求最短路径的 Dijkstra 形式模型与算法

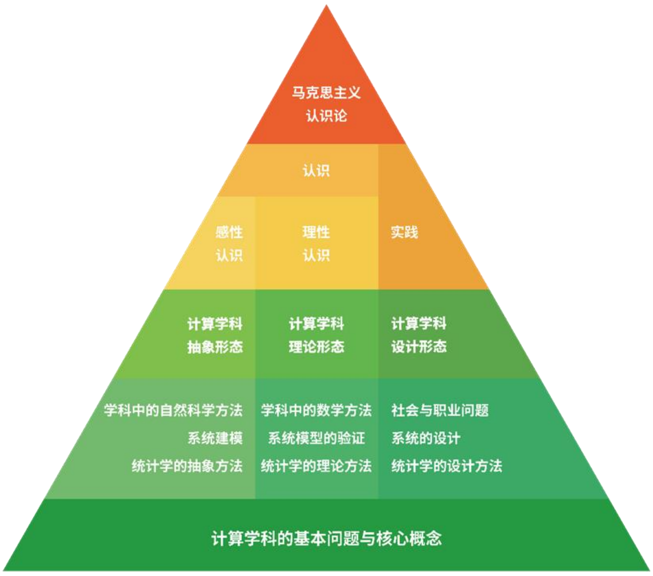
计算机课程思政虚拟教研室 科学思维样例 3

一、教学目标

本案例能够置于 Bloom 分类法知识维度的“元认知知识”位置，学生学习后能够达到 Bloom 分类法认知过程维度的“评估”层次。

二、本案例课程思政的关注点

1.本案例内容在计算学科课程思政总体框架中的位置：计算学科抽象、理论和设计三形态，计算学科的基本问题与核心概念。



2.科学思维可拆分为可衡量、可检验的抽象、理论和设计三形态。其中，求最短路径的 Dijkstra 形式模型与算法的抽象形态包括形式模型、算法伪代码描述 。理论形态包括 Dijkstra 的理论形态包括算法正确性证明和时间复杂度分析。设计形态包括 Dijkstra 的设计形态包括 Dijkstra 算法举例说明、python 代码实现。

3.在本案例中，要求与 CC2020 中的“主动性”品行，以及 CS2023“算法基础”中的“严谨性、创造性、坚持不懈”品行对齐，并与该案例绑定在一起进行可操作性解释。

三、本案例中的抽象、理论和设计三形态

1. 求最短路径的 Dijkstra 算法的问题描述

给定一个带权有向图 $G=(V,E,W)$，其中 V 表示图中顶点构成的集合，$E\subseteq \mathrm{V}\times\mathrm{V}$ 为图中的所有边构成的集合，权重函数 $W:E\rightarrow R^{+}$，映射每条边到非负实数值的权重上。图中的一条路径是顶点的交替序列 $e=v_0v_1v_2\ldots v_n$，满足 $(v_0,v_1),(v_1,v_2),\ldots,(v_{n-1},v_n)\in E$，也称 e 为从顶点 v_0 到顶点 v_n 的一条路径。路径的权重是构成该路径的所有边上的权重之和。给定顶

点 $s,x \in V$ ，从顶点 s 到顶点 x 的最短路径为其所有路径中权重最小的一条路径。该路径的权重为其最短路径权重。

为了给出最短路径上的顶点，对于每个顶点设置一个前驱顶点 pre，该前驱顶点也可能为空（用-1 表示）。从顶点 x 开始的前驱结点链翻转过来，就是从顶点 s 到顶点 x 的一条最短路径。

单源最短路径问题是一种常见的最短路径问题，即给定的一个带权图和一个顶点称为源点，求得源点到其它顶点的最短路径。在图中的每条边的权重非负时，这个问题可以通过使用一种称为 Dijkstra 算法的贪心策略来求解。

为展示 Dijkstra 算法在抽象、理论和设计三个过程中的具体内容，设计了求最短路径问题的 Dijkstra 形式模型与算法步骤，并证明了算法的正确性和分析了算法的时间复杂度，最后给出了算法实现。使得该问题的解决过程简单易读，读者可以从问题的形式模型出发，通过一步步推演求解问题。

2. Dijkstra 算法的抽象形态

Dijkstra 算法的抽象形态，包括 Dijkstra 形式模型、算法描述。

（1）Dijkstra 形式模型

Dijkstra 形式模型是一个六元组 $Dijkstra=(G,s,S,pre,dis,P)$ 其中：

①$G=(V,E,W)$ 表示给定的边上权重非负的带权有向图，其中 V 为顶点集，$E \subseteq V \times V$ 为边集，权值函数 $W: E \to R^+$，对于每条边 $(a,b) \in E$，$W(a,b)$ 表示边上的权重；$s \in V$ 表示给定的源点；

②
$$\begin{aligned}&S \subseteq V,\\&pre: V \to V \cup \{-1\},\\&dis: V \to R^+,\\&P: V \to V^+;\end{aligned}$$

③
$$S=\{s\};$$
$$\forall i \in V, dis(i)=\begin{cases}W(s,i), & (s,i)\in E\\ \infty, & \text{否则}\end{cases},$$
$$pre(i)=\begin{cases}\text{s}, & (s,i)\in E\\ -1, & \text{否则}\end{cases};$$

④$\exists x \in V-S \text{且} dis(x)=\min\limits_{k \in V-S} dis(k)$；

⑤$S=S\cup\{x\}$；

⑥
$$\forall y \in V-S \text{且} (x,y)\in E,$$
$$dis(y)=\begin{cases}dis(x)+W(x,y), & dis(y)\ge dis(x)+W(x,y)\\ dis(y), & \text{否则}\end{cases}$$

$$pre(y)=\begin{cases} x, & dis(y)\ge dis(x)+W(x,y) \\ pre(y), & \text{否则} \end{cases}$$
;

⑦当 $S=V$ 时，对于每个顶点 $i\in V$ ，从源点 s 到顶点 i 的最短路径权重为 $dis(i)$，从源点 s 到顶点 i 最短路径为 $P(i)=\begin{cases} \text{空}, & pre(i)=\text{-1} \\ s, & pre(i)=\text{s} \\ P(pre(i))i, & \text{否则} \end{cases}$ 。

（2）Dijkstra 算法伪代码描述

Algorithm 1 Dijkstra

Input： 一个边上的权值非负的带权有向图 $G=(V,E,W)$，一个源点 s 。

Output： dis,P 分别表示 G 中顶点 s 到其它顶点的最短路径权重和最短路径。

$S\leftarrow\{s\}$

for $i\in V$ **do**

    if $(s,i)\in E$ **then**

        $dis(i)\leftarrow W(s,i)$

        $pre(i)\leftarrow s$

    else

        $pre(i)\leftarrow\text{-1}$

while $V-S\ne\varnothing$

$\exists x\in V-S$ 且 $dis(x)=\min\limits_{k\in V-S} dis(k)$

    $S\leftarrow S\cup\{x\}$

for $\forall y\in V-S$ 且 $(x,y)\in E$ **do**

if $dis(y)\ge \mathrm{dis}(x)+W(x,y)$ **then**

$dis(y)=\mathrm{dis}(x)+W(x,y)$

            $pre(y)=x$

for $i\in V$ **do**

    $P(i)=\mathrm{FindMinPath}(pre,i)$

return dis,P

Algorithm 2 $\mathrm{FindMinPath}(pre,i)$

---

if $pre(i)=-1$ **then**

```
    return 空
if pre(i) = s  then
    return s
else
return   FindMinPath(pre, pre(i))i
```

3. Dijkstra 算法的理论形态

Dijkstra 的理论形态包括算法正确性证明和时间复杂度分析。

（1）Dijkstra 算法的正确性

引理 1 Dijkstra 算法中，如果在顶点 x 未移动到 S 前，对于所有在 S 中顶点 i，从源点 s 到 i 最短路径权重为 $dis(i)$，那么在顶点 x 移动到 S 后，从源点 s 到顶点 x 最短路径权重也为 $dis(x)$。

证明 设 $\forall i \in V$，从源点 s 到顶点 i（真实）最短路径权重为 $\delta(i)$。如果在顶点 x 未移动到 S 前，对于所有在 S 中顶点 i，$\delta(i)=dis(i)$，我们利用采用反证法证明在顶点 x 移动到 S 后 $\delta(x)=dis(x)$。假设 Dijkstra 算法将顶点 x 移动到 S 后，$\delta(x)\neq dis(x)$。由于 $dis(x)$ 作为 $\delta(x)$ 的上界，故 $dis(x)>\delta(x)$，应存在一条权重为 $\delta(x)$ 的从源点 s 到顶点 x 的最短路径，如图 1 所示，不妨设其为 $s,\ldots,m,n,\ldots x$，其中边（m,n）横跨 S 和 $V-S$，即 $m\in S, n\in V-S$。由于 $m\in S$ 可得 $\delta(m)=dis(m)$。而 $s,\ldots,m,n$ 是最短路径 $s,\ldots,m,n,\ldots x$ 的子路径，且最短路径的子路径也是最短路径，可以得到 $\delta(n)=\delta(m)+W(m,n)=dis(m)+W(m,n)$

通过算法第 12 行可知，算法对顶点 m 出发的所有边（包括边（m,n））已进行过更新操作，故 $dis(n)\le dis(m)+W(m,n)$。合并上述公式，可得 $dis(n)\le\delta(n)$，由于 $dis(n)$ 作为 $\delta(n)$ 的上界，故有 $dis(n)=\delta(n)$。最短路径 $s,\ldots,m,n,\ldots x$ 中，n 出现在 x 之前，故 $dis(x)>\delta(x)\ge\delta(n)=dis(n)$。从而有 $dis(x)>dis(n)$，$i\neq y$，顶点 x 不应为下一个选择的顶点，这与算法第 10 行矛盾。

假设不成立。

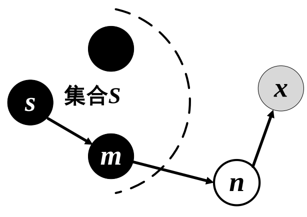


图 1 从源点 s 到顶点 x 的最短路径

定理 1 给定一个边上的权值非负的带权有向图 $G=(V,E,W)$ 和一个源点 s，Dijkstra 算法能找到从源点 s 到其它顶点的最短路径权重。

证明 证明循环不变式：

$\forall i \in S$，从源点 s 到顶点 i 的最短路径权重为 $dis(i)$。

初始：在第一迭代开始前，即当 $|S|=1$ 时，初始化 $S=\{s\}$，S 中没有源点外其它顶点，显然循环不变式成立。

保持：假设在第 k 次迭代后，即当 $|S|=k$ 时循环不变式成立。那么在顶点 x 移动到 S 后 $|S|=k+1$ 时，由假设和引理 1 可知，从源点 s 到顶点 x 的最短路径权重也为 $dis(x)$，从而移动后，对于 $\forall i \in S$，从源点 s 到顶点 i 的最短路径权重为 $dis(i)$。故第 $k+1$ 次迭代后循环不变式成立。

终止：Dijkstra 算法每次迭代后 S 中都会增加一个顶点，这样迭代 $|V|-1$ 次后，$V-S=\varnothing$（即 $S=V$）时终止，故在第 $|V|-1$ 次迭代后，此时 $S=V$ 循环不变式成立，则 $\forall i \in V$，从源点 s 到顶点 i 的最短路径权重为 $dis(i)$。因此算法正确。

（2）Dijkstra 算法的时间复杂度

假设输入图由邻接表表示，边上的权重存放在邻接表的边结点中。对于集合 S，用数组 $S[1..n]$ 表示。初始时，$S[1]=1$，并且对于所有的 i，$2 \le i \le n$，$S[i]=0$，运算 $S \leftarrow S \cup \{j\}$ 时，将 $S[j]$ 的值置 1 来实现。

对 Dijkstra 算法进行初始化，花费时间为 $O(|V|)$，找到最短路径权重最小的顶点花费时间为 $O(|V|)$，总的执行 for 循环花费时间 $O(|V|^2)$。由于是输入图由邻接表表示，算法需遍历其它顶点所有邻接的边，所以算法更新操作花费时间为 $O(|E|)$。根据以上得出算法的时间复杂度为 $O(|V|^2+|E|)$。

4. Dijkstra 算法的设计形态

Dijkstra 的设计形态包括 Dijkstra 算法举例说明、c++代码实现。

（1） 举例

给定一个边上权值非负的带权有向图 G 和源点 s，图 2 所示。

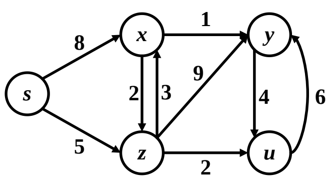


图 2 带权有向图 G

设 $Dijksta=(G,s,S,pre,dis,P)$ 是一个 Dijkstra 形式模型。由图我们可以得到 $G=(V,E,W)$ 和源点 s。其中

$V=\{s,x,y,z,u\}$,

$E=\{<s,x>,<s,z>,<x,y>,<x,z>,<y,u>,<z,x>,<z,y>,<z,u>,<u,y>\}$,

$W=\{<s,x,8>,<s,z,5>,<x,y,1>,<x,z,2>,$
$<y,u,4>,<z,x,3>,<z,y,9>,<z,u,2>,$
$<u,y,6>\}$。

利用 Dijkstra 算法求源点 s 到其它每个顶点的最短路径权重和最短路径过程如下。

1）初始化

初始化集合 S： $S=\{s\}$；

初始化 V 上的函数 dis, pre：

$<s,x>,<s,z>\in E$，从而 $dis(x)=W(s,x)=8, dis(z)=W(s,z)=5,$ $pre(x)=pre(z)=s$；

$<s,s>,<s,y>,<s,u>\notin E$，从而 $dis(s)=dis(y)=\mathrm{dis}(u)=\infty$，$pre(s)=pre(y)=pre(u)=-1$，如图 3 所示。

V	*s*	*x*	*y*	*z*	*u*
S	1	0	0	0	0
pre	-1	*s*	-1	*s*	-1
dis	0	8	∞	5	∞

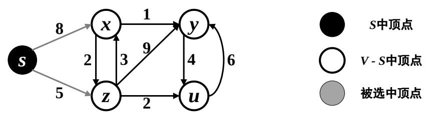


图 3 初始化

2）迭代

选择、移动、更新操作迭代 $|V|-1=4$ 次。

① 第一次迭代，如图 4 所示。

选择： $z\in V-S$ 且 $dis(z)=\min\{dis(x),dis(y),dis(z),dis(u)\}=5$，故选择顶点 z。

移动：将顶点 z 移动到 S 中，令 $S=S\cup\{z\}=\{s,z\}$。

更新：顶点 $y\in V-S$ 且 $<z,y>\in E$，$dis(y)\ge dis(z)+W(z,y)$，故更新 $dis(y)=14$，$pre(y)=z$；顶点 $u\in V-S$ 且 $<z,u>\in E$，$dis(u)\ge dis(z)+W(z,u)$，故更新 $dis(u)=7$，$pre(u)=z$。

V	*s*	*x*	*y*	*z*	*u*
S	1	0	0	0	0
pre	-1	*s*	*z*	*s*	*z*
dis	0	8	14	5	7

∞→14

∞→7

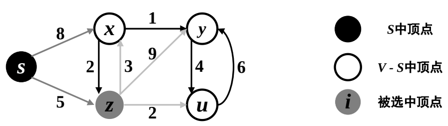


图 4 确定从源点到顶点 z 的最短路径

② 第二次迭代，如图 5 所示。

选择： $u \in V-S$ 且 $dis(u)=\min\{dis(x),dis(y),dis(u)\}=7$ ，故选择顶点 u 。

移动： 将顶点 u 移动到 S 中，令 $S=S\cup\{u\}=\{s,z,u\}$；

更新： 顶点 $y\in V-S$ 且 $<u,y>\in E$， $dis(y)\geq dis(u)+W(u,y)$，故更新 $dis(y)=13$，$pre(y)=u$ 。

V	s	x	y	z	u
S	1	0	0	1	0
pre	-1	s	u	s	z
dis	0	8	13	5	7

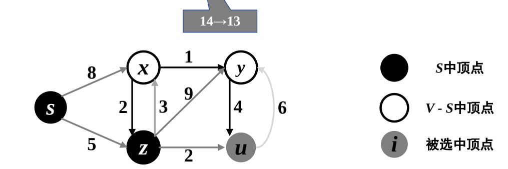


图 5 确定从源点到顶点 u 的最短路径

③ 第三次迭代，如图 6 所示。

选择： $x\in V-S$ 且 $dis(x)=\min\{dis(x),dis(y)\}=8$，故选择顶点 x；

移动： 将顶点 x 移动到 S 中，令 $S=S\cup\{x\}=\{s,x,z,u\}$；

更新： 顶点 $y\in V-S$ 且 $<x,y>\in E$， $dis(y)\geq dis(x)+W(x,y)$，故更新 $dis(y)=9$，$pre(y)=x$ 。

V	s	x	y	z	u
S	1	0	0	1	1
pre	-1	s	x	s	z
dis	0	8	9	5	7

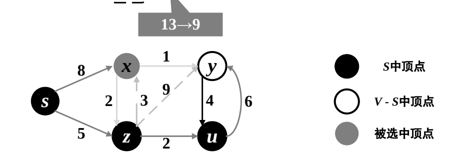


图 6 确定从源点到顶点 x 的最短路径

④ 第四次迭代，如图 7 所示。

选择： $y\in V-S$ 且 $dis(y)=\min\{dis(y)\}=9$，故选择顶点 y 。

移动： 将顶点 y 移动到 S 中，令 $S=S\cup\{y\}=\{s,x,y,z,u\}$；

更新： $V-S=\varnothing$，故不用更新。

V	s	x	y	z	u
S	1	1	0	1	1
pre	-1	s	x	s	z
dis	0	8	9	5	7

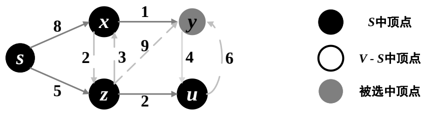


图 7 确定从源点到顶点 y 的最短路径

3）求解每个顶点的最短路径

每次迭代后 S 中都会增加一个顶点，这样迭代 $|V|-1=4$ 次后，$V-S=\varnothing$ 即 $S=V$ 。由形式模型可知，当 $S=V$ 时，对于每个顶点 $i\in V$ ，从源点 s 到顶点 i 的最短路径权重为 $dis(i)$，如图 8 所示。所以：

对于顶点 x，$dis(x)=8$；

对于顶点 y，$dis(y)=9$；

对于顶点 z，$dis(z)=5$；

对于顶点 u，$dis(u)=7$ 。

递归地求源点 s 到其他各顶点的最短路径，如图 8 所示，其具体过程如下：

对于顶点 x，$pre(x)=s$，从而 $P(x)=P(pre(x))x=P(s)x=sx$；

对于顶点 y，$pre(y)=x$，从而 $P(y)=P(pre(y))y=P(x)y=sxy$；

对于顶点 z，$pre(z)=s$，从而 $P(z)=P(pre(z))z=P(s)z=sz$；

对于顶点 u，$pre(u)=z$，从而 $P(u)=P(pre(u))u=P(z)u=szu$ 。

V	s	x	y	z	u
dis	0	8	9	5	7

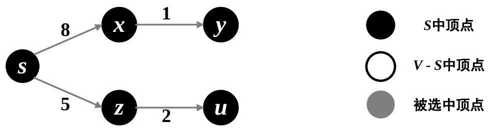


图 8 确定从源点到每个顶点的最短路径

（2） Python 实现

根据 Dijkstra 算法形式模型，使用 python 实现该算法。图 9 给出了 python 算法代码和以图 2 中数据运行的结果。

```python
def dijkstra(G, s):
    V = set(G.keys())  # Set of all vertices in G
    dis = {v: float('infinity') for v in V}
    pre = {v: -1 for v in V}
    dis[s] = 0
    S = set()
    while S != V:
        u = min((V - S), key=lambda x: dis[x])
        S.add(u)
        for v in G[u]:
            if v not in S:
                alt = dis[u] + G[u][v]
                if alt < dis[v]:
                    dis[v] = alt
                    pre[v] = u
    return dis, pre
1 个用法

def find_min_path(pre, i):
    path = []
    while i != -1:
        path.append(i)
        i = pre[i]
    return path[::-1]
```

```
输入顶点的数量：5
输入边的条数：9
输入顶点的名称，用空格分隔：s x y z u
输入所有边和权重（格式：a b 8，用逗号分隔不同边）：s x 8,s z 5,x y 1,x z 2,y u 4,z x 3,z y 9,z u 2,u y 6
图的邻接表：
s -> {'x': 8, 'z': 5}
x -> {'y': 1, 'z': 2}
y -> {'u': 4}
z -> {'x': 3, 'y': 9, 'u': 2}
u -> {'y': 6}
输入源点名称：s

源点s到顶点、最短路径权重、最短路径：
顶点          最短路径权重          最短路径
s           0               s
x           8               s -> x
y           9               s -> x -> y
z           5               s -> z
u           7               s -> z -> u
```

图 9 Python 算法代码和运行结果

四、专业品行

在探讨 Dijkstra 算法的教学目标、课程思政的关注点以及抽象、理论、设计三形态内容时，明显地体现了算法学习不仅仅涉及技术知识的掌握，还关联到专业品行的培养。根据 CC2020 中的“主动性”品行，以及 CS2023 中“算法基础”的“创造性、严谨性、坚持不懈”的品行，对 Dijkstra 算法的学习和实践中的专业品行进行可操作性解释具有重要价值。

1.主动性（Proactivity）

在 Dijkstra 算法的学习过程中，主动性表现为积极探索算法的核心原理和应用领域。理解算法的工作步骤和逻辑是基础，并进一步探索如何优化算法性能（例如，通过采用优先队列改进时间复杂度）。主动性鼓励学习者深化理解，并在解决实际问题时有效应用算法知识。

2.创造性（Inventiveness）

Dijkstra 算法提供了广泛的创新空间，激发学习者在多个领域内寻找新的应用场景，例如社交网络分析或推荐系统等。这种跨学科应用不仅展现了算法知识的灵活性，也体现了创造性在解决问题中的重要作用。

3.严谨性（Meticulousness）

Dijkstra 算法的正确性证明和时间复杂度分析强调学习过程中的严谨性。通过证明算法

的正确性和分析其性能，能使学习者更深入理解算法。这种严谨的思维方式在设计形态的编程实践中同样重要，要求考虑各种边界条件和潜在的优化策略，确保算法实现的准确与高效。

4.坚持不懈（Persistence）

学习和应用 Dijkstra 算法可能面临多种挑战，如性能优化、处理特殊情况、以及算法在实际问题中的应用等。面对这些挑战，持之以恒地寻找解决方案是必需的。这种坚持不懈的态度对于任何学科的学习都是必需的，尤其在计算机科学领域。

通过结合 Dijkstra 算法案例与专业品行的教学，不仅加深学生对算法和计算的理解，更重要的是培养他们将知识和品行应用于社会实践，面对复杂问题时，并以正确的态度和方法解决问题解决实际问题，为社会进步贡献力量。

五、激励、唤醒和鼓励同学们向上的途径

在学习 Dijkstra 算法案例后，鼓励学生继续学习其他相关的内容。比如：Dijkstra 算法的时间复杂性较高，可以通过使用优先队列等数据结构来优化算法，提高其执行效率；当图中存在边权重为负数的情况时，Dijkstra 算法不再适用，需要使用其他算法，如 Bellman-Ford 算法；图中存在多条具有相同最短路径权重的路径情况，Dijkstra 算法可能无法给出唯一的最短路径结果，学生还可以进一步讨论如何处理这种多解的情况。通过完成该案例，学生能够掌握使用图结构的能力，理解图结构如何转换为符号的形式描述，并具备运用计算模型来解决问题的能力。

Dijkstra 算法案例唤醒同学们对形式模型价值的认识，深刻体会通过采用“第一性原理”来简化和讨论计算问题的有效性，进而鼓励他们将这种思维方式应用于各学科，以降低问题的复杂性，提高解决问题的效率和质量。

六、习题

1. 根据 Dijkstra 形式模型 \(G = (V, E, w, s, S, pre, dis)\) 和所给的算法伪代码描述，解释如何通过形式模型初始化算法的输入，并简述算法伪代码的执行流程。
2. 引用上述中的“引理 1”和“定理 1”，请尝试解释为什么 Dijkstra 算法能保证找到从源点到其它顶点的最短路径权重。你的解释应包括对“引理 1”的理解以及它如何支持“定理 1”的证明。
3. 考虑一个带权有向图 G，顶点集 V = {A, B, C, D, E} 和边集 E = {(A, B, 4), (A, C, 2), (B, C, 5), (B, D, 10), (C, D, 3), (D, E, 1), (C, E, 4)}，其中每个元组 (u, v, w) 表示从顶点 (u) 到顶点 (v) 的边，权重为 w，且 A 是源点。使用 Dijkstra 算法计算从源点 A 到所有其他顶点的最短路径及其权重，并列出每个顶点的前驱顶点 pre。请展示算法的每一步更新过程。

4. 基于上述关于 Dijkstra 算法时间复杂度的分析，解释为什么使用邻接表表示的图和优先队列作为集合 Q 的数据结构可以提高算法的效率。请详细说明这种数据结构选择对算法时间复杂度的影响，并讨论在不同类型的图（例如稀疏图和密集图）中的表现差异。

参考文献

[1] 陈国良. 计算机课程思政虚拟教研室文化建设 [J]. 计算机教育, 2023(11):1-2.

[2] 董荣胜，古天龙，殷建平. 计算学科课程思政教学指南 [J]. 计算机教育, 2024(01): 7-15.

[3] Thomas H. Cormen 等著, 殷建平等译. 算法导论 (第 3 版) [M]. 北京：机械工业出版社, 2012.

[4] 董荣胜.计算机科学导论—思想与方法（第 4 版）[M].北京：高等教育出版社，2024.