### 原理概述：

1. 初始化：

- 为每个节点维护两个信息：从起始节点到该节点的当前已知最短路径的长度（距离）和该最短路径上的上一个节点。

- 将起始节点的距离设置为 0，其他节点的距离设置为无穷大（或一个非常大的值）。

2. 贪心选择：

- 在每一步中，选择一个未被标记的节点，它的已知最短路径是目前已知的最小值。

- 初始时选择起始节点。

3. 松弛操作：

- 对于选定的节点，计算从起始节点经过该节点到其他邻接节点的路径长度。

- 如果通过该节点的路径比已知的路径短，更新邻接节点的距离和路径信息。

- 这一步通过贪心的方式逐渐扩展已知最短路径的范围。

4. 标记节点：

- 将选定的节点标记为已访问，确保不会再次考虑它。

5. 重复步骤：

- 重复以上步骤，直到所有节点都被标记为已访问，或者所有标记的节点的距离都是无穷大。

算法的基本原理如下：

1. 初始化：

- 每个节点初始化自己到其他节点的距离矢量，通常将自身到自身的距离设置为0，到其他节点的距离初始化为无穷大。

- 每个节点还会记录到其他节点的下一跳节点的信息，即在通往目标节点的最短路径上的下一个节点。

2. 信息交换：

- 节点周期性地将其距离矢量发送给邻居节点。

- 邻居节点接收到距离矢量后，更新自己的距离矢量表，并将更新后的信息传播给它的邻居。

3. 更新距离：

- 当节点接收到邻居节点的距离矢量时，它会比较接收到的距离矢量与自身已有的距离矢量。

- 如果接收到的距离矢量提供了更短的路径，则更新自身的距离矢量，并将下一跳节点更新为发送节点。

4. 迭代：

- 算法通过上述信息交换和距离更新的过程进行迭代，直到网络中的所有节点的距离矢量收敛到最小值。

- 由于节点的距离矢量不断更新，网络中的每个节点都会逐渐了解到达其他节点的最短路径。

5. 路径获取：

- 一旦算法收敛，节点就可以使用距离矢量表来确定到达网络中任何其他节点的最短路径。

- 通过距离矢量表中的下一跳信息，节点可以沿着最短路径转发数据包。

实验目的总结：

通过完成这个实验，学生将达到以下目标：

1. 实现算法：学会使用Python实现Dijkstra算法和距离矢量算法。

2. 理解网络路由：理解Dijkstra算法和距离矢量算法在计算网络节点之间最短路径和距离时的应用。

1. 解决网络问题：能够处理给定网络拓扑结构和指定节点的输入，输出该节点到其他节点的距离列表和路径。

实验内容：  
  
任务一总结：

\*\*实验目的：\*\*

1. \*\*算法实现：\*\* 通过Python实现Dijkstra算法。

2. \*\*算法应用：\*\* 理解Dijkstra算法在计算网络节点之间最短路径和距离时的应用。

3. \*\*网络问题解决：\*\* 能够处理给定网络拓扑结构和指定节点的输入，输出该节点到其他节点的距离列表和路径。

\*\*实验步骤和代码解析：\*\*

1. \*\*初始化：\*\*

- 使用优先队列（最小堆）加速查找最短路径。

- 初始化距离字典和路径列表。

2. \*\*Dijkstra算法主体：\*\*

- 通过优先队列实现贪心选择，选择当前已知最短路径的节点。

- 对选定节点的邻居进行松弛操作，更新距离和路径。

- 使用堆结构维护节点，确保每次选择距离最小的节点进行松弛。

3. \*\*路径获取：\*\*

- 通过递归方式获取路径。

- 输出每个节点到其他节点的最短路径和距离。

4. \*\*应用：\*\*

- 针对给定的网络拓扑结构和指定节点，运行Dijkstra算法。

- 输出指定节点到其他节点的距离列表和路径。

- 重复以上步骤，覆盖所有节点。

\*\*输出结果：\*\*

- 距离列表和路径，以及各节点到其他节点的具体距离。

- 结果按节点分组，清晰展示每个节点的最短路径和距离。

任务二总结：

\*\*实验目的：\*\*

1. \*\*算法实现：\*\* 通过Python实现距离矢量算法。

2. \*\*算法应用：\*\* 理解距离矢量算法在计算网络节点之间最短路径和距离时的应用。

3. \*\*网络问题解决：\*\* 能够处理给定网络拓扑结构和指定节点的输入，输出该节点到其他节点的距离列表和路径。

\*\*实验步骤和代码解析：\*\*

1. \*\*初始化：\*\*

- 初始化距离表和下一跳表。

- 利用给定的网络拓扑结构初始化距离和下一跳信息。

2. \*\*松弛操作：\*\*

- 对于每个节点，通过迭代计算和更新距离表和下一跳表。

- 松弛操作检查是否存在更短的路径，如果存在则更新相应信息。

3. \*\*距离矢量算法主体：\*\*

- 主循环迭代次数为节点数目，每次迭代对所有节点进行松弛操作。

4. \*\*路径获取：\*\*

- 通过递归方式获取路径。

- 输出每个节点到其他节点的最短路径和距离。

5. \*\*应用：\*\*

- 针对给定的网络拓扑结构和指定节点，运行距离矢量算法。

- 输出指定节点到其他节点的距离列表和路径。

- 重复以上步骤，覆盖所有节点。

\*\*输出结果：\*\*

- 距离列表和路径，以及各节点到其他节点的具体距离。

- 结果按节点分组，清晰展示每个节点的最短路径和距离。

\*\*总结：\*\*

- 通过该实验，学生将掌握距离矢量算法的实现方式，理解其在网络路由中的应用，以及解决网络问题的能力。

简化后的：

\*\*实验简述：\*\*

1. \*\*初始化：\*\*

- 利用优先队列（最小堆）加速最短路径查找，初始化距离字典和路径列表。

2. \*\*Dijkstra算法：\*\*

- 通过贪心选择节点，使用优先队列实现，邻居松弛操作，维护距离和路径，堆结构确保选择最小距离的节点。

3. \*\*路径获取：\*\*

- 递归方式获取路径，输出每节点到其他节点的最短路径和距离。

4. \*\*应用：\*\*

- 针对网络结构和指定节点，运行Dijkstra算法，输出指定节点到其他节点的距离列表和路径。

\*\*输出结果：\*\*

- 距离列表、路径，各节点到其他节点的具体距离，按节点分类展示最短路径和距离。

\*\*实验简述：\*\*

1. \*\*初始化：\*\*

- 初始化距离表和下一跳表，利用网络拓扑结构初始化距离和下一跳信息。

2. \*\*松弛操作：\*\*

- 迭代计算和更新距离表和下一跳表，检查是否存在更短路径并更新信息。

3. \*\*距离矢量算法：\*\*

- 迭代次数为节点数目，每次迭代对所有节点进行松弛操作。

4. \*\*路径获取：\*\*

- 递归方式获取路径，输出每节点到其他节点的最短路径和距离。

5. \*\*应用：\*\*

- 针对网络拓扑结构和指定节点，运行距离矢量算法，输出指定节点到其他节点的距离列表和路径。

\*\*输出结果：\*\*

- 距离列表和路径，各节点到其他节点的具体距离，按节点分类展示最短路径和距离。