数据结构

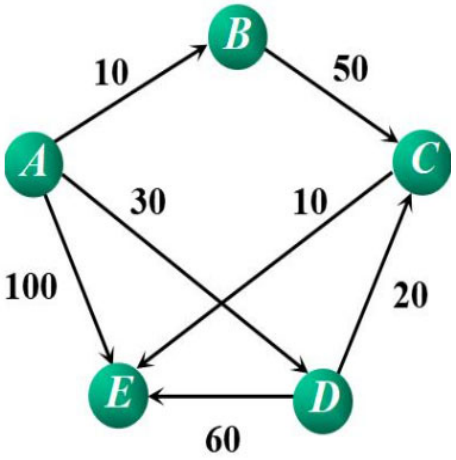
图的单源最短路径算法

Outline

- 图的最短路径问题
- Dijkstra算法的基本原则
- Dijkstra算法的贪心求解过程
- Dijkstra算法与Prim算法的比较

图的最短路径问题

- **问题的提出**
  - **假如要在计算机上建立一个交通咨询系统，可用图或网的结构来表示实际的交通网络，用顶点表示城市，边表示城市之间的交通联系。这个交通咨询系统可以回答旅客提出的各种问题。**如：
    —位旅客想从A城到B城，希望选择中转次数最少的路线。
    —位司机想选择一条从A城到B城所需距离最短的路径?



- **最短路径问题?**
  - ✓ **如果从图中某一顶点(称为源点)到达另一顶点(称为终点)的路径可能不止一条，如何找到一条路径使得沿此路径上各边上的权值总和达到最小。**

图的最短路径问题

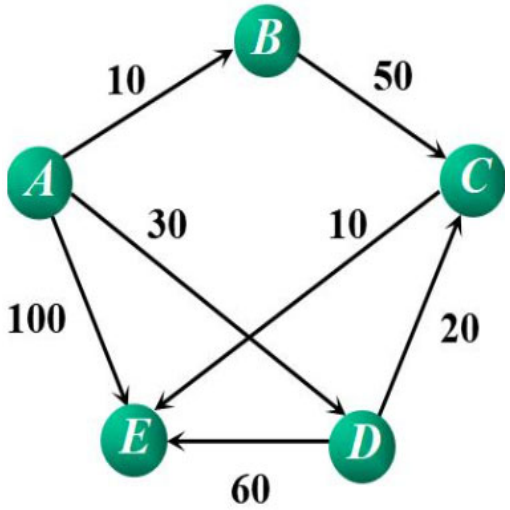
问题解法：

- **边上权值非负情形的单源最短路径问题**

  — **Dijkstra算法**

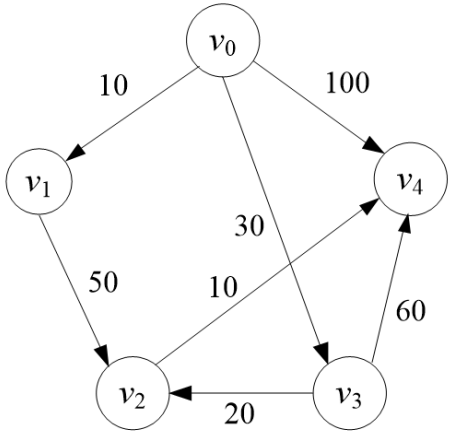
- **所有顶点之间的最短路径**

  — **Floyd算法**



Dijkstra算法的单源最短路径求解策略

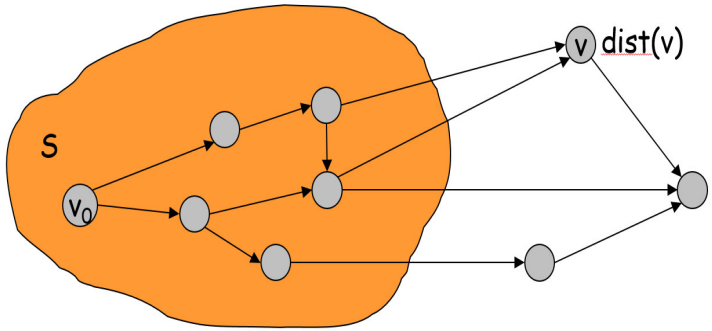
- **Dijkstra提出一种按路径长度递增次序，逐步产生从源点到其它各顶点的单源最短路径。**
- **为什么按照这个次序求解？**



源点	中间顶点	终点	最短路径长度
v_0		v_1	10
v_0		v_3	30
v_0	v_3	v_2	50
v_0	v_3, v_2	v_4	60

Dijkstra算法的基本原则

从源点v_0到其它任何顶点v的最短路径仅经过比v距离更近的顶点作为中间顶点。



Dijkstra算法的基本原则

从源点v_0到其它任何顶点v的最短路径仅经过比v距离更近的顶点作为中间顶点。

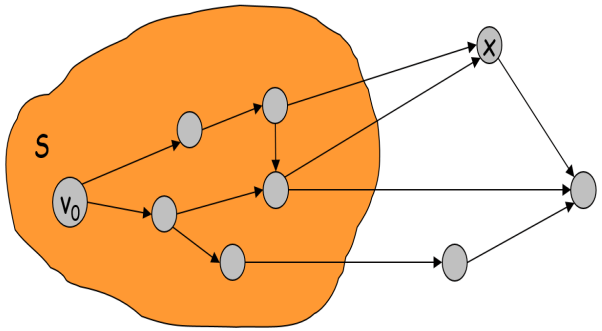
假设已求得最短路径的顶点集合$S=\{v_{p1}, v_{p2}, \cdots, v_{pk}\}$，则下一条最短路径（设终点为$x$）：

- ✓或者为弧〈v_0, x〉；
- ✓或者为一条只经过S中的某些顶点而最后到达终点x的路径。

Dijkstra算法的基本原则

假设已求得最短路径的顶点集合$S=\{v_{p1},v_{p2},\cdots,v_{pk}\}$，则下一条最短路径（设终点为$x$）：

- ✓ 或者为弧〈v_0, x〉；
- ✓ 或者为一条只经过S中的某些顶点而最后到达终点x的路径。

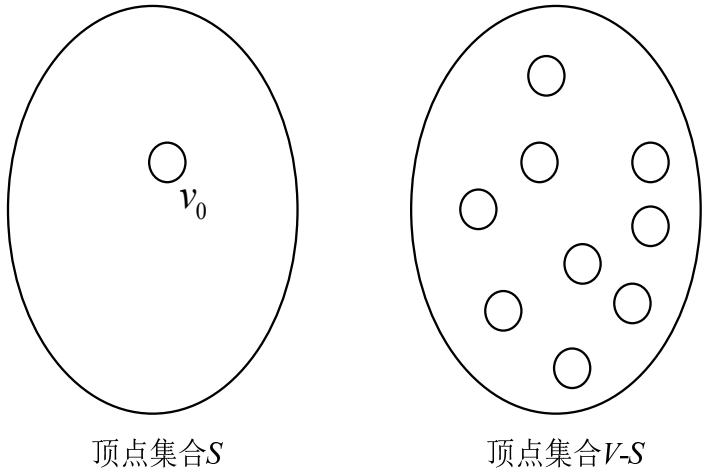


证明：假设此路径上存在一个不在集合S中出现的顶点，则说明存在终点不在S而路径长度比此更短的路径，这是不可能的！

因为我们是按长度递增的次序来产生各条最短路径的，而且各边的权值非负，故长度比此路径短的所有路径均已产生，它们的终点必在集合S中。

Dijkstra算法的贪心求解步骤

- 设置一个集合*S*，用于存放已求出的最短路径的顶点，则尚未确定最短路径的顶点属于集合*V–S*。
- 初始状态时，集合*S*中只有源点；
- 循环执行贪心选择逐步扩充集合*S*，直到集合*V–S*中的顶点全部加入到集合*S*中。
  - 每一步在*V–S*中选择当前离源点距离最短的顶点插入到集合*S*中



实现的关键问题：每一步做贪心选择时，如何有效地计算集合*V–S*中的各顶点对应的最短距离值？

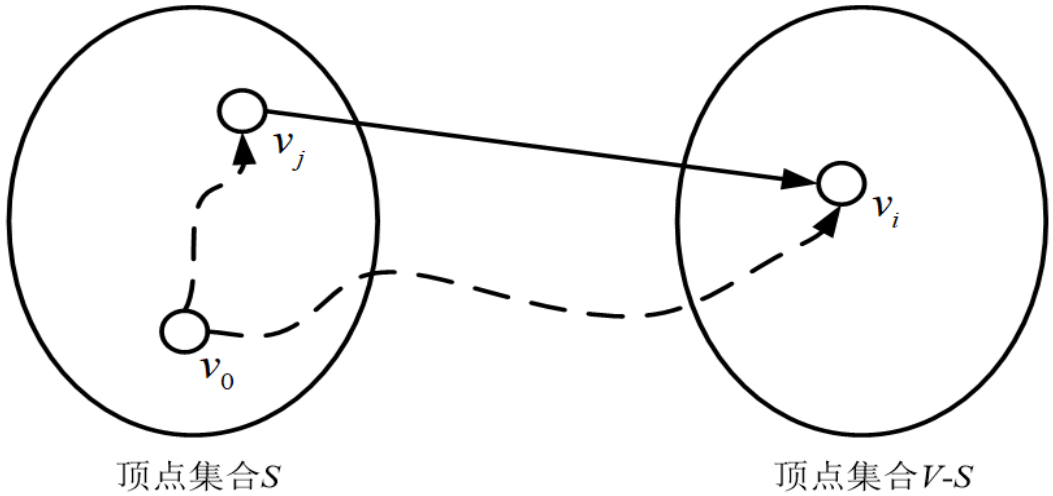
集合V–S中顶点最短距离值的计算

- 引入一个辅助数组dist[]。它的每一个分量dist[i]表示当前确定的从源点v_0到V-S中顶点v_i的最短路径的长度（当前最短距离）。
- 注意：算法执行过程中数组dist的元素值是一个增量的变化过程。
- 初始状态：
  - 若从源点v_0到顶点v_i有边，则dist[i]为该边上的权值；否则为+∞。
- 当从集合V-S中选择一个当前距离最小的顶点v_j加入集合S后，需要及时调整V-S中剩余的各个顶点的当前最短距离值。

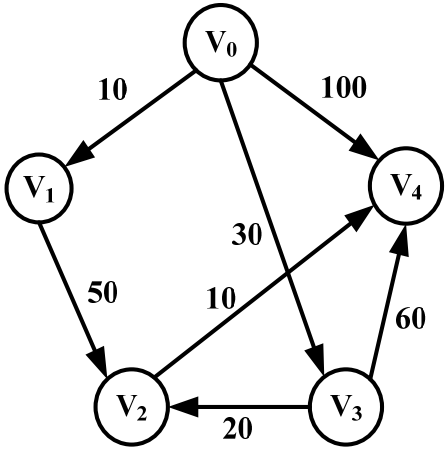
顶点最短距离dist值的调整方法

- **当从集合V-S中选择一个当前距离最小的顶点v_j加入集合S后，需要调整V-S中剩余顶点的当前最短距离值。**
- **对于V-S中的顶点v_i，其最短距离值的调整方法是：**

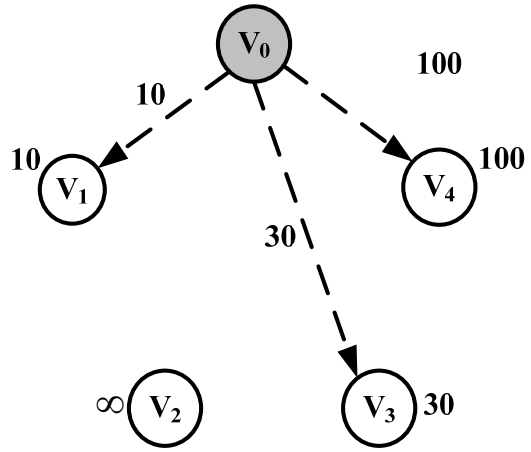
$$dist[i]=Min\{dist[i],\ dist[j]+cost(j,\ i)\ \}$$



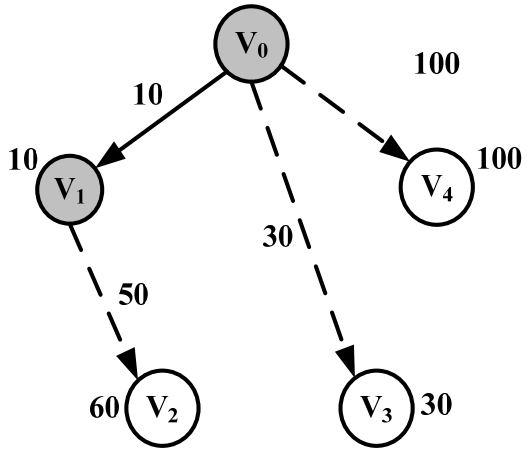
Dijkstra算法求单源最短路径示例



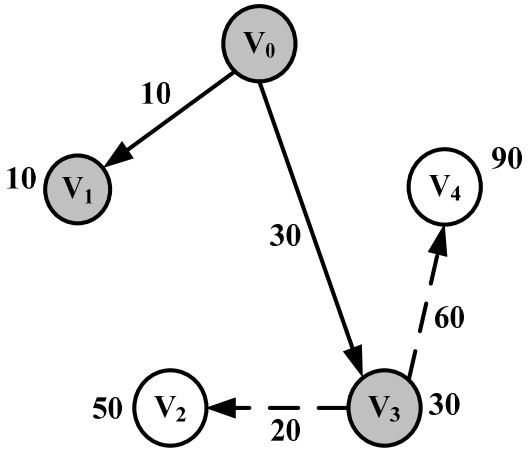
(a) 带权有向图G_9



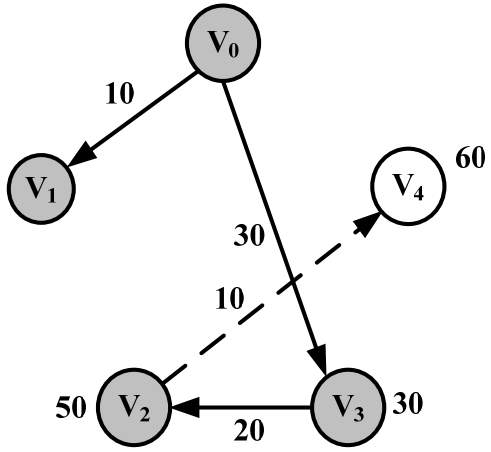
(b)



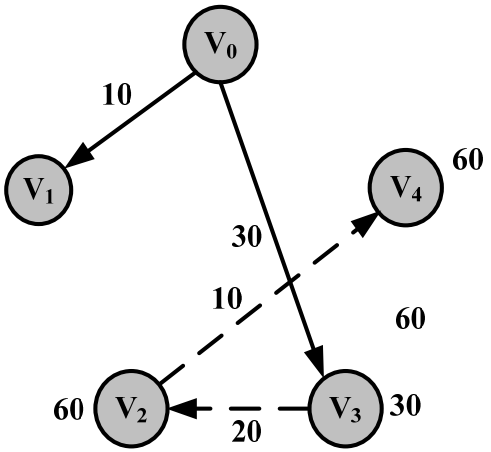
(c)



(d)



(e)



(f)

Dijkstra算法的具体实现

- 图的存储结构？
  - **邻接矩阵方式**

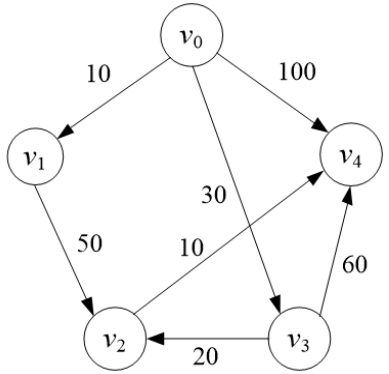
- 如何标识图中各顶点在算法执行过程中是否已被加入到集合S中？
  - 设置一个一维数组S[]，并规定：
    - S[i]=0　顶点v_i未加入集合S
    - S[i]=1　顶点v_i已加入集合S
- 如何记录Dijkstra算法所求出的从源点到各顶点的最短路径？
  - 引入一个数组path[]，其中，path[i]中保存了从源点到终点v_i的最短路径上该顶点的前驱顶点的序号。

```
//求出从编号v的顶点到其余各点的最短路径，path[]中存放路径，dist[]中存放路径长度
template<class T>
void MGraph<T>::Dijkstra(int v, int path[], int dist[]) {
    bool *s=new bool[vexnum];
    for(i=0;i<vexnum;i++){
        s[i]=false;  dist[i]=GetEdgeValue(v,i);          //距离初始化
        if(dist[i]<INFINITY || i==v)   path[i]=v;
        else path[i]= -1;          //表示顶点i前驱顶点不是v
    }
   dist[v]=0; s[v]=true;
   for(i=1; i<vexnum; i++){
       min=INFINITY;          //设置最短路径初值为足够大的数
       for( j=0;j<vexnum;j++)
           if(!s[j] && dist[j]<min){
                k=j; min=dist[j];
            }
       s[k]=true;    //将离v最近的顶点加入s集
       for(int w=0; w<vexnum; w++)
           if(!s[w] && dist[w]>dist[k]+GetEdgeValue(k,w))
                dist[w]=dist[k]+GetEdgeValue(k,w);  path[w]=k;
            }
    }
    delete[]s;
}
```

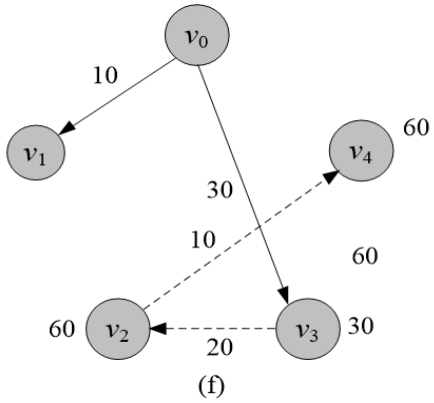
算法的时间复杂度为 $O(n^2)$

Dijkstra算法与Prim算法的比较

- **Dijkstra算法与求最小生成树的Prim算法思想非常相似?**
  - ✓ **在图(f)中从树根（源点）到其余每个顶点均存在一条有向路径，且无回路存在。故称这棵有向生成树为G_9的最短路径树，简称SPT树。**
  - ✓ **这样，对一个带权有向图求单源最短路径问题实际上可看成是对一个带权有向图构造SPT树的问题。**



(a) 带权有向图G_9



(f)

思考题 

1. Dijkstra算法只能适用于图中仅包含非负权值边的情形?

   为什么?

2. Dijkstra算法的贪心选择准则是什么？你如何理解?

Thank you for your attention!