will seek to minimize \hat{S} until they reach the **Local Minimum**, where \hat{S} will get “trapped” since all following options are worse.



Greedy Method, Minimum $f(i)$:

$f(i)$: Manhattan distance to x

For this problem instance, the greedy choices lead to a bad state.

4	3	2	3
3	2	1	2
2	1	x	1
3	2	1	2
4	3	2	3

- currently at $[2,0]$ (the circle);
- current partial solution: $\hat{S}:\{D,D\}$;
- $f(\hat{S})$ estimate to reach x is 2.
- $f(\hat{S} \cup D) = 3$
- $f(\hat{S} \cup R) = 1 \leftarrow$ Next Greedy Choice

One Possible Solution: Escape!

The Greedy method is sometimes complemented with additional heuristics to escape the local minima.

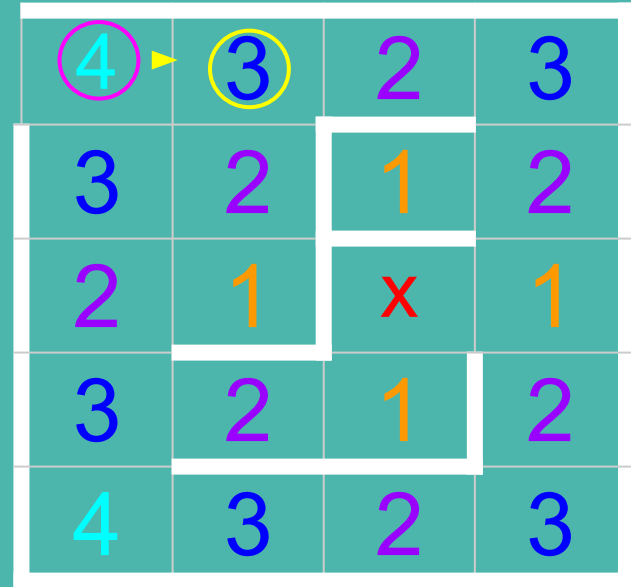
One such option is to **keep track of intermediate states** where you “could have chosen another path of similar cost”.

Another is to **tag states that lead to local minima** in order to avoid them when searching an alternative path.

Greedy Method, Minimum $f(i)$:

$f(i)$: Manhattan distance to x

In the path to x , we might keep track of possible “alternatives”.

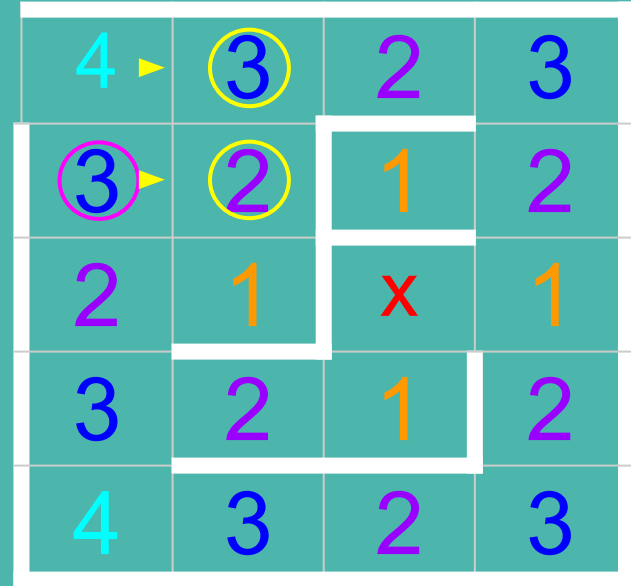


- currently at $[0,0]$ (the circle);
- partial solution: $\hat{S}:\{\emptyset\}$;
- $f(\hat{S})$ estimate to reach x is 4.
- $f(\hat{S} \cup D) = 3 \leftarrow$ Next Greedy Choice
- $f(\hat{S} \cup R) = 3 \leftarrow$ Tag Alternative

Greedy Method, Minimum $f(i)$:

$f(i)$: Manhattan distance to x

In the path to x , we might keep track of possible “alternatives”.

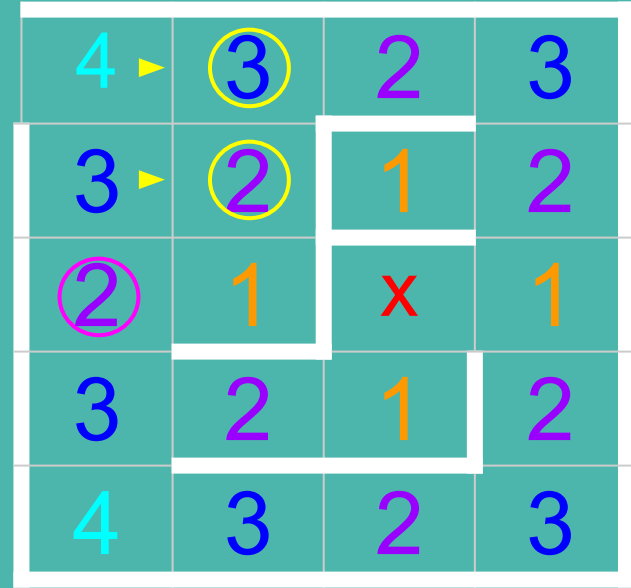


- currently at $[1,0]$ (the circle);
- partial solution: $\hat{S}:\{D\}$;
- $f(\hat{S})$ estimate to reach x is **3**.
- $f(\hat{S} \cup D) = 2 \leftarrow$ **Next Greedy Choice**
- $f(\hat{S} \cup R) = 2 \leftarrow$ **Tag Alternative**

Greedy Method, Minimum $f(i)$:

$f(i)$: Manhattan distance to x

In the path to x , we might keep track of possible “alternatives”.

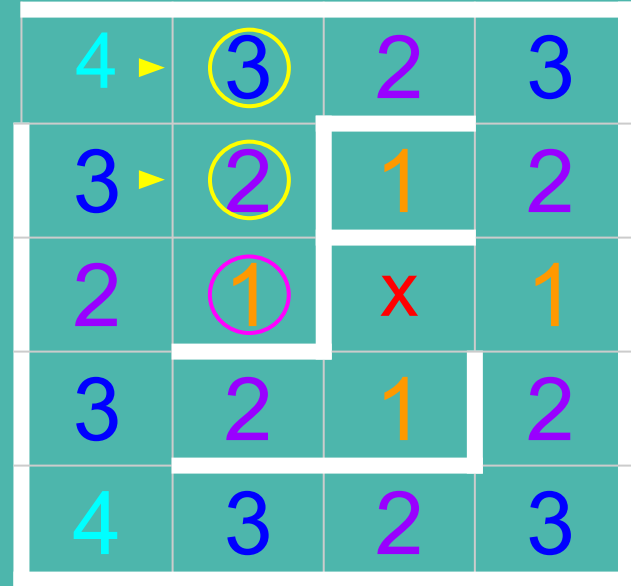


- currently at $[2,0]$ (the circle);
- partial solution: $\hat{S}:\{D,D\}$;
- $f(\hat{S})$ estimate to reach x is 2.
- $f(\hat{S} \cup D) = 3$
- $f(\hat{S} \cup R) = 1 \leftarrow$ Next Greedy Choice

Greedy Method, Minimum $f(i)$:

$f(i)$: Manhattan distance to x

In the path to x , we might keep track of possible “alternatives”.



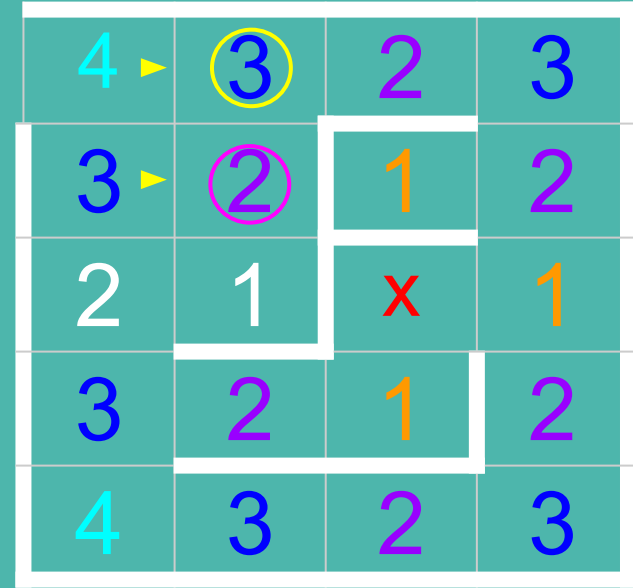
- currently at $[2,1]$ (the circle);
- partial solution: $\hat{S}:\{D,D,R\}$;
- $f(\hat{S})$ estimate to reach x is **1**.
- $f(\hat{S} \cup U) = 2$ ← worse!
- $f(\hat{S} \cup L) = 2$ ← worse!

Greedy Method, Minimum $f(i)$:

$f(i)$: Manhattan distance to x

In the path to x , we might keep track of possible “alternatives”.

There are many little details to decide like here: Backtrack or Advance

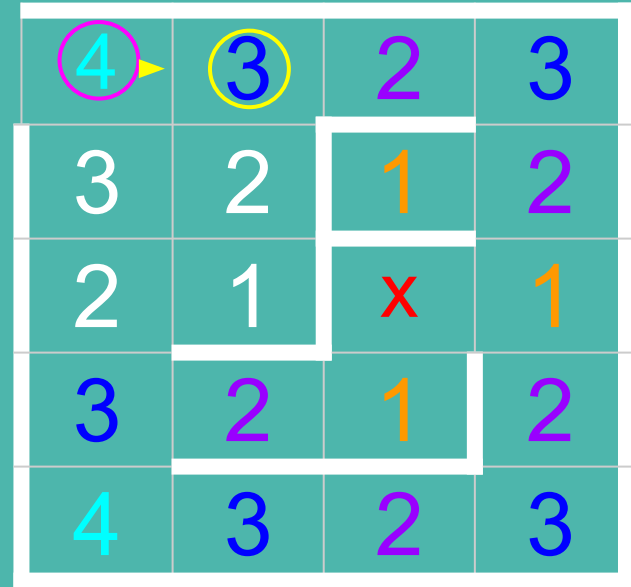


- currently at $[1,1]$ (the circle);
- partial solution: $\hat{S}:\{D,R\}$;
- $f(\hat{S})$ estimate to reach x is 2.
- $f(\hat{S} \cup D) = 1$ ← Tagged as Bad
- $f(\hat{S} \cup U) = 3$ ← Seen Before ***

Greedy Method, Minimum $f(i)$:

$f(i)$: Manhattan distance to x

In the path to x , we might keep track of possible “alternatives”.



- Return to $[0,0]$ (the circle);
- Use previous $\hat{S}:\{D,R\}$; to:
- update partial solution: $\hat{S}:\{\emptyset\}$;
- $f(\hat{S})$ estimate to reach x is 4.
- $f(\hat{S} \cup R) = 3 \leftarrow$ Next Greedy Choice

Greedy Method, Minimum $f(i)$:

$f(i)$: Manhattan distance to x

In the path to x , we might keep track of possible “alternatives”.

4	3	2	3
3	2	1	2
2	1	x	1
3	2	1	2
4	3	2	3

- currently at $[0,1]$ (the circle);
- partial solution: $\hat{S}:\{R\}$;
- $f(\hat{S})$ estimate to reach x is **3**.
- $f(\hat{S} \cup D) = 2 \leftarrow$ Tagged as Bad
- $f(\hat{S} \cup R) = 2 \leftarrow$ Next Greedy Choice

Greedy Method, Minimum $f(i)$:

$f(i)$: Manhattan distance to x

In the path to x , we might keep track of possible “alternatives”.

4	3	②	3
3	2	1	2
2	1	x	1
3	2	1	2
4	3	2	3

- currently at $[0,2]$ (the circle);
- partial solution: $\hat{S}:\{R,R\}$;
- $f(\hat{S})$ estimate to reach x is 2.
- $f(\hat{S} \cup R) = 3 \leftarrow$ Next Greedy Choice

Greedy Method, Minimum $f(i)$:

$f(i)$: Manhattan distance to x

In the path to x , we might keep track of possible “alternatives”.

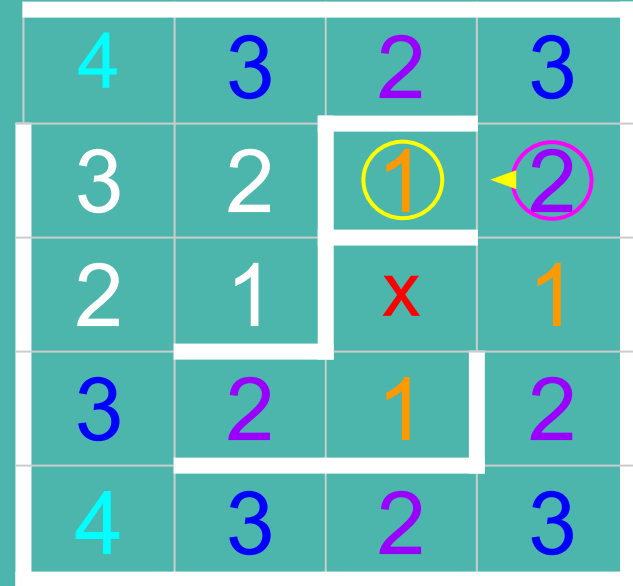
4	3	2	3
3	2	1	2
2	1	x	1
3	2	1	2
4	3	2	3

- currently at $[0,3]$ (the circle);
- partial solution: $\hat{S}:\{R,R,R\}$;
- $f(\hat{S})$ estimate to reach x is **3**.
- $f(\hat{S} \cup D) = 2 \leftarrow$ **Next Greedy Choice**

Greedy Method, Minimum $f(i)$:

$f(i)$: Manhattan distance to x

In the path to x , we might keep track of possible “alternatives”.

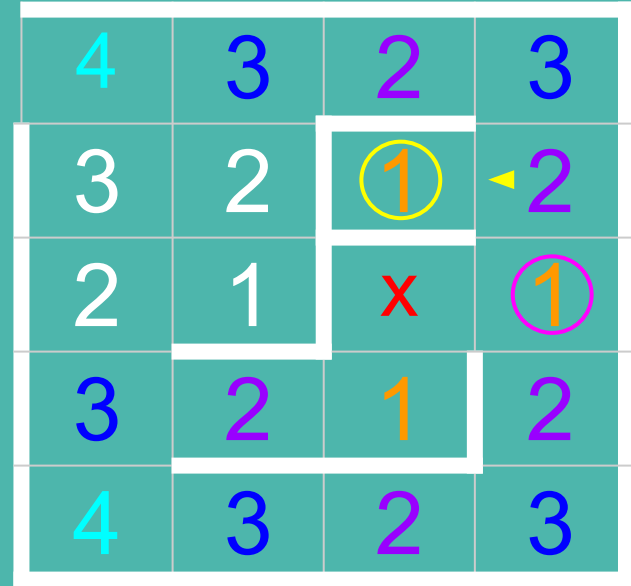


- currently at [1,3] (the circle);
- partial solution: $\hat{S}:\{R,R,R,D\}$;
- $f(\hat{S})$ estimate to reach x is 2.
- $f(\hat{S} \cup D) = 1 \leftarrow$ Next Greedy Choice
- $f(\hat{S} \cup L) = 1 \leftarrow$ Tag Alternative

Greedy Method, Minimum $f(i)$:

$f(i)$: Manhattan distance to x

In the path to x , we might keep track of possible “alternatives”.



- currently at $[2,3]$ (the circle);
- partial solution: $\hat{S}:\{R,R,R,D,D\}$;
- $f(\hat{S})$ estimate to reach x is 1.
- $f(\hat{S} \cup D) = 2$
- $f(\hat{S} \cup L) = 1 \leftarrow$ Next Greedy Choice

Greedy Method, Minimum $f(i)$:

$f(i)$: Manhattan distance to x

In the path to x , we might keep track of possible “alternatives”.

4	3	2	3
3	2	①	2
2	1	⊗	1
3	2	1	2
4	3	2	3

- currently at $[2,3]$ (the circle);
- partial solution: $\hat{S}:\{R,R,R,D,D,L\}$;
- $f(S)$ to reach x is **0**.

Another example of Greedy pitfall

You run the “Vectors and Algorithms store”, where you sell vectors for 7 cents and Algorithms for 50 cents.

A customer buys 1 Algorithm and 2 vectors (**64 cents**) and gives you **1 dollar**.

You have many coins of the following denominations: **A:{0.25, 0.10, 0.5, 0.01}**

How can you give change in order to minimize the number of coins you give the customer?

What if you live in OddLand where their coins are: **A:{0.25, 0.15, 0.01}**

Types of Games

So far, you've seen games where we

- try to maximize or minimize an approximate value
 - 0/1 Knapsack
- try to reach a specific state by the best possible path
 - Maze
- try to beat an opponent by reaching a final state where we win
 - Chess

What changes in the formulations?

The definition of an intermediate game **state**, the heuristic **function**, and the **goal** or stopping conditions.

End of Part 1

Let's Look at the Next Lab and HW5