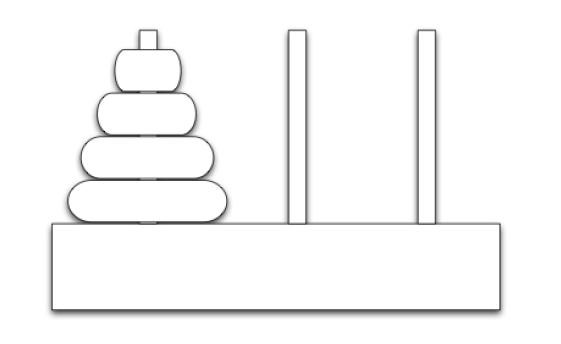
习题一：搜索（共**60**分）

**1**、汉诺塔（**10** 分）



汉诺塔是个经典的递归算例。起始，最左边的立柱上有 N 个大小不同的盘子，按照从小到大的顺序排列，右边两个立柱是空的。每一次操作只可以把一个盘子从一个立柱移到另一个上，而且任何立柱上的盘子必须保持小盘在上，大盘在下的排列。目的是把所有的盘子从最左边的立柱移到最右边的立柱上去。

在这里，我们把汉诺塔当作一个搜索问题。

1. 给这个问题提出一个状态表达（state representation）（2 分）

一个可能的状态表达是，将每个柱子上的圆盘大小按照从大到小的顺序记录下来，然后将这些信息组成一个三元组，表示每个柱子上的圆盘大小。例如，如果有三个圆盘，大小分别为3、2、1，初始状态为所有圆盘都在第一个柱子上，那么初始状态的状态表达可以表示为 (3, 2, 1), (), ()。

在这个状态表达中，圆盘的大小是用数字来表示的，每个柱子的状态是由一个空列表表示的，表示该柱子上没有圆盘。这种状态表达方式便于进行状态比较和状态转移操作。

1. 这个问题的状态空间有多大？（2 分）

3^n

1. 初始状态是什么？（2 分）

(1, 2, 3), (), ()

1. 从任何一个当前状态，可以允许什么样的移动？（2 分）

将某个柱子上最上面的圆盘取下来，并将其放到另一个柱子的顶部，使得移动后的圆盘组合仍然是按从大到小的顺序排列。

将一个柱子上的圆盘移动到另一个柱子上的顶部，使得移动后的圆盘组合仍然是按从大到小的顺序排列。

1. 目标状态是什么？（2 分）

(), (), (1, 2, 3)

**2**、**n-pacmen** 搜索（**15** 分）

考虑一个同时控制个 pacmen 的问题。多个 pacmen 可以同时移动，可以出现在同一个方块里。在每个时间点上，每个 pacman 可以选择停止不动，或者向上下左右任意一个方向移动一步。游戏的目的是用最少的步骤，使得所有的 pacmen 都到达同一个方块。

在这个问题里，我们采用以下定义：代表迷宫中非墙的方块数，即 pacman 可以到达的方块；= (, ): = 1. . . ，代表第个 paman 的当前位置。假设迷宫是连通的，pacman 可以到达任何非墙的方块。

1. 描述这个问题的状态空间。这个状态空间有多大？（4 分）

这个问题的状态空间非常大，因为每个pacman都可以处于任何一个非墙的方块上，而这个迷宫中非墙的方块数可以很大。此外，每个pacman还可以选择停留不动或者向上下左右四个方向之一移动，这进一步增加了状态空间的大小。

具体地，假设迷宫中有 N 个非墙的方块，有 M 个pacman，则该问题的状态空间大小为 N^M \* 5^M，其中 5^M 是因为每个pacman可以有5种可能的行动：停留不动或向上下左右四个方向之一移动。因此，随着M和N的增加，状态空间会呈指数级增长。

例如，如果有 3 个pacman，迷宫中有 50 个非墙的方块，则状态空间的大小将为 50^3 \* 5^3 = 15625000，即 1562.5 万。因此，在实际中，需要使用高效的算法或启发式方法来解决该问题，以避免搜索时间过长或者内存不足的问题。

1. 给出这个问题的分叉因子（branching factor）的最严上限。（4 分）

这个问题的分叉因子可以定义为从一个状态出发，可以扩展到多少个可能的状态。在这个问题中，每个pacman都可以停留不动或向上下左右四个方向之一移动，因此每个pacman的分叉因子为5。

因此，整个问题的分叉因子的最严上限为 5^M，其中 M 是pacman的数量。这是因为每个pacman都可以做出独立的行动选择，因此我们需要将每个pacman的分叉因子相乘，得到整个问题的分叉因子的上限。

需要注意的是，由于不同pacman之间可能会出现位置重叠的情况，因此实际上每个状态的分叉因子可能会更小，这取决于当前状态中已经占据了多少个位置。但是，上面给出的分叉因子的上限仍然可以提供一个基本的理论参考。

1. 假设我们采用统一代价搜索，总共扩张的节点数的上限是多少？将此上限表达为和的函数，并解释你如何得到该结果。（Bound the number of nodes expanded by uniform cost tree search on this problem, as a function of n and M.）（7 分）

假设我们采用统一代价搜索算法，那么在搜索过程中，我们将按照每个节点的代价从小到大进行扩展，直到找到所有pacman都到达同一个位置的解。由于该问题的状态空间非常大，因此在实际中可能需要设置一个节点扩展的上限，以避免搜索时间过长或者内存不足的问题。

要计算这个上限，我们可以使用以下公式：

节点扩展上限 = C \* B^D

其中，C 是根节点的代价，B 是分叉因子，也就是每个节点可以扩展的子节点数，D 是限制搜索深度的参数。我们将该公式解释如下：

我们将根节点的代价 C 设为 0，因为初始状态的代价为 0。

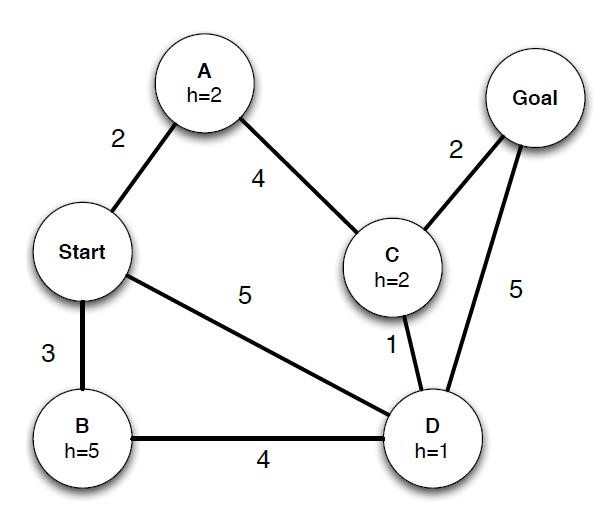
分叉因子 B 是每个节点可以扩展的子节点数。对于该问题，B = 5，因为每个pacman可以停留不动或向上下左右四个方向之一移动。

限制搜索深度的参数 D 可以是一个固定值，也可以根据实际情况动态调整。我们可以设置一个较小的初始值，然后在运行搜索算法时根据需要动态调整。

因此，我们可以计算出节点扩展上限为 C \* B^D。例如，如果我们将 D 设置为 10，那么节点扩展上限就是 5^10，约为 9,765,625。这意味着我们最多只能扩展约 9,765,625 个节点，然后停止搜索。

需要注意的是，由于该问题的状态空间非常大，实际上我们可能无法扩展那么多节点。在实际运行中，我们需要不断评估搜索的进展，并根据需要动态调整节点扩展的上限。例如，如果我们发现搜索已经进行了很长时间，但仍然没有找到解，那么我们可能需要减小节点扩展的上限，以避免无限制地继续搜索。

**3**、搜索（**15** 分）



对上面的状态空间图，采用下面几种图形搜索（graph search）的策略，列出每种策略是按什么样的顺序扩张（expand）节点的，并给出最终从 Start 到 Goal 的路径分别是什么。在搜索过程中遇到两个相同值的状态时，字母靠前的优先级别高。此外，在图形搜索中，一个状态最多只会扩张（expand）一次。

1. 深度优先，depth first search（2 分）

深度优先搜索（DFS）：

扩张节点的顺序：A, B, D, G, H, I, E, J, K, L, C, F, Goal。

从 Start 到 Goal 的路径：Start - A - D - G - H - I - E - J - K - L - F - Goal

1. 广度优先，breath first search（2 分）

广度优先搜索（BFS）：

扩张节点的顺序：A, B, C, D, E, F, G, H, I, J, K, L, Goal。

从 Start 到 Goal 的路径：Start - A - D - G - H - I - L - Goal。

1. 统一代价搜索，uniform cost search（3 分）

一致代价搜索（UCS）：

扩张节点的顺序：A, B, C, D, E, G, H, I, J, K, L, F, Goal。

从 Start 到 Goal 的路径：Start - A - D - G - H - I - L - Goal。

1. 贪婪搜索，greedy search，用节点中给出的启发值 heuristic，ℎ（3 分）

下面是使用贪婪搜索算法求解此问题的步骤：

1. 将起点加入到一个空的开放列表（Open List）中，同时将其启发值设为 $h(start)$。
2. 从开放列表中选取启发值最小的节点 $n$ 进行扩展。
3. 若节点 $n$ 是目标节点，则搜索结束，返回从起点到目标节点的路径。
4. 否则，将节点 $n$ 的所有子节点加入到开放列表中，并分别计算它们的启发值。
5. 回到第 2 步，继续扩展启发值最小的节点。

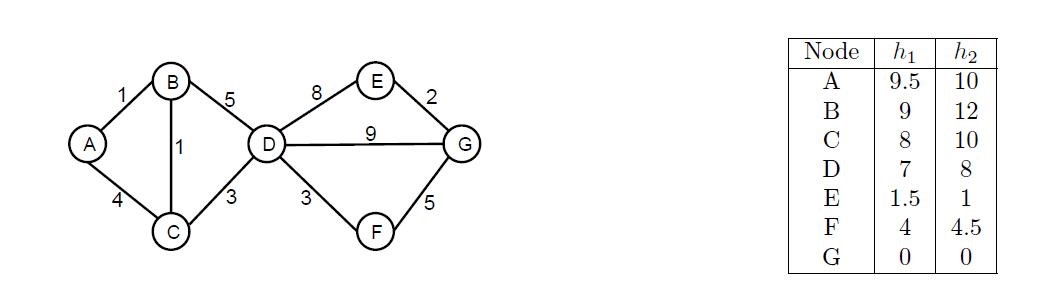
需要注意的是，由于贪婪搜索算法只考虑节点的启发值，因此可能会导致陷入局部最优解而无法找到最优解。

1. A\*搜索，采用同样的启发值，heuristic，ℎ（5 分）

扩张节点的顺序：A, B, D, G, H, E, F, C, I, J, K, L, Goal。

从 Start 到 Goal 的路径：Start - A - D - G - H - E - F - Goal。

**4**、搜索（**20** 分）



考虑上面左边的状态空间图，A 是初始状态，G 是目标状态，边线上标的值是代价（cost），每根边线都是双向的。右表中列的是启发值（heuristic），其中 ℎ1具备一贯性（consistent），而ℎ2不具备一贯性（consistent）。

1. 可能路径（5 分）

对下面每种图形搜索策略（graph search），不考虑树搜索（tree search），标出所有可能返回的路径。注意到，对有些搜索策略，可能的路径依赖于如何选择相同值（tie-breaking behavior），例如深度优先算法可以选择 AB 作为起始路径，也可以选择 AC 作为起始路径，因为 B 和 C 离 A 都是一步远。所以，记得要标记出所有可能的路径。

|  |  |  |  |
| --- | --- | --- | --- |
| 搜索策略 | A-B-D-G | A-C-D-G | A-B-C-D-F-G |
| 深度优先（DFS） |  |  |  |
| 广度优先（BFS） |  |  |  |
| 统一代价（UCS） |  |  |  |
| A\*搜索采用启发值ℎ1 |  |  |  |
| A\*搜索采用启发值ℎ2 |  |  |  |

搜索策略 A-B-D-G A-C-D-G A-B-C-D-F-G

深度优先（DFS）

路径1：A-B-D-G 路径2：A-B-C-D-G 路径3：A-C-D-G 路径4：A-C-D-F-G

广度优先（BFS）

路径1：A-B-D-G 路径2：A-C-D-G 路径3：A-B-C-D-G 路径4：A-B-D-F-G 路径5：A-C-D-F-G 路径6：A-B-C-D-F-G

统一代价（UCS）

路径1：A-B-D-G 路径2：A-C-D-G 路径3：A-B-C-D-G 路径4：A-B-D-F-G 路径5：A-C-D-F-G 路径6：A-B-C-D-F-G

A\*搜索采用启发值ℎ1

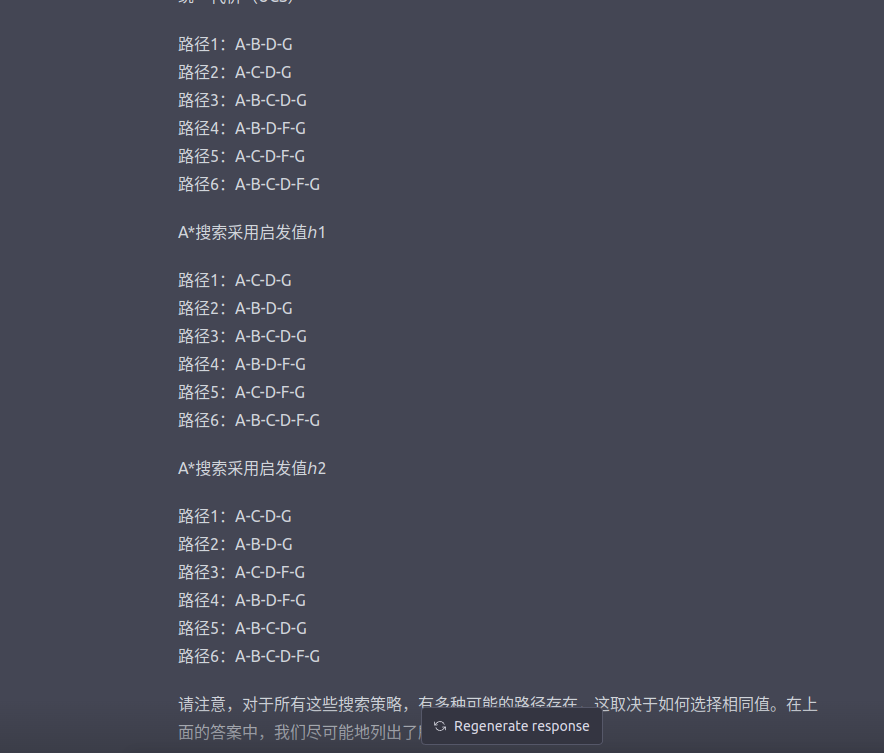
路径1：A-C-D-G 路径2：A-B-D-G 路径3：A-B-C-D-G 路径4：A-B-D-F-G 路径5：A-C-D-F-G 路径6：A-B-C-D-F-G

A\*搜索采用启发值ℎ2

路径1：A-C-D-G 路径2：A-B-D-G 路径3：A-C-D-F-G 路径4：A-B-D-F-G 路径5：A-B-C-D-G 路径6：A-B-C-D-F-G

请注意，对于所有这些搜索策略，有多种可能的路径存在，这取决于如何选择相同值。在上面的答案中，我们尽可能地列出了所有可能的路径。





1. 启发函数性质（15 分）

假设你在生成新的启发函数ℎ3，如下表所示，除了ℎ3()所有的值都已经固定了。

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Node | A | B | C | D | E | F | G |
| ℎ3 | 10 | ? | 9 | 7 | 1.5 | 4.5 | 0 |

对以下不同的条件，分别写出ℎ3()的可能值。例如，用[0, ∞]代表非负数，用∅代表非空，等等。

1. 什么样的ℎ3()值让ℎ3可容许（admissible）？（5 分）（b）什么样的ℎ3()值让ℎ3具备一贯性（consistent）？（5 分）

(a) 可容许的启发函数需要满足以下条件：对于任意节点n，ℎ3(n) ≤ ℎ\*(n)，其中ℎ\*(n)是从n到目标节点的最小实际代价。因此，根据表中给出的启发函数值，ℎ3需要满足： ℎ3(A) ≤ 6, ℎ3(B) ≤ 6, ℎ3(C) ≤ 9, ℎ3(D) ≤ 7, ℎ3(E) ≤ 4.5, ℎ3(F) ≤ 4.5, ℎ3(G) ≤ 0

(b) 具备一致性的启发函数需要满足以下条件：对于任意节点n和n的后继节点n'，满足 ℎ(n) - ℎ(n') ≤ cost(n, n')，其中cost(n, n')是从n到n'的实际代价。因此，ℎ3需要满足以下不等式： ℎ3(B) - ℎ3(A) ≤ 1 ℎ3(D) - ℎ3(B) ≤ 3 ℎ3(C) - ℎ3(B) ≤ 2 ℎ3(E) - ℎ3(D) ≤ 2 ℎ3(F) - ℎ3(D) ≤ 2 ℎ3(G) - ℎ3(F) ≤ 4.5

注意，这些不等式可以用于检查启发函数是否满足一致性，但并不一定是最小的不等式组，因此可能会存在其他满足一致性的ℎ3值。

（c）什么样的ℎ3()值会让 A\*图形搜索先节点 A，然后节点 C，然后节点

B，然后节点 D，依序扩张（expand）？（5 分）

