Chapter 3 - Part 2

Informed (Heuristic) Search

*Best-First Search, Greedy, and an Introduction to A**

Letting the agent see further than the next step

COMP338 - Artificial Intelligence
Birzeit University | Second Semester 2025/2026

Where we left off - uninformed search

In previous sessions we learnt about a search agent that knows nothing about the problem beyond the problem definition itself: the **initial state**, **the actions**, the **result** of each action, the **goal test**, and the **step cost**.

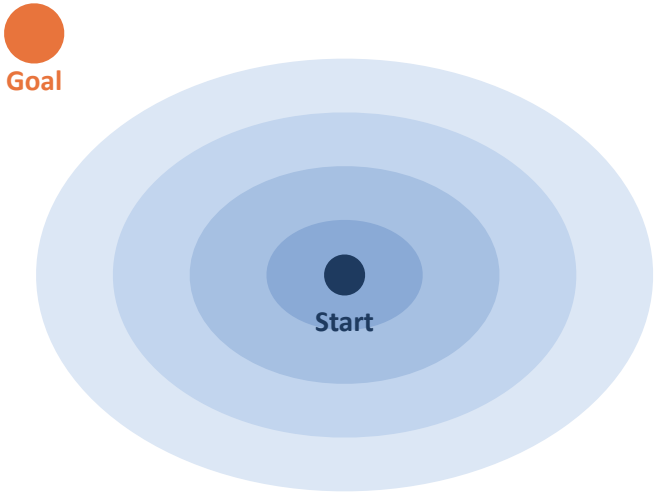
BFS	UCS	DFS	IDS
Breadth-First	Uniform-Cost	Depth-First	Iterative Deepening
orders frontier	*orders frontier*	*orders frontier*	*orders frontier*
by depth	**by g(n)**	**deepest first**	**depth-limited DFS**

The common thread

Every uninformed strategy decides what to expand next using only structural information - depth or accumulated cost. None of them looks at where the goal actually is.

The wall: uninformed search expands too much

Imagine you are in Hebron and want to drive to Ramallah. UCS will expand cities in every direction - even those obviously moving away from Bucharest - until cumulative cost forces it elsewhere.



UCS expansion (no goal awareness)

The question

If we already know where Bucharest is on the map, **why doesn't the agent use that knowledge?**

Humans never search in all directions equally - we look toward the destination first.

Quick test: making change

Pause and try this on paper before I move on

The problem

You are a cashier. A customer is owed **67 shekels** in change.

Your till has coins of:
1, 5, 10, 20, 50.

Hand over the ***smallest possible number*** of coins.

Quick test: making change

Pause and try this on paper before I move on

The problem

You are a cashier. A customer is owed **67 shekels** in change.

Your till has coins of:
1, 5, 10, 20, 50.

Hand over the ***smallest possible number*** of coins.

A natural strategy

At each step, take the **largest coin** that does not exceed the remaining amount.

Coins given: 50, 10, 5, 1, 1.

Five coins, total = 67. No backtracking, no exploring alternatives.

When the simple rule breaks

Now change the denominations. Suppose the till has only coins of 1, 6, 10. The customer needs 12 shekels.

Greedy says

Take 10 → remaining 2
Take 1 → remaining 1
Take 1 → done

3 coins: 10 + 1 + 1

Truly optimal

Take 6 → remaining 6
Take 6 → done

2 coins: 6 + 6

A heuristic guides search well but does not guarantee the best solution. We will need A* to fix this (we'll visit later.)

The main idea of informed search

Use problem-specific knowledge to choose what to expand next.

Uninformed

What does the agent know?
Only the problem definition itself.

What guides the search?
Structural cues: depth, accumulated cost.

Examples
BFS, UCS, DFS, IDS.

Informed (heuristic)

What does the agent know?
Problem definition plus extra hints - a map, a distance estimate, domain knowledge.

What guides the search?
An evaluation function f(n) ranks frontier nodes by desirability.

Examples
Greedy best-first, A*, RBFS, SMA*.

Reference: AIMA 4th ed., Russell & Norvig (2021), Section 3.5.

The heuristic function h(n)

h(n) estimated cost of the cheapest path from node n to a goal state.
Computed from the state at n only - never from the path that led there.

Problem-specific
Different problems need different heuristics

Non-negative
h(n) ≥ 0 for all n

h(goal) = 0
Zero estimate at any goal state

The classic example: straight-line distance

Romania route-finding: $h_{SLD}(n)$ = straight-line distance from city n to Bucharest. It cannot be the wrong direction (the road is at least as long as the line), so it is a safe lower bound.

Examples (km): h(Arad) = 366, h(Sibiu) = 253, h(Pitesti) = 100, h(Bucharest) = 0.

Best-first search: a general framework

Best-first search is not a single algorithm - it is a family. You pick an evaluation function f(n), and the algorithm always expands the frontier node with the **smallest f**.

The mechanism

1. Initialize the frontier with the start node.
2. Pick the node with the lowest f from the frontier.
3. If it is a goal, return the path.
4. Otherwise, expand it and add its children.
5. Repeat.

Choice of f gives the family

UCS

$f(n) = g(n)$ *(g: only past cost matters)*

Greedy best-first.

$f(n) = h(n)$ *(h: only the future estimate matters)*

A*

$f(n) = g(n) + h(n)$

UCS is technically a special case of best-first - informed methods just choose richer f.

Greedy best-first search

"Always expand the node that looks closest to the goal."

Evaluation function:

$$f(n) = h(n)$$

Cost so far is ignored.

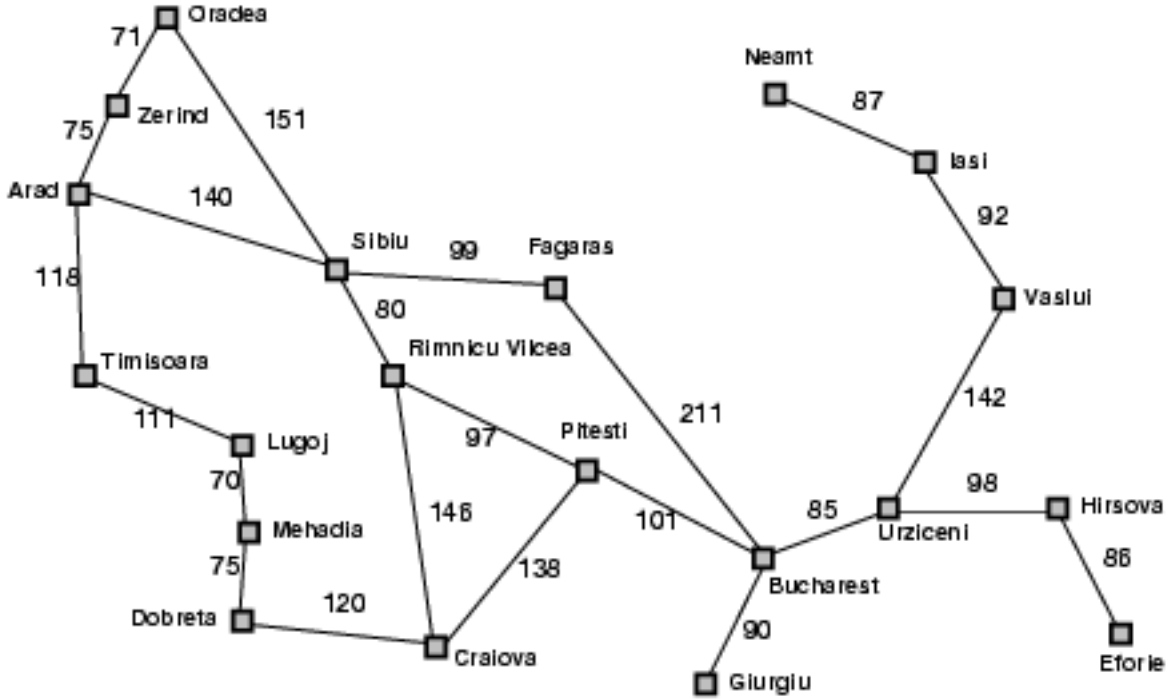
Strength

Often very fast. With a sensible heuristic, it dives straight toward the goal and ignores most of the search space.

Weakness

It trusts the estimate completely. A misleading h can send it down a long path, and it never reconsiders.

Greedy best-first search



Straight–line distance to Bucharest	
Arad	366
Bucharest	0
Craiova	160
Dobreta	242
Eforie	161
Fagaras	176
Giurgiu	77
Hirsova	151
Iasi	226
Lugoj	244
Mehadia	241
Neamt	234
Oradea	380
Pitesti	10
Rimnicu Vilcea	193
Sibiu	253
Timisoara	329
Urziceni	80
Vaslui	199
Zerind	374

Greedy in action: Arad to Bucharest

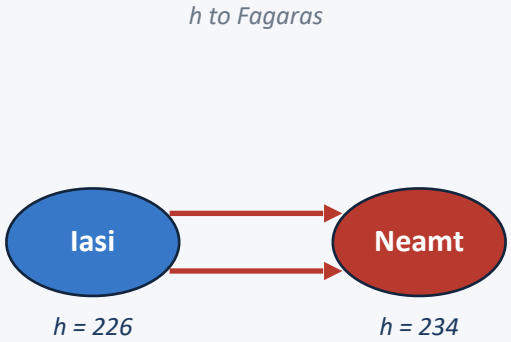
Heuristic: straight-line distance to Bucharest. Frontier always picks the smallest h.

Step	Node expanded	h to Bucharest	Frontier picks next
1	**Arad**	366	Sibiu (253), Timisoara (329), Zerind (374)
2	**Sibiu**	253	Fagaras (176), Rimnicu (193), Arad (366)…
3	**Fagaras**	176	Sibiu (253), Bucharest (0)
4	**Bucharest**	**0**	**Goal - return path**

Path returned: Arad → Sibiu → Fagaras → Bucharest. Total cost = 450 km.
Note that the truly optimal path is Arad → Sibiu → Rimnicu Vilcea → Pitesti → Bucharest, costing 418 km.

When greedy goes wrong: the loop trap

Greedy can do worse than miss the optimum: it can fail to terminate. The classic case on the Romania map is going from Iasi toward Fagaras through Neamt.



Neamt is a dead end: its only neighbour is Iasi.

The infinite loop

Suppose tree-search greedy is at Neamt. Its only neighbour is Iasi (h=226), which looks better.

From Iasi without an explored-set, Neamt looks attractive again relative to backtracking.

Result: ***Iasi to Neamt to Iasi to Neamt to ...*** indefinitely.

Greedy best-first: Complete? **No in tree-search** Optimal? **No** Time / Space **O(b^m) worst case**

A* search

Hart, Nilsson and Raphael, 1968 - the most influential search algorithm in AI

$$f(n) = g(n) + h(n)$$

known cost so far + estimated cost remaining

The single change: add g back in. That tiny edit gives optimality, completeness, and modern AI's most-used search algorithm.

Why A* fixes the greedy trap

Recall: greedy went Arad → Sibiu → Fagaras → Bucharest (450 km), missing the better route through Pitesti (418 km). Let us see what A* does at the critical decision point - Sibiu.

At Sibiu, A compares two children using f(n) = g(n) + h(n):*

	Path so far	g(n)	h(n)	f(n) = g + h
Choose Fagaras	Arad → Sibiu → Fagaras	140+99=239	176	**415**
Choose Rimnicu Vilcea	Arad → Sibiu → Rimnicu	140+80=220	193	**413**

→ A* picks Rimnicu (lower f). That is the route to the actual optimum.

Greedy was tricked because Fagaras looks closer (h=176) than Rimnicu (h=193). *A* is not tricked because the cost so far also matters: Rimnicu is much cheaper to reach.*

Three algorithms, one priority queue

UCS $f = g(n)$	Greedy best-first $f = h(n)$	A* $f = g(n) + h(n)$
Strategy	**Strategy**	**Strategy**
Cheapest known	Looks closest	Best total estimate
Uses heuristic?	**Uses heuristic?**	**Uses heuristic?**
No	Only h	Yes (with g)
Complete?	**Complete?**	**Complete?**
Yes	No (tree-search)	Yes
Optimal?	**Optimal?**	**Optimal?**
Yes	No	Yes (**admissible** h)
Speed	**Speed**	**Speed**
Slow - explores in all directions	Often very fast, but risky	Fast and reliable

All three are best-first searches. The only difference is how the priority is computed.

What we covered, what comes next

Today

- Why uninformed search is not enough.
- The heuristic function h(n) - estimated cost to a goal.
- Best-first search as a general framework.
- Greedy best-first: f = h. Fast but not optimal, and it can loop.
- A* introduction: f = g + h. The fix to the greedy trap.

Next session

- Full A* trace on the Romania map.
- Admissibility and consistency - when is A* optimal?
- Designing heuristics: 8-puzzle case study.
- Dominance and the max combination rule.
- Memory-bounded variants: RBFS and SMA*.

Reading: AIMA 4th ed. (Russell & Norvig, 2021), Chapter 3 - Section 3.5 (Informed Search Strategies).

References

Primary textbook

Russell, S., & Norvig, P. (2021). *Artificial Intelligence: A Modern Approach* (4th ed., Global Edition). Pearson. Chapter 3 - Solving Problems by Searching, Section 3.5.

Original A* paper

Hart, P. E., Nilsson, N. J., & Raphael, B. (1968). A formal basis for the heuristic determination of minimum cost paths. *IEEE Transactions on Systems Science and Cybernetics*, 4(2), 100–107.

Course materials

Jarrar, M. (2018). Lecture Notes on Heuristic Informed Search Algorithms. Birzeit University.

https://www.jarrar.info/courses/AI/Jarrar.LectureNotes.Ch3.InformedSearch.pdf

Further reading on heuristic functions

Pearl, J. (1984). *Heuristics: Intelligent Search Strategies for Computer Problem Solving*. Addison-Wesley. (Classic reference for heuristic design.)