**AI – Heuristic and informed search – 04**

***Heuristic* h(n)**

A heuristic is a function h(n) is a guess to produce a feasible solution in a reasonable time span

**h(n) = estimated cost of the cheapest path from the state at node n to a goal state**

This means that we are trading off on solution optimality (e.g. the shortest path), but we rather gain on time, and **a** solution is guaranteed to be found

***Best-first search (example of informed search)***

Best-first search refers to a **class** of search algorithms, in which the most promising (according to predefined rules) is chosen to be explored first, usually implying the usage of **distance as heuristic** via setting a **priority queue**

To search the graph space, the BFS\* method uses two lists for tracking the traversal:

* + An ‘Open’ list that keeps track of the current ‘immediate’ nodes available for traversal
  + A ‘Closed’ list that keeps track of the nodes already traversed.

A natural question would be “**how does this differ from Dijkstra’s algorithm?”**

BFS\* fails on weighted graphs, as it will choose **distance** over **cost of edge traversal**

***Greedy best-first search***

Greedy BFS\* will always expand to the most promising node (i.e. nearest), as said before, the example in the slides uses **straight line distance** (which is, trivially, **less** than the actual distance)as heuristic for the example, and it certainly finds a solution, just not the best

The usage of backtracking is considered, but Greedy BFS\* focuses on exploring the most promising path **without considering alternative paths**

It can lead to dead ends and/or be suboptimal, and will need backtracking if we want more out of it; The heuristic is:

**f(n) = h(n)**

***A\* algorithm***

**Optimal for positive costs**

A\* algo comes from the fact that heuristic is often misleading, and operates by having the **evaluation function** (aka fitness number) to minimize:

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

where:

* n is the next node on the path
* g(n) is the cost of the path from the start node to n
* h(n) is a heuristic that estimates the cost of the cheapest node from n to the goal

***Admissibility and consistency of heuristic*** (see Manhattan distance)

Having talked about A\*, we introduce two concepts that make a heuristic “good”:

* **Admissibility**

h(n) **never** overestimates the cost of reaching the goal, so it’s not higher than the lowest possible one (this property makes A\* optimal)

* **Consistency**

Graphical user interface, text, application

Description automatically generatedh(n) is **always** less than or equal to the estimated distance from any neighbour vertex to the goal, plus the cost to reach that neighbour

**A consistent heuristic is always admissible, while the converse is not true**

When talking about consistency, we focus solely on the heuristic function

Proof : **Consistency → Admissibility**