# 最短路径算法

## Dijkstra —— 贪心算法

从一个顶点到其余顶点的最短路径

设G=(V,E)是一个带权有向图，把图中顶点集合V分成两组，第1组为已求出最短路径的顶点（用S表示，初始时S只有一个源点，以后每求得一条最短路径v,...k，就将k加到集合S中，直到全部顶点都加入S）。第2组为其余未确定最短路径的顶点集合（用U表示），按最短路径长度的递增次序把第2组的顶点加入S中。

步骤：  
  
1. 初始时，S只包含源点，即`S={v}`，顶点v到自己的距离为0。U包含除v外的其他顶点，v到U中顶点i的距离为边上的权。  
2. 从U中选取一个顶点u，顶点v到u的距离最小，然后把顶点u加入S中。  
3. 以顶点u为新考虑的中间点，修改v到U中各个点的距离。  
4. 重复以上步骤知道S包含所有顶点。

## Floyd —— 动态规划

Floyd 算法是解决任意两点间的最短路径的一种算法，可以正确处理有向图或负权（但不可存在负权回路）的最短路径问题。该算法的时间复杂度为

，空间复杂度为

设

为从

到

的只以

集合中的节点为中间节点的最短路径的长度。

因此，

。伪代码描述如下：

// let dist be a |V| × |V| array of minimum distances initialized to ∞ (infinity)  
 for each vertex v  
 dist[v][v] ← 0  
 for each edge (u,v)  
 dist[u][v] ← w(u,v) // the weight of the edge (u,v)  
 for k from 1 to |V|  
 for i from 1 to |V|  
 for j from 1 to |V|  
 if dist[i][j] > dist[i][k] + dist[k][j]  
 dist[i][j] ← dist[i][k] + dist[k][j]  
 end if