DS4001-25SP-HW1：搜索

TA：李睿哲

2025 年 3 月 14 日

作业说明

本次作业包括书面部分（问题回答）和编程部分（代码填空）。

书面部分需使用 LaTeX 完成，模板文件为 `report_template.tex`。注意：为方便助教批改，你还需要将编程部分完成后的代码复制到模板中的对应位置。书面作业完成后，请导出为 `report.pdf` 文件。

编程部分的完整代码位于 `Project` 文件夹中。你仅需在 `submission.py` 中指定位置完成代码，具体如下：

```
# BEGIN_YOUR_CODE

# END_YOUR_CODE
```

除 `submission.py` 之外，请勿修改其他任何文件。建议每完成一部分代码后，及时运行单个样例进行测试，例如执行 `python grader.py 1a-1-basic`，以便快速排查代码问题。每个测试样例的编号已经在 README 中给出。

完成所有代码后，运行 `python grader.py` 可查看当前得分。在本次作业中，每一道代码题的得分将有一半来自 grader.py 给出的分数，另一半来自助教对代码的打分。`grader.py` 包含可见样例和隐藏样例，你在本地运行时看到的得分仅为可见样例的得分，最终成绩将同时包含可见和隐藏样例。

完成所有内容后，请将 `submission.py` 与 `report.pdf` 打包成名为“学号 _ 姓名 _HW1.zip”的压缩文件，并上传至 BB 系统。助教将根据你提交的文件进行评分。

注意事项:

- 本次作业必须独立完成，任何形式的抄袭都不允许。如发现有互相抄袭的情况，抄袭者将平分获得的分数。
- 作业的截止时间为 2024 年 4 月 10 日 23:59:59。迟交 1/2/4/7 天分别会扣除 5/15/40/100 分。
- 如果在做作业过程中遇到问题，可以在 Github Issue 中提问或者线下在答疑课提问，也欢迎同学们相互解答问题。

作业正文

简介 [0%]

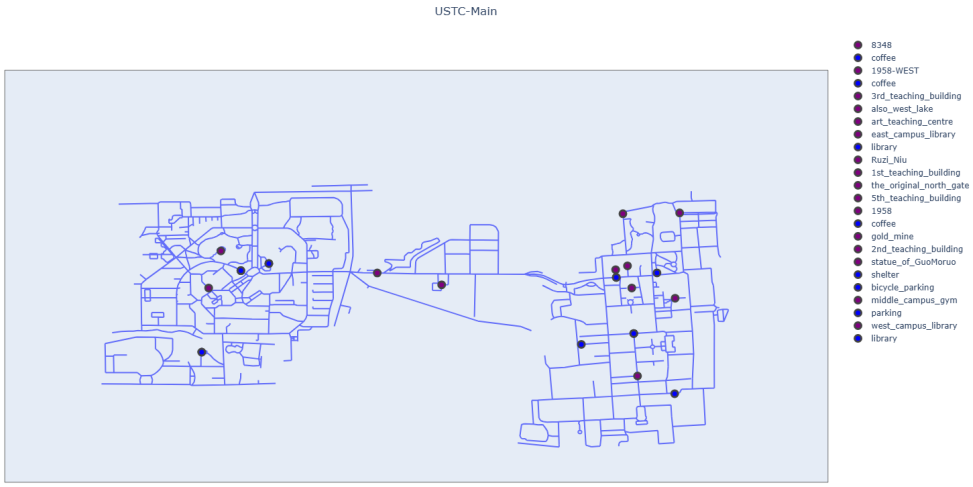

图 1: 你可以说出图中各个点对应哪些地方吗?（不要求回答）

现有地图软件均有路线规划的功能，可以找到从 A 点到 B 点的最佳路径，若仅用距离来衡量路径的优劣，则该问题则退化为了求最短路径。

在本次作业中，我们将制作一个“校内路线规划”程序，求解从校内一点到另一点的最短路径。此外，我们还将为该程序增加更强大的功能：你不仅可以直接查询从宿舍到三教的最短路径，还可以进行更复杂的查询，比如从宿舍出发，途经图书馆（不限东区图书馆或西区图书馆）和咖啡店，最后到达某个自习室的最短路径。

0 代码理解 [20%]

在实际科研和工业应用中，我们很少从零开始编写完整的代码，而更多地是在现有代码框架的基础上进行补充和修改。这需要我们拥有“代码理解”的能力。这种能力不仅要求我们能够快速掌握他人设计的程序结构和逻辑，更需要我们判断哪些部分可以直接复用，哪些部分需要根据具体需求进行调整。

现在，你得到了一份“校内路线规划”项目的半成品（即 Project 文件夹），请你按照以下步骤阅读该项目

0.1 了解项目背景 [0%]

请阅读 README 文档，提取其中可能会用到的信息。

0.2 环境搭建与初步运行 [4%]

(a) **[截图]** 根据你所得到的信息和 Homework0 中教给大家的方法，配置该项目的运行环境。配置完成后，运行 `visualization.py`, 并将你运行得到的结果截图放在实验报告中。

0.3 整体结构分析 [6%]

你发现，这位项目原作者有点懒，他并没有在 README 里直接交代完整的文件结构，因此你需要根据文件的命名先简要猜测它们的作用和功能。

(b) **[配对]** 现在，请你阅读文件夹与文件的命名，将下列文件或文件夹匹配上对应的功能或作用:`(1)data; (2)grader.py; (3)graderUtil.py; (4)mapUtil.py; (5)submission.py; (6)util.py; (7)visualization.py`

- A. 是我需要补全的文件
- B. 里面一定有一些对我有用的类/函数
- C. 这是一个存储了地图的文件夹
- D. 处理地图时可能会用到这个文件
- E. 这是测试和打分用的
- F. 这是对测试和打分而言有用的文件
- G. 这是用来可视化的

0.4 详细代码阅读 [10%=2%*5]

我们先关注 util.py，看看其中有哪些是我们可能会用到的。

现在，请阅读类 State、SearchProblem、SearchAlgorithm、Heuristic（即 line5-line79）的代码及其注释，回答以下问题：

(c) **[多选]** 一个 `State` 类的实例（instance）中，`memory` 的数据类型可能是：

- A. None

- B. `str`
- C. `list`
- D. `dict`
- E. `tuple`

(d) **[单选]** `SearchProblem` 中，方法（method）`successorsAndCosts()` 的功能是：

- A. 判定是否成功解出该搜索问题，并返回搜索出来的状态序列及其代价（距离）
- B. 确定一个状态的所有可能的后继状态，并以列表形式返回

(e) **[多选]** 下列说法正确的是：

- A. 这一部分代码仅对类的接口做了声明，并不涉及具体实现
- B. `SearchProblem` 类中缺少与问题有关的相关信息，它的子类或者实例需要添加相关变量
- C. 假设 `a` 是 `SearchProblem` 的一个实例，`b` 是该问题对应的 `SearchAlgorithm` 的一个实例，且所有功能均已被我们实现，则可以用一句代码 `b.solve(a)` 求解出该搜索问题
- D. 类 `Heuristic` 中，方法 `evaluate()` 的功能是精确计算从当前状态到终止状态的代价

简要浏览 `util.py` 剩余部分的代码及其注释（即 line80-172），我们发现，好心且帅气的助教已经帮大家写好了 Dijkstra 算法的代码，大家后续直接调用即可，不需要自己实现了（相信大家在其他课程中已经写过多次）！现在，请根据这一段代码回答以下问题：

(f) **[多选]** 一个 `UniformCostSearch` 的实例包含以下哪些变量：

- A. `self.verbose`
- B. `self.actions`
- C. `self.pathCost`
- D. `self.numStatesExplored`
- E. `self.pastCosts`

(g) **[简答]** 我们发现，`UniformCostSearch` 类中有一个奇怪的变量，叫做 `verbose`。关于这个变量到底表示什么，作者并没有给出注释，相信大家也不想阅读后面的这几十行代码。请利用你最喜欢的 AI 模型，让它帮你回答这一点吧！（例如，你可以复制 line84-line138，询问 DeepSeek 这段代码中的 `verbose` 表示什么）。请以截图或者文字总结的形式将它的回答放在报告中。

相信大家现在已经掌握了阅读代码的基本方法，接下来，让我们正式开始我们的旅程吧！

1 问题 1：查找最短路径 [29%]

现在我们有一张由一组位置组成的校园地图。每个位置都有：

- 一个独特的编码，例如，6608996258（str），
- 该位置的（纬度、经度）坐标，例如，(31.4299866, 117.175519)，
- 一组描述位置类型的标签（tags），例如，amenity=coffee。

位置之间有一组连接（connection）；每个连接都有一个距离（以米为单位），且是双向的（即从 A 到 B 的距离为 100 米，那么从 B 到 A 的距离也是 100 米）。

运行 `python mapUtil.py > readableUSTCMap`，你将在 readableUSTCMap 这个文件中看到如图2所示的数据，里面显示了每个位置的编码、坐标、标签、邻居及其到邻居的距离。你可以从这个文件中选择你感兴趣的位置/标签。此外，你还可以在nominatim 中找到你感兴趣地点的经纬度，然后将其添加到 `data/USTC-landmarks.json` 中，从而制作你的个性化地图。

```
2706839901 (31.8387188,117.2630146): label=2706839901
  -> 3064296110 [distance = 90.53494123442582]
  -> 2705289534 [distance = 41.93390564229068]
  -> 10582890636 [distance = 28.420636834406395]
  -> 2706839903 [distance = 79.85677317437133]
2706839903 (31.8387652,117.2638582): label=2706839903 landmark=east_campus_library amenity=library
  -> 2894917873 [distance = 53.87837985772386]
  -> 2724738770 [distance = 72.59990133133641]
  -> 2706839901 [distance = 79.85677317437133]
2724738770 (31.8388128,117.2646247): label=2724738770
  -> 2706839903 [distance = 72.59990133133641]
  -> 2758031713 [distance = 50.43582472289227]
  -> 2879179244 [distance = 20.744718513929488]
```

图 2: readableUSTCMap

现在，你需要解决以下问题：给定一个特定的起点和一个标签，你需要规划从这个起点出发到最近的具有这个标签的地点的最短路径。

1.1 建模 [10%=6%+4%]

请回顾课堂中所讲的内容，尝试将最短路径问题归约到一般化的搜索问题中。

(a) **[代码]** 补全 `submission.py` 中的 `ShortestPathProblem`。你需要实现 `startState()`、`isEnd(state)` 和 `successorsAndCosts(state)`。[提示：完成本问的过程中，你可能发现你还需要阅读其它文件的部分代码，使用其他文件中有用的类/函数。以下快捷键 (适用于 PyCharm 或 VSCode) 或许对你有帮助：Ctrl+F，查找/替换；Ctrl+ 左键单击，跳转到接口/定义处]

(b) **[代码]** **[截图]** 由于搜索问题的算法已经被助教写好了，因此，你只需要创建一个搜索问题的实例就可以直接运行了！请在 `getUSTCShortestPathProblem` 中创建一个搜索问题的实例。然后运行 `python grader.py 1b-custom` 以生成 `path.json`，里面记录了搜索出的最短路径。生成后，运行 `python visualization.py --path-file path.json` 可以可视化你的结果。请将你的结果截图放在报告中。

1.2 算法 [19%=6%+5%+2%+6%]

相信大家在其它课程中已经代码实现过 Dijkstra 算法，因此在本次实验中我们并不需要大家再次把这个算法默写出来。但我们需要关注 Dijkstra 算法为什么是正确的，在什么情况下是正确的。下面，我们从算子的角度理解 Dijkstra 算法的正确性。

给定一个有 n 个点的简单有向图 $G(V,E)$，边 $(i,j) \in E$ 的权值用 w_{ij} 表示，权值满足 $w_{ij} \geq 0$，并假设如果 $(i,j) \notin E$，则 $w_{ij} = +\infty$，且定义 $w_{ii} = 0$。我们希望求解从某一源点 s 到任意汇点 u 的最短路径距离 $d_s(u)$（可能为 $+\infty$），定义源点 s 到自身的最短路径距离为 $d_s(s) = 0$。现在，我们使用 $d_s^{(k)}(u)$ 来估计 $d_s(u)$ 的上界，其中 k 是迭代更新的次数，$d_s^{(0)}(u)$ 被初始化为：

$$d_s^{(0)}(u) = \begin{cases} 0, & \text{if } u = s \\ +\infty, & \text{otherwise} \end{cases}$$

定义 $Dijkstra$ 算子 $\mathcal{D}_k$ 为作用在函数 $d_s^{(k)}(u)$ 上的算子，满足：

$$v_k = \begin{cases} s, & \text{if } k = 0 \\ argmin_{v \in V - \{v_0,\dots,v_{k-1}\}} d_s^{(k)}(v), & \text{otherwise} \end{cases}$$

$$d_s^{(k+1)}(u) = \mathcal{D}_k \circ d_s^{(k)}(u) = min_{v \in \{v_0,\dots,v_k\}} \{d_s^{(k)}(v) + w_{vu}\}$$

由 Dijkstra 算子，我们得到了一个序列 $v_0, \dots, v_{n-1}$

(c) **[简答]** 请证明 $\forall k, u,\ d_s(u) \leq d_s^{(k+1)}(u) \leq d_s^{(k)}(u)$。

(d) **[简答]** 请证明 $\forall k,\ d_s^{(k)}(v_k) = d_s(v_k)$, 从而说明，算子执行 n-1 次后，任意点到源点的最短路径距离均能被发掘，以此证明 Dijkstra 算法的正确性。

在东区图书馆旁、一教门口都有一栋神秘的建筑[1]。传说中，它是通往异世界的入口，某种特殊的虫洞连接了这 2 个位置，使得二者之间的路径距离是负值。若传说是真的，我们将负边权的边加入到我们的地图中，你设计的程序还能正确运行吗?

图 3: 一教门口

图 4: 东图旁

(e) **[判断]**Dijkstra 算法能否直接在存在负边权的图上运行。

[1]为了你的人身安全，请勿擅自闯入这些建筑

(f) **[简答]** 给分与上一问独立

若你认为可以，(1) 请说明这一改动对算法的正确性没有影响；(2) 请巧妙地利用 Dijkstra 算法，使其在多项式时间内解决“寻找图中的哈密顿路”这一问题（用自然语言描述或者伪代码描述）。

若你认为不行，(1) 请结合你 1(d) 的证明说明理由；(2) 请问你可以怎样改进，使得存在负边权边时，程序仍然能正确运行（用自然语言描述或者伪代码描述）。

2 问题 2：查找带无序途径点的最短路径 [17%]

让我们接触一个更强大的功能：无序途径点（unordered waypoints）！在打车软件中，你可以指定一个路径必须经过的序列。例如，从 A 点到 X 点再到 Y 点到 B 点，其中 X、Y 是“途径点”。但是，在这里我们要考虑途径点无序的情况：A → X → Y → B 和 A → Y → X → B 都是允许的。在我们的程序中，X、Y 和 B 分别由一个标签（tag）指定。这是一个非常有用的功能，比如在西区上了一上午课后，你可能正在回家宿舍的路上，但需要去菜鸟驿站拿包裹，去肯德基吃饭，再去书店买一些笔记本，获得一条短且快速的路径去途径所有这些地方非常方便。

2.1 建模 [10%=6%+4%]

(a) **[代码]** 定义 `WaypointsShortestPathProblem`，使得给定一个 `startLocation`、一组 `waypointTags` 和一个 `endTag`，可以找到路过所有途径点的最短路径。注意，单个位置可以用于满足多个标签。和问题 1 一样，你需要实现 `startState()`、`isEnd(state)` 和 `successorsAndCosts(state)`。有很多方法可以实现这个搜索问题，所以你应该仔细考虑如何设计你的状态空间，以优化搜索效率。

(b) **[代码]** **[截图]** 选择一个起始位置、一组途径点标签和一个结束标签，并编写 `getUSTCWaypointsShortestPathProblem()` 以创建一个搜索问题的实例。与问题 1b 类似，运行 `python grader.py 2b-custom` 以生成 `path.json`，然后进行可视化并在浏览器中打开。本题中请选用与问题 1(b) 相同的一组 `startLocation` 和 `endTag`，并将截图与 1(b) 对比。

2.2 算法 [7%=2%+5%]

我们在搜索 2(b) 中创建的实例时，搜索算法仍然用的是助教写的 Dijkstra 算法，你的同行小李对此提出了质疑。请解决他提出的问题：

(c) **[判断]** 考虑如下场景：起点为 A，需经过无序途径点集合 X,Y,Z，终点为 B。假设在搜索过程中，某条路径已到达终点 B，但尚未经过途径点 X（即该路径顺序可能为 A→Y→Z→B）。此时你的程序会将此路径判定为有效终点状态吗?

如果会，请你重新检查你问题 2(a) 的答案；

如果不会（即算法会继续探索其他可能路径），请回答以下问题。

(d) **[简答]** 针对上述场景，小李提出质疑：如果算法允许路径先到达 B（未完成所有途径点），之后又继续探索其他路径（如从 B 出发访问 X 再返回 B），这种回溯行为将违反 1(c) 中 Dijkstra 算法 $d_s(u) \le d_s^{(k)}(u)$ 的基本性质，从而无法保证算法的正确性（即无法保证最终搜索出的路径是带无序途径点的最短路径）。

请分析小李的质疑是否成立：

若成立，说明如何改进算法使得其可以在本题情形下正确运行（使用自然语言或者伪代码）；

若不成立，请说服小李，并说明小李的误区在哪里。

问题 3：使用 A* 加快搜索速度 [32%]

尽管在刚才的测试中，我们的程序各方面表现都不错。但假设我们的地图扩大，例如变成合肥市地图，甚至中国地图，UCS 时间性能的劣势就会凸显出来，这时候，我们可以用 A* 来加快我们的搜索速度。

2.3 将 UCS 转化为 A*[4%]

(a) **[代码]** 请你回顾课堂中所讲的 A* 算法。我们注意到，只需修改一下每条边的代价，便可将一个 A* 搜索转变为一个 UCS 搜索。现在，请你完成 `aStarReduction`, 利用一个已知的 heuristic，将一个 A* 搜索问题转化成 UCS 搜索问题。

2.4 实现启发式函数 [18%=3%+6%+3%+6%]

做完上一问的改动之后，对于新的搜索问题，我们仍然使用 Dijkstra 算法来找到最短路径。但是我们注意到，A* 算法的正确性对启发式函数有一定要求。现在，我们需要为对本次作业的问题 1、问题 2 设计一个合理的启发式函数，使得 A* 算法正确运行。

对于问题 1，我们使用每个结点到终点的直线距离作为启发式函数。

(b) **[简答]** 说明该启发式函数是一致的 (consistent)，从而保证了 A* 算法的正确性。

(c) **[代码]** 实现 `StraightLineHeuristic` 类。（提示：你可以在该类中新加一些变量，从而减少调用 evaluate 时的时间复杂度）

对于问题 2，我们使用不带途径点的最短路径长度作为启发式函数。

(d) **[简答]** 说明该启发式函数是一致的 (consistent)，从而保证了 A* 算法的正确性。

(e) **[代码]** 实现 `NoWaypointsHeuristic` 类。（提示：你可以在该类中新加一些变量，从而减少调用 evaluate 时的时间复杂度）

2.5 利用合肥市地图对比运行时间 [10%=4%+6%]

(f) **[代码]** **[简答]** 实现 `getHefeiShortestPathProblem` 函数和 `getHefeiShortestPathProblem_with Heuristic` 函数，它们分别会返回一个 UCS 搜索问题和一个变成 UCS 的 A* 搜索问题。运行 `python grader.py 3f-without_Heuristic` 和 `python grader.py 3f-with_Heuristic` 对比它们的运行时间，并把运行时间附在你的报告中。

(g) **[简答]** 如果在合肥市地图上运行，你的程序还存在什么缺陷？你有办法进一步减少运行时间吗？给出一种可能的解决方案（用尽可能详细的文字描述）

作业反馈

体验反馈 [2%]

你可以写下任何反馈，包括但不限于以下几个方面：课堂、作业、助教工作等等。

(a) [**必做**] 你在本次作业花费的时间大概是?

(b) [**选做**] 你可以随心吐槽课程不足的方面，或者给出合理的建议。若没有想法，直接忽略本小题即可。你的回答不会影响给分，若仍然担心，你可以通过该匿名问卷回答