Алгоритм A*

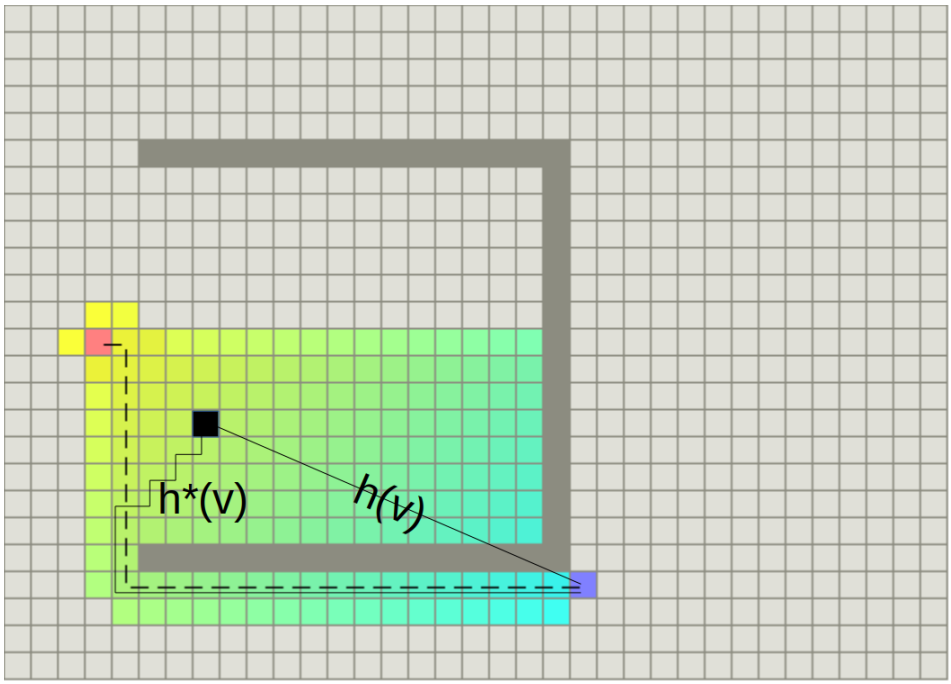
Задача

Мы хотим переместить робота из одной точки в другую. Перед тем как робот начинает движение, он должен знать, по какой траектории осуществлять движение. При этом хотелось бы, чтобы робот пошел по безопасному пути, избегая все препятствия. И, конечно, мы хотим, чтобы робот выбрал самый короткий путь для движения. Такой путь позволяет найти алгоритм A*.

Математическая постановка задачи

Пространство, в котором движется робот, представляется в виде неориентированного взвешенного *графа* $G = (V, E)$ с весовой функцией $g : E \longrightarrow R$, которая представляет собой стоимость перемещения по ребру. *Вершины графа* - положения, которые может занимать робот.
Ребра - пути, по которым робот может перемещаться, чтобы добраться из одной вершины в другую. Обозначим множество всех путей за P
В начале нам известны стартовая вершина v_S и целевая вершина v_F.
Определим *путь* $\pi\{v_S, \ldots, v_F\}$ как последовательность вершин и ребер, которая связывает v_S и v_F
Вес пути - сумма весов ребер, через которые проходит путь:
$g(\pi\{v_S, \ldots, v_F\}) = \sum_{i=S}^{F} g(v_i)$
Найти: путь из v_S в v_F с наименьшей стоимостью:
$\pi\{v_S, \ldots, v_F\} = argmin_{\pi \in P} g(\pi)$

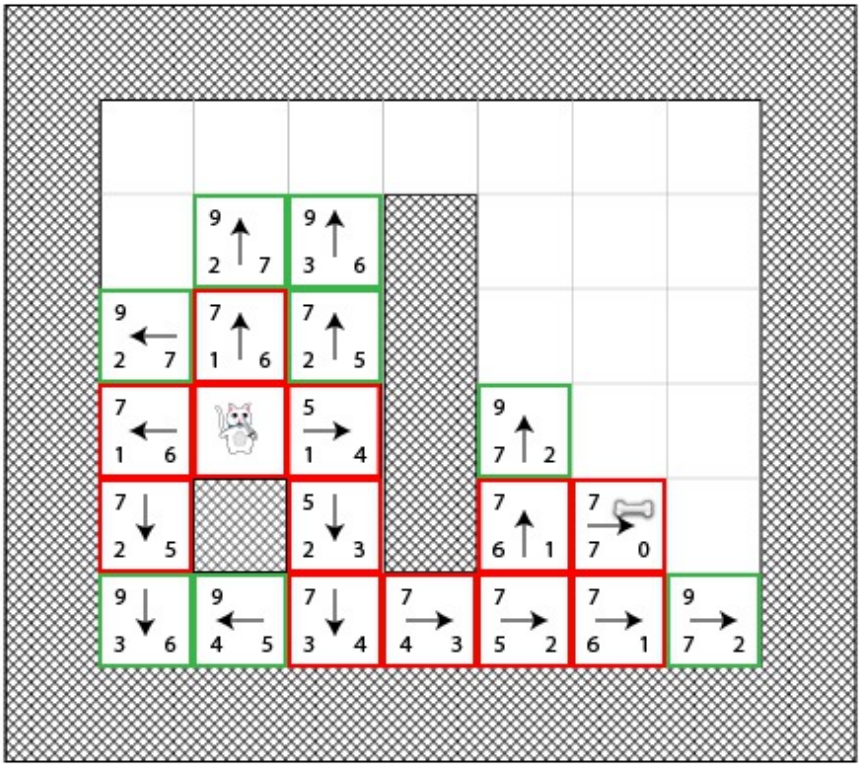
Для решения задачи определим *эвристическую функцию*:
$f(v_i) = g(v_i) + h(v_i)$
где $g(v_i = g(\{v_S, \ldots, v_i\})$ - стоимость пути из v_S в v_i
$h(v_i)$ - предполагаемая стоимость пути из вершины v_i в v_F.
Функция $h(v_i)$ при этом должна удовлетворять условию $h(v_i) \leq h^*(v_i)$, где h^* - действительная стоимость пути из v_i в v_F. Если это условие не будет выполнено, алгоритм может найти путь, который не является самым коротким.
Разница между $h(v)$ и $h^*(v)$ показана на рисунке



В данной реализации в качестве функции $h(v_i)$ используется евклидово расстояние между двумя точками.
Таким образом мы знаем стоимость перемещения из точки старта в текущую точку и предполагаем расстояние до цели.

Идея алгоритма

На каждом шагу алгоритма мы раскрываем ячейку, т.е. узнаем всех ее соседей, считаем для каждого функцию $f(v)$, обозначаем текущую ячейку как parent для этого соседа и запоминаем соседа (добавляем его в отдельный список open). Следующая ячейка для раскрытия - ячейка из open с минимальным значением функции $f(v)$.
Раскрытые ячейки запоминаем, помещая их в список closed. К ним больше не возвращаемся.



Раскрывая ячейки, мы рано или поздно придем к конечной вершине v_F. В результате мы можем восстановить путь как последовательность parent ячеек. Т.к. мы все время шли по ячейкам с минимальным значением $f(v)$, то путь будет иметь минимальную стоимость.

Algorithm 1 A* algorithm

```
open = ∅;
close = ∅;
open.push(start);
g[start] = 0;
f[start] = g[start] + h[start];
while Q.size() != 0 do
    current = вершина из open с минимальным значением f;
    if current == goal then
        return true;
    end if
    open.remove(current);
    close.push(current);
    for v : соседи вершины current do
        gtemp[v] = g[current] + d[current, v]
        if v ∈ open and gtemp  g[v] then
            continue;
        end if
        if v ∉ open or gtemp  g[v] then
            parent[v] = current;
            g[v] = gtemp;
            f[v] = g[v] + h[v];
            if v ∉ close then
                close.push(v);
            end if
        end if
    end for
end while
return false;
```