Local and Global Heuristic search Ngwepe FM

**Local Heuristic Search General Idea:**

* Start with a random or initial state.
* Evaluate the current state using a heuristic.
* Move to a neighbour state that has a **better heuristic value**.
* Repeat until no neighbour is better or when a **local maximum** is reached.

Pseudocode:

FUNCTION LocalHeuristicSearch(problem):

current\_state ← problem.initial\_state

LOOP:

neighbors ← GetNeighbors(current\_state)

IF neighbors is empty:

RETURN current\_state // No way to move, return current solution

next\_state ← neighbor in neighbors with highest heuristic value

IF heuristic(next\_state) ≤ heuristic(current\_state):

RETURN current\_state // No better neighbor, local maximum

current\_state ← next\_state

## Limitations:

* **Can get stuck** at **local maxima/minima**.
* **Doesn’t backtrack** — doesn't remember where it came from.
* **No global guarantee** of best solution.

**Global Heuristic Search General Idea:**

**Global heuristic search** algorithms explore the **entire or large parts of the state space**, using heuristics to guide the search toward the **goal state**, while avoiding local traps. It **remembers paths, evaluates multiple paths**, and **considers the global best** using both the cost so far and the estimated cost to goal.

## Benefits:

* **Guaranteed optimality** (with admissible heuristic in A\*)
* **Finds global best solution**, not just a local improvement
* **Backtracking supported** through came\_from map

## Limitations:

* **Higher memory usage** — keeps track of many nodes
* **Slower** than local methods for very large spaces
* Depends heavily on the **quality of heuristic**

**Examples include Best first search and A\* search:**

Like the depth-first and breadth-first search, **best-first search** uses two-lists.

1. OPEN: to keep track of the current fringe of the search.
2. CLOSED: to record states already visited.
3. Order the states on OPEN according to some heuristic estimate of their closeness to a goal.
4. Each iteration of the loop considers the most promising state on the OPEN list.
5. Your solution is found in the closed set

With best-first, node is selected for expansion based on evaluation function f(n).

Often, for best-first algorithms, f is defined in terms of a heuristic function, h(n).

Best-First Search algorithms constitute a large family of algorithms, with different evaluation functions.

Each has a heuristic function h(n)

Recall:

* g(n) = cost from the initial state to the current state n.
* h(n) = estimated cost of the cheapest path from node n to a goal node.
* f(n) = evaluation function to select a node for expansion (usually the lowest cost node).

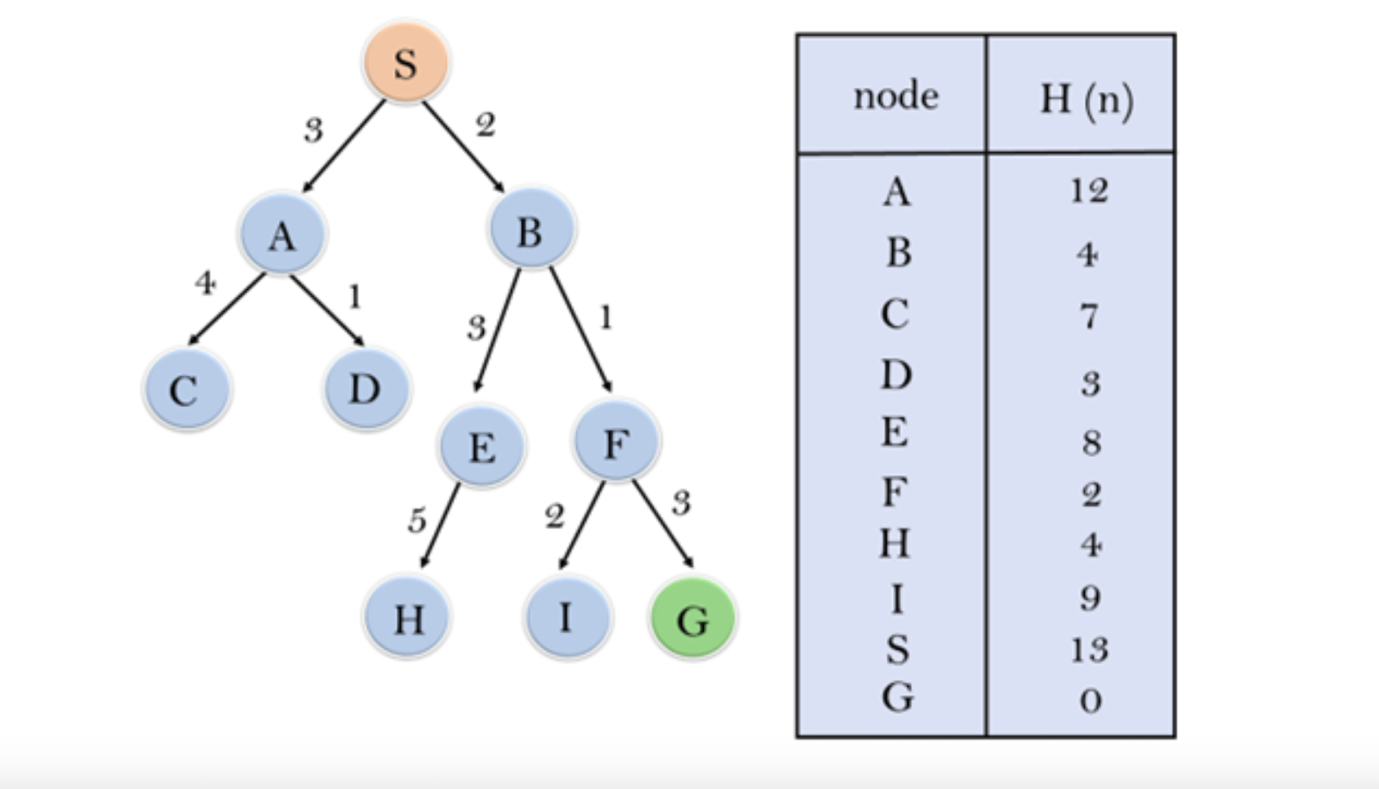
**What is Best-First Search?**

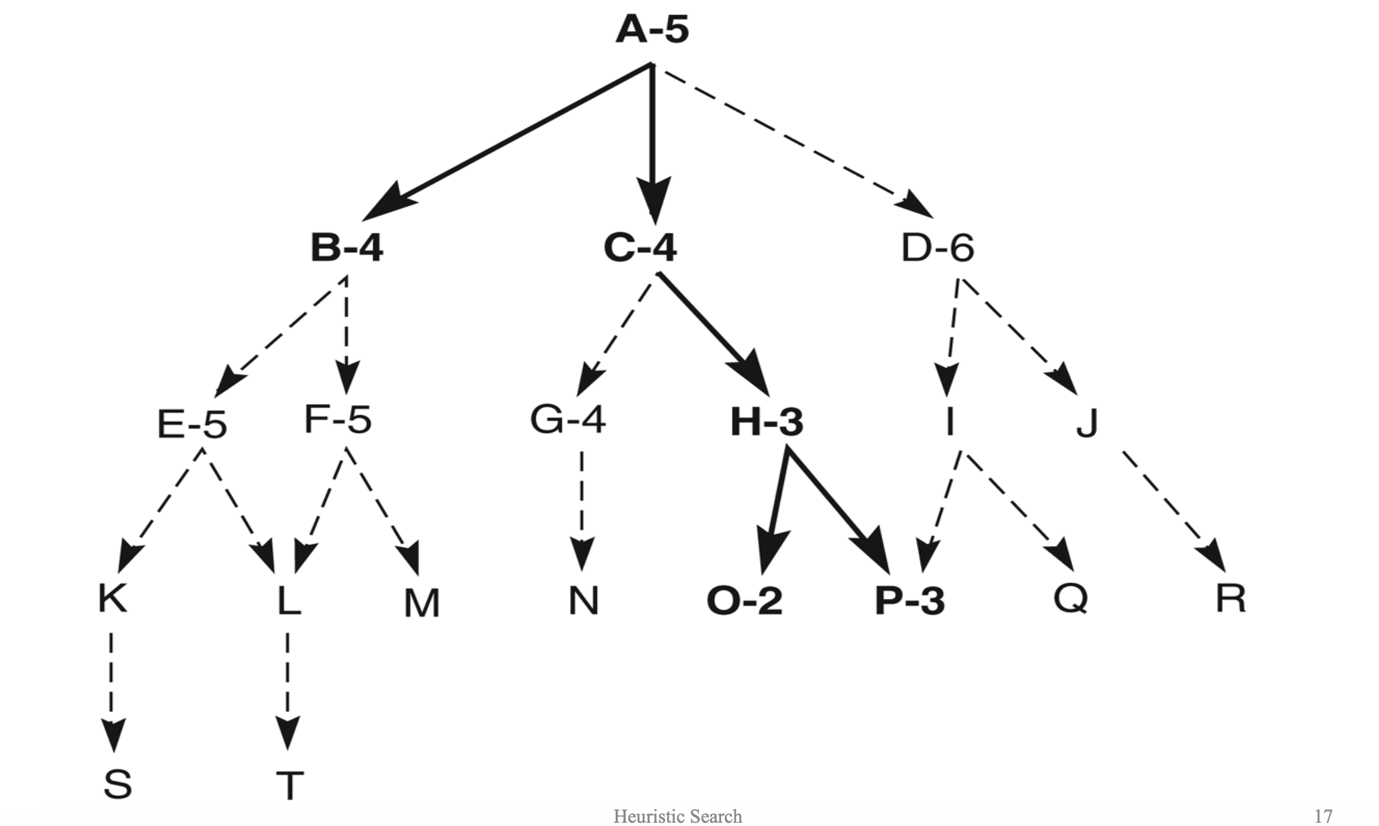
* Best-First Search is a **global heuristic search** strategy.
* It uses a **heuristic function h(n)** to estimate how close a node is to the goal.
* It selects the **next node with the lowest h(n)** from the frontier (priority queue).

## Hand-Run Best-First Search:

1. **Start at initial node**, insert it into the frontier.
2. **Repeat:**
   * Remove node with **lowest h(n)** from frontier.
   * If it's the goal, stop!
   * Otherwise, expand its neighbors.
   * Add neighbors to the frontier (skip already explored nodes).
   * Sort the frontier based on heuristic values.
3. **Track the explored nodes** to avoid cycles.
4. (Optional) **Track parent pointers** to reconstruct the path.

NB: Re-order nodes in the open set by heuristic merit(best leftmost)

1.

2.

3. 