

A*

本文介绍 A* 搜索算法。



A* 搜索算法 (A* search algorithm, A* 读作 A-star), 简称 A* 算法, 是一种在带权有向图上, 找到给定起点与终点之间的最短路径的算法。它属于图遍历 (graph traversal) 和最佳优先搜索算法 (best-first search), 亦是 BFS 的改进。

过程

A* 算法的目标是找到有向图上从起点 s 到终点 t 的最短路径。设 $d(x, y)$ 为结点 x 与 y 之间的距离, 也就是它们之间最短路径的长度。记 $g(x) = d(s, x)$ 为从起点 s 到结点 x 的距离函数, $h^*(x)$ 为从结点 x 到终点 t 的距离函数, $h(x)$ 为 $h^*(x)$ 的一个估计¹。最后, 记从 s 出发经由 x 到达 t 的最短路径长度的估计为

$$f(x) = g(x) + h(x).$$

搜索时, A* 算法每次从优先队列中取出一个 f 最小的结点。然后, 将它的所有后继结点 x 都推入优先队列中, 并利用实际记录的 $g(x)$ 和估计的 $h(x)$ 更新 $f(x)$ 。

性质

由于 $h^*(x)$ 的实际值在搜索的时候是未知的, 所以, 需要使用容易计算的 $h(x)$ 作为它的估计。A* 搜索的实际复杂度就取决于这一估计函数 $h(x)$ 的性质。容易想象, 如果 $h \equiv h^*$, 也就是说, 估计是精确的, 那么, 搜索过程就会严格按照最短路径前进。而如果 $h \equiv 0$, 那么, A* 算法就退化为 [Dijkstra 算法](#); 当 $h \equiv 0$ 并且边权为 1 时, 这就是 [BFS](#)。

假设图没有负权边。如果估计 $h(x)$ 永远不超过实际距离 $h^*(x)$, 即 $0 \leq h \leq h^*$, 那么, A* 算法就一定能够找到最优解。满足这一条件的估计函数 $h(x)$ 称为 **可采纳的** (admissible)。根据前文的讨论, h 越接近 h^* , 相应的 A* 算法效率就越高。一般来说, 在最差情形中, 算法会经过所有满足

$$f(x) = g(x) + h(x) \leq C^*$$

的结点, 其中, C^* 是起点 s 和终点 t 之间的最短距离。直觉上, h 越接近 h^* , 每次扩展时, 能够满足该条件的后继结点就越少, 因此, 算法搜索到的分支就越少。所以, A* 算法可以看作是对搜索算法的一种「剪枝」优化。

如果 h 不仅是可采纳的, 还是一致的 (consistent), 即

$$h(x) \leq h(y) + d(x, y),$$

那么，A* 算法不会将已经弹出队列的结点再次加入队列。一致性条件，可以理解为结点 x, y, t 之间的三角形不等式。

例题

A* 算法的一个经典应用是解决 k 短路问题。关于该问题的描述、A* 做法，以及复杂度更优的可持久化可并堆做法，请移步 [k 短路问题](#) 页面。

本节介绍一个可以用 A* 算法解决的经典问题。

八数码

在 3×3 的棋盘上，摆有八个棋子，每个棋子上标有 1 至 8 的某一数字。棋盘中留有一个空格，空格用 0 来表示。空格周围的棋子可以移到空格中，这样原来的位置就会变成空格。给出一种初始布局和目标布局（为了使题目简单，设目标状态如下），找到一种从初始布局到目标布局最少步骤的移动方法。

123
804
765

解题思路

h 函数可以定义为，不在应该在的位置的棋子个数。容易发现， h 既是可采纳的，也是一致的。此题可以使用 A* 算法求解。

参考代码

```
1  #include <algorithm>
2  #include <cstring>
3  #include <iostream>
4  #include <queue>
5  #include <set>
6  using namespace std;
7  constexpr int dx[4] = {1, -1, 0, 0}, dy[4] = {0, 0, 1, -1};
8  int fx, fy;
9  char ch;
10
11 struct matrix {
12     int a[5][5];
13
14     bool operator<(matrix x) const {
15         for (int i = 1; i <= 3; i++)
16             for (int j = 1; j <= 3; j++)
17                 if (a[i][j] != x.a[i][j]) return a[i][j] < x.a[i][j];
18         return false;
19     }
20 } f, st;
21
22 int h(matrix a) {
23     int ret = 0;
24     for (int i = 1; i <= 3; i++)
25         for (int j = 1; j <= 3; j++)
26             if (a.a[i][j] != st.a[i][j] && a.a[i][j] != 0) ret++;
27     return ret;
28 }
29
30 struct node {
31     matrix a;
32     int t;
33
34     bool operator<(node x) const { return t + h(a) > x.t + h(x.a); }
35 }
36 } x;
37
38 priority_queue<node> q; // 搜索队列
39 set<matrix> s;         // 防止搜索队列重复
40
41 int main() {
42     cin.tie(nullptr)->sync_with_stdio(false);
43     st.a[1][1] = 1; // 定义标准表
44     st.a[1][2] = 2;
45     st.a[1][3] = 3;
46     st.a[2][1] = 8;
47     st.a[2][2] = 0;
48     st.a[2][3] = 4;
49     st.a[3][1] = 7;
```

```

50     st.a[3][2] = 6;
51     st.a[3][3] = 5;
52     for (int i = 1; i <= 3; i++) // 输入
53         for (int j = 1; j <= 3; j++) {
54             cin >> ch;
55             f.a[i][j] = ch - '0';
56         }
57     s.insert(f);
58     q.push({f, 0});
59     while (!q.empty()) {
60         x = q.top();
61         q.pop();
62         if (!h(x.a)) { // 判断是否与标准矩阵一致
63             cout << x.t << '\n';
64             return 0;
65         }
66         for (int i = 1; i <= 3; i++)
67             for (int j = 1; j <= 3; j++)
68                 if (!x.a.a[i][j]) fx = i, fy = j; // 查找空格子 (0号点) 的
69 位置
70         for (int i = 0; i < 4; i++) { // 对四种移动方式分别进行搜索
71             int xx = fx + dx[i], yy = fy + dy[i];
72             if (1 <= xx && xx <= 3 && 1 <= yy && yy <= 3) {
73                 swap(x.a.a[fx][fy], x.a.a[xx][yy]);
74                 if (!s.count(x.a))
75                     s.insert(x.a),
76                     q.push({x.a, x.t + 1}); // 这样移动后, 将新的情况放
77 入搜索队列中
78                 swap(x.a.a[fx][fy], x.a.a[xx][yy]); // 如果不这样移动的情
79 况
80             }
63         }
64     }
65     return 0;
66 }

```

参考资料与注释

- [A* search algorithm - Wikipedia](#)

1. 此处的 h 意为 heuristic。详见 [启发式搜索 - 维基百科](#) 和 [A* search algorithm - Wikipedia](#) 的 Bounded relaxation 一节。←

🔧 本页面最近更新：2025/8/10 14:51:39，[更新历史](#)

✎ 发现错误？想一起完善？ [在 GitHub 上编辑此页！](#)

👤 本页面贡献者: [lr1d](#), [Enter-tainer](#), [Henry-ZHR](#), [hsfzLZH1](#), [ksyx](#), [ouuan](#), [Xeonacid](#), [billchenchina](#), [c-forrest](#), [ChungZH](#), [cn-Mouxy](#), [flylai](#), [greyqz](#), [iamtwz](#), [interestingLSY](#), [kenlig](#), [leoleoasd](#), [NachtgeistW](#), [ree-chee](#), [StudyingFather](#), [WaterWan](#), [Wh1tD](#)

© 本页面的全部内容在 [CC BY-SA 4.0](#) 和 [SATA](#) 协议之条款下提供, 附加条款亦可能应用