0.1 A* Search

- *Algorithm:* A* Search (algo. 1)
- *Input:* A graph $G = \langle V, E \rangle$, starting point S and goal G
- *Complexity:* $O(b^d)$, where d is the depth of solution and b is branching factor
- *Data structure compatibility:* priority queue, graph
- *Common applications:* search optimization, game development

Problem. A* Search

A* (or A-star) is a classic graph-tranversal and path finding algorithm. It is proposed by Peter Hart, Nils Nilsson and Bertram Raphael from Stanford Research Institute in 1986 [2]. It is an extension on the previous Dijkstra's algorithm. Introducing of concept of heuristic functions, A* is complete and optimal as well as optimally efficient [4], on the contrary to its ancestor. This makes it still one of the prior choices in many application cases.

Description

One of the complete and optimal algorithm is the uniform-cost search (UCS), originating from Dijkstra [4], where in each iteration the nodes which would add least cost would be chosen, i.e. the node closest to the current one. UCS expands the nodes evenly in all directions, making it slow and space-costly.

An naive heuristic search, greedy best-first algorithm, always chooses the nodes which seems closest to the goal. These distances are made by estimation, which can give us a genaral direction of searching and greatly save run-time in some cases. However, since the heuristics values are estimated, what solution the algorithm would return dependent on the choice of heuristic functions. This means that this scheme is neither complete nor optimal. What's more, the quality of heuristic function also affects the degree of complexity reduction.

The A* algorithm is the combination of the above two algorithm. As can be see in the pseudocode below1. Denote the cost from starting point S to node n to be $g(n)$ and the heuristic to be $h(n)$, what A* algorithm pays attention to is their sum $f(n) = g(n) + h(n)$. In each iteration, the nodes with smallest $f(n)$ would be seleted and expanded on. The algorithm terminates when the goal is reached.

To run A*, we need to initialize two sets *open* and *closed*, where stores the nodes to be expanded and having been expanded. What means by "expanding" a node is to visit its neighbours and let them wait to be expanded. Firstly, add the starting point to *open*. We would always selete and remove a node n from *open* with the smallest $f(n)$ so as to expand it and add to *closed*. Then we calculate the cost of the neighbours of n and add them to *open*. The algorithm continues until we the node wo remove from *open* is exactly the goal. If goal is not returned even if *open* is empty, we can be sure there is no such a path.

There are two points to be made here. Firstly, since we always need to selete the node from *open* with the smallest f, we are actually applying a priority queue here, whose priority is just the value of f. Another point is that although A* algorithm can reach the goal by travalling along the shortest path, it can return directly return it. **Backtracking** is necessary to record and generate the path. One of the simple way is to attach the nodes with their predecessors when pushing them into the *open* set.

It should be clarified that to guarantee the optimality, the quality of heuristic function matters. The first requirement is **admissible** heuristic, which implies that the heuristic never over-estimates the potential costs. Another property to ensure optimality is consistency. A heuristic is consistent if for every node and each of its

successor n' generated by action a, there is a triangle inequality

$$h(n) - h(n') \leq cost(n, n').$$

That is to say, the difference of heuristic for two neighbouring nodes should never be larger than the true cost to travel between them.

It is sufficient to show that consistency is a stricter requirement than admissibiliy, and being consistent suffices to be admissible. Implementation of A* is easy and able to be reused, but the establishment of a good (consistent) heuristic for the application case is what important and hard job to do.

A* can find the shortest path on a graph for given starting point and goal, therefore it can be applied to all the mathematical or engineering problems that can be transferred in a graph search problem, such as navigation through a maze, routing communication traffics and integerted circuit design. A* itself is invented in the project *Shakey the robot* as the way of path searching [3]. In modern time, one of the major field using A* is game development. As the most popular path-finding algorithm nowadays [1], A* is what help most of the AI players find their way or decide their action.

Algorithm 1: A* Search

Input : A graph $G = \langle V, E \rangle$, starting point S and goal G
Output: the shortest path from S to G if exists

```
closed ← {}
open ← priority queue with f as priority
set S.g = 0
Push(open, S) with S.f = 0 + h(S)
while open is not empty do
    node ← remove_front(open)
    if node == G then
        return
    end if
    if node not in closed then
        add node to closed
        for n in Adjcent(node) do
            n.g = node.g + Distance(node, n)
            Push(open, n) with n.f = n.g + h(n)
        end for
    end if
end while
return Failure
```

References.

[1] Xiao Cui and Hao Shi. "A*-based pathfinding in modern computer games". In: *International Journal of Computer Science and Network Security* 11.1 (2011), pp. 125–130 (cit. on p. 2).

[2] Peter E Hart, Nils J Nilsson, and Bertram Raphael. "A formal basis for the heuristic determination of minimum cost paths". In: *IEEE transactions on Systems Science and Cybernetics* 4.2 (1968), pp. 100–107 (cit. on p. 1).

[3] Andra Keay and Silicon Valley Robotics. *Shakey is first robot to receive IEEE Milestone award*. URL: https://robohub.org/shakey-is-first-robot-to-receive-ieee-milestone-award/ (cit. on p. 2).

[4] Stuart J. Russell, Peter Norvig, and Ernest Davis. *Artificial intelligence: a modern approach*. 3rd ed. Prentice Hall series in artificial intelligence. Upper Saddle River: Prentice Hall, 2010. ISBN: 9780136042594 (cit. on p. 1).