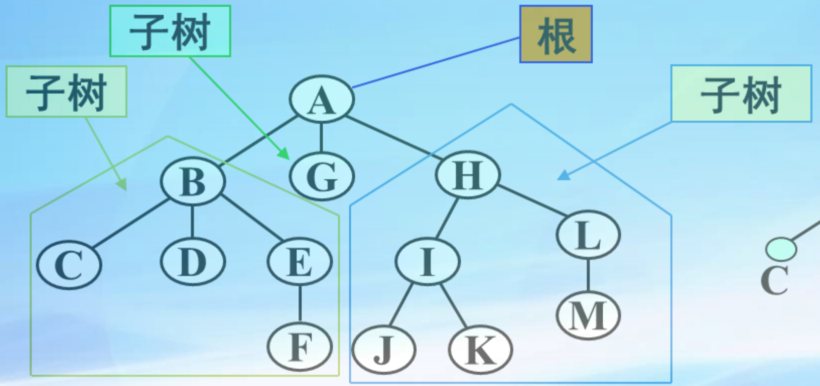
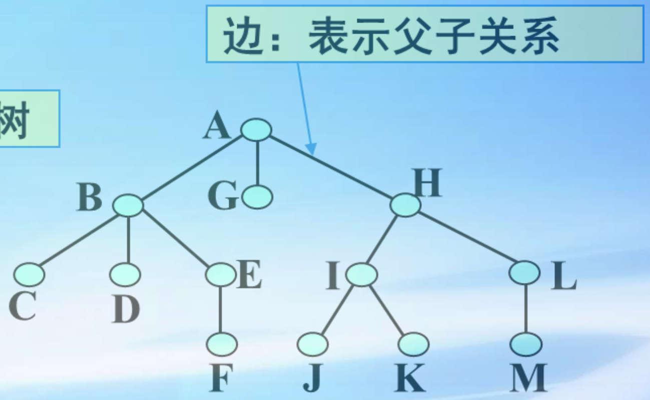
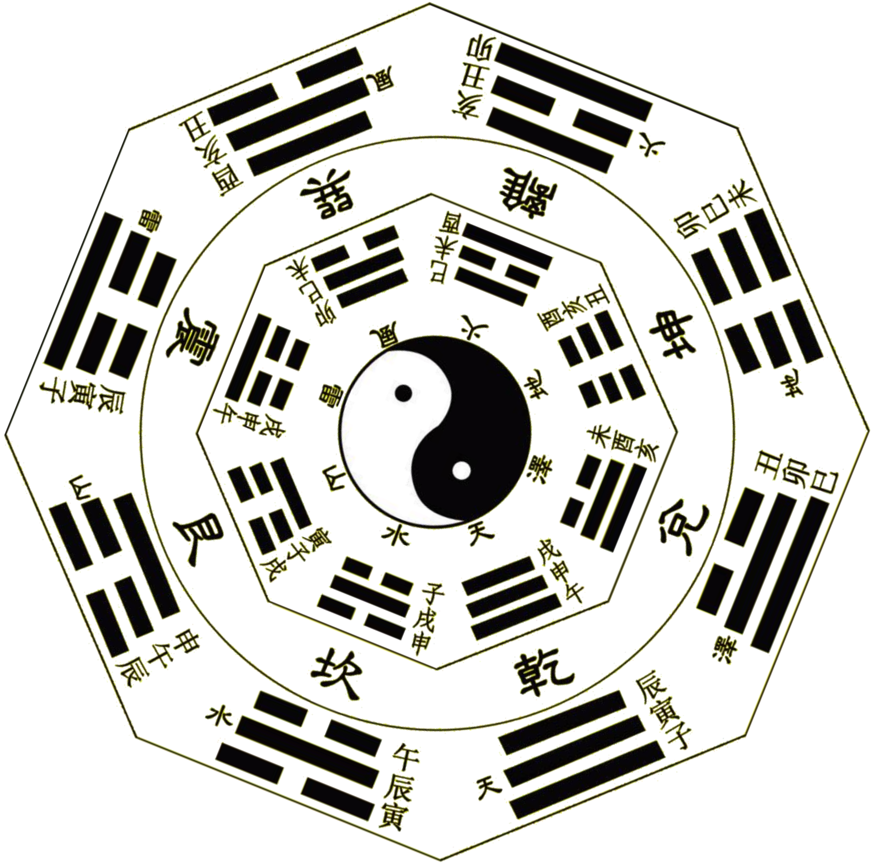
《数据结构》

最短路径





旅行家的困扰



旅行家的困扰

新疆特克斯县“八卦城”

旅行家的困扰

特克斯县

怎么样帮助困扰的旅行家找到去各个地点的最短路线呢？

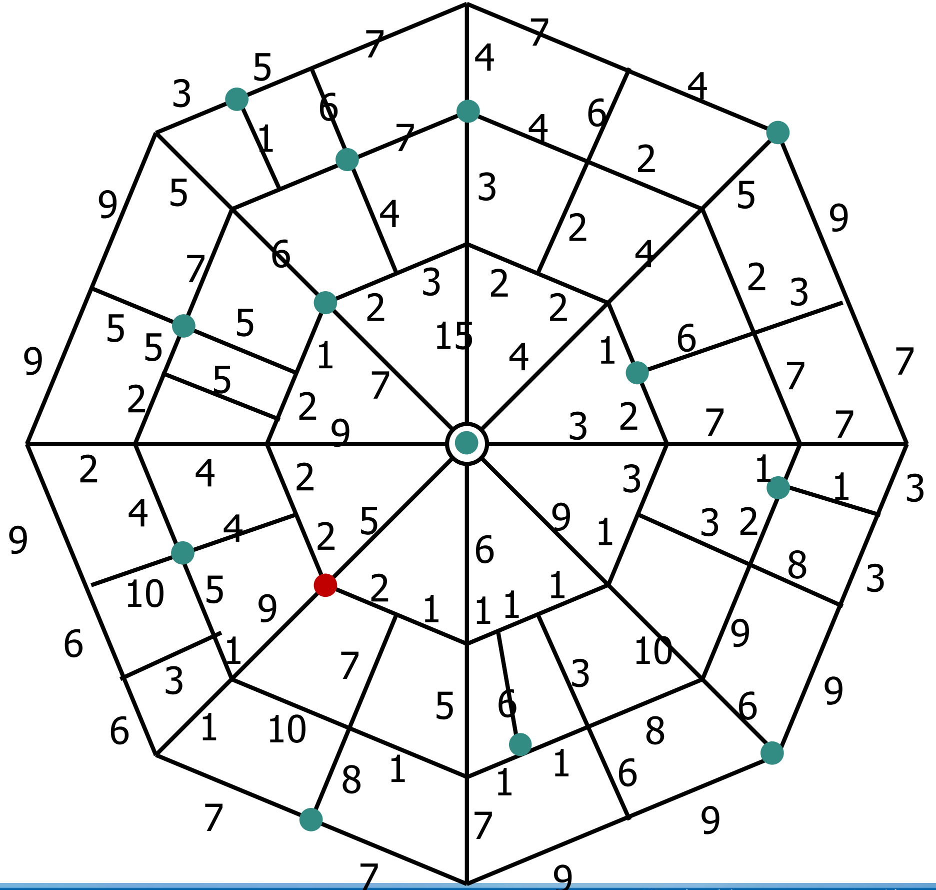
- 旅行家居住的旅馆
- 旅行家想去的地点

使用加权有向图进行建模：

1、将特克斯县中的道路交点看作图的顶点

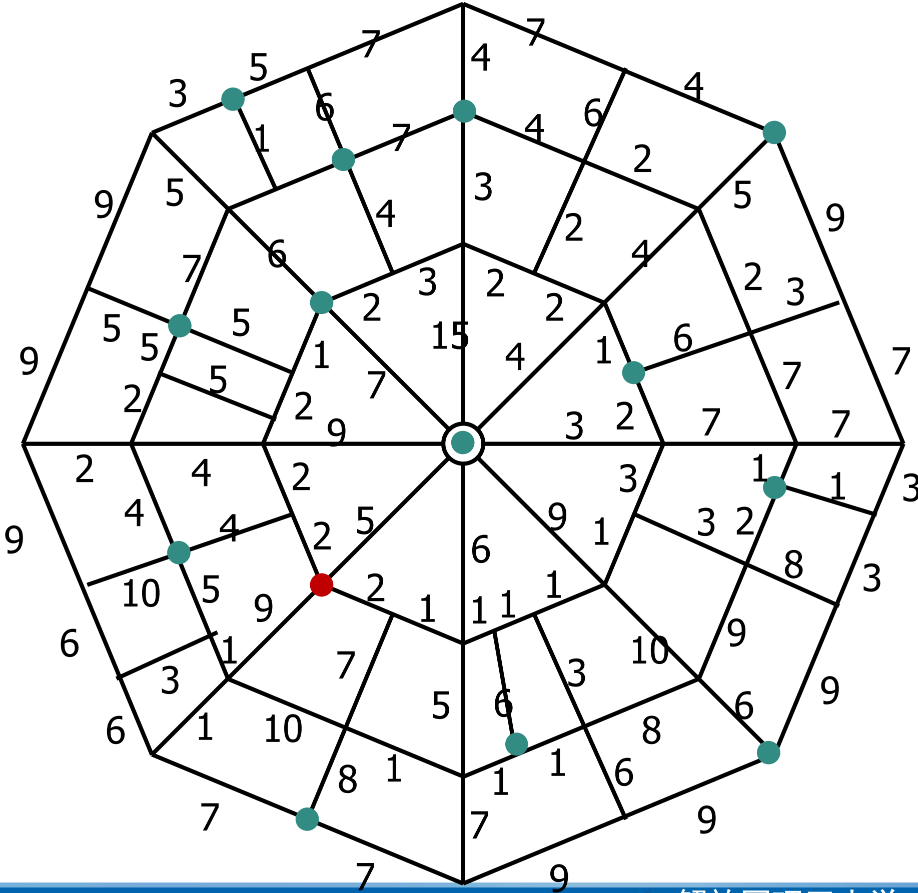
2、将城市中的道路看作图的边（假设均为双行道）

3、将道路的通行时间作为边的耗费



问题求解

以红色顶点为起点，求解其到其他任意顶点的最短路径——求单源最短路径问题



教学目标和要求

1.理解最短路径的基本概念和实际意义

2.使用Dijkstra算法原理，求解单源最短路径序列

3.编程实现Dijkstra算法，并解决简单的应用问题

最短路径

单源最短路径：某点（称为源点）到其余各点的最短路径

每对顶点之间的最短路径

Dijkstra算法

适用于：有向加权图，无向加权图，0/1图

算法基本思想

与Prim算法类似，采用路径延伸法

求解过程中，将顶点分成生长点和非生长点，每次选最短的路径，修改最短路径

生长点：源点s和已确定最短路径的顶点

非生长点：未确定最短路径的顶点

使用待定路径表暂存源点s到每个非生长点的“当前最短路径”

Dijkstra算法描述

算法描述

引入两个变量P(v)和D(v)，P(v)代表从源点s到非生长点v的当前待定路径，用顶点序列表示，D(v)代表当前P(v)的长度

步骤1）构造初始待定路径表

对每个非生长点v（共n-1个），定初值：

P(v)=sv，D(v)=边<s，v>的长度C<s，v>，如果边<s，v>不存在，则认为其长度为∞

算法描述

步骤1）构造初始待定路径表

步骤2）循环n-2遍

①选择最短的待定路径

从待定路径表中选出一条最短的，设其为s到v的路径，该路径便是s到v的最短路径，v变成为新的生长点

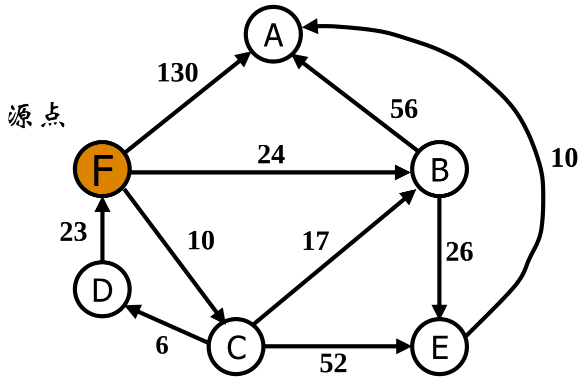
②修改待定路径表

对剩下的每个非生长点w，设其待定路径为P(w)，长度为D(w)，比较D(w)与D(v)+C<v，w>的大小，这里的v是新生长点：

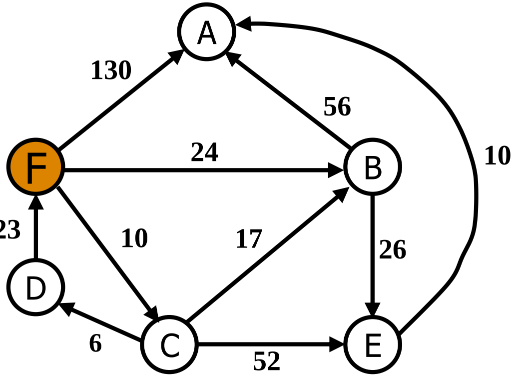
如果D(w)≤D(v)+C<v，w>什么也不做

否则，将D(w)改为D(v)+C<v，w>，同时将P(w)改为P(v)接w，表示从s到v后再到w，比原来从s到w（不过v）更短

Example



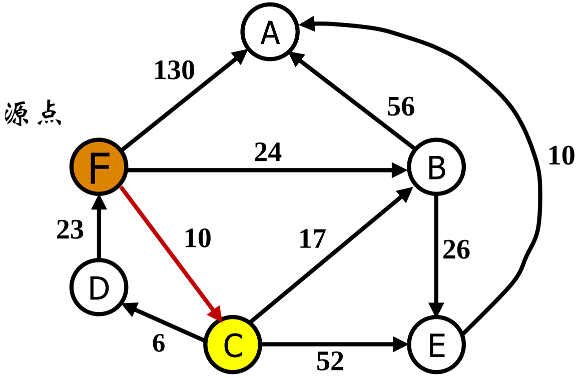
Example



步骤1

生长点	A	B	C	D	E
F					

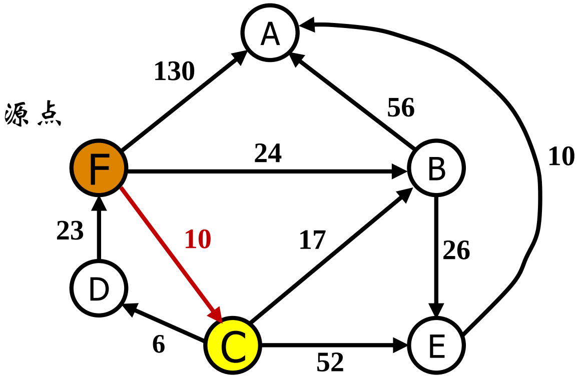
Example



生长点	A	B	C	D	E
F	P(A)=FA D(A)=130	P(B)=FB D(B)=24	P(C)=FC D(C)=10	P(D)=空 D(D)=∞	P(E)=空 D(E)=∞

Dijkstra算法示例

Example

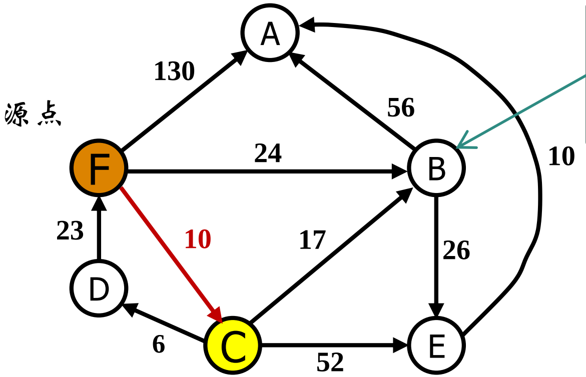


选定C

生长点	A	B	C	D	E
F	P(A)=FA D(A)=130	P(B)=FB D(B)=24	**P(C)=FC D(C)=10**	P(D)=空 D(D)=∞	P(E)=空 D(E)=∞
C	P(A)=FA D(A)=130				

Dijkstra算法示例

Example



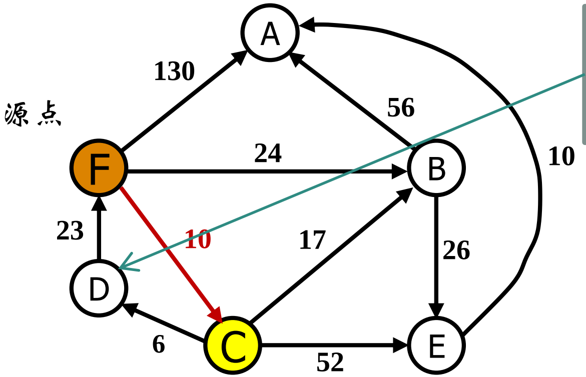
D(B)<(D(C)+C<C,B>)
即：24<(10+17)=27

选定C

生长点	A	B	C	D	E
F	P(A)=FA D(A)=130	P(B)=FB D(B)=24	**P(C)=FC** **D(C)=10**	P(D)=空 D(D)=∞	P(E)=空 D(E)=∞
C	P(A)=FA D(A)=130	P(B)=FB D(B)=24			

Dijkstra算法示例

Example



$D(D)>(D(C)+C<C,D>)$

即：$\infty>(10+6)=16$

$P(D)=P(C)\cup\{D\}$

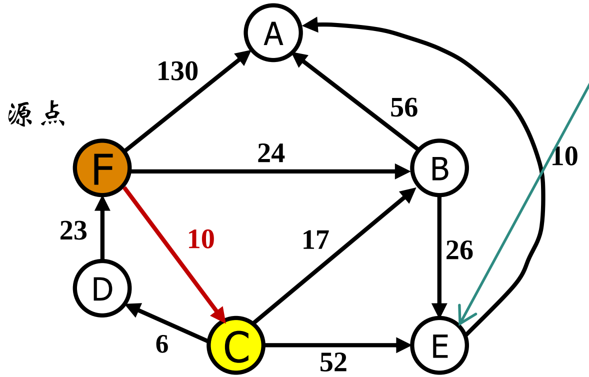
$=\{FCD\}$

选定C

生长点	A	B	C	D	E
F	P(A)=FA D(A)=130	P(B)=FB D(B)=24	**P(C)=FC D(C)=10**	P(D)=空 D(D)=∞	P(E)=空 D(E)=∞
C	P(A)=FA D(A)=130	P(B)=FB D(B)=24		P(D)=FCD D(D)=16	

Dijkstra算法示例

Example



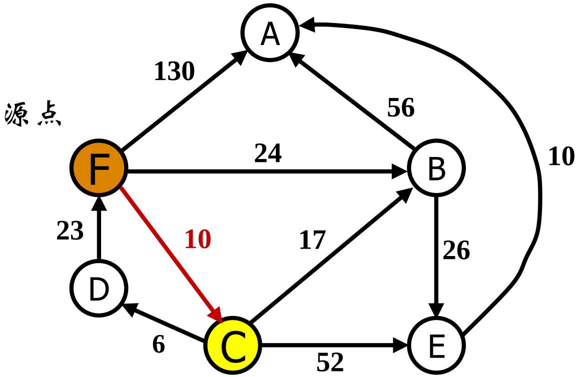
$D(E)>(D(C)+C<C,E>)$
即：$\infty>(10+52)=62$

P(E)=P(C)U{E}
={FCE}

选定C

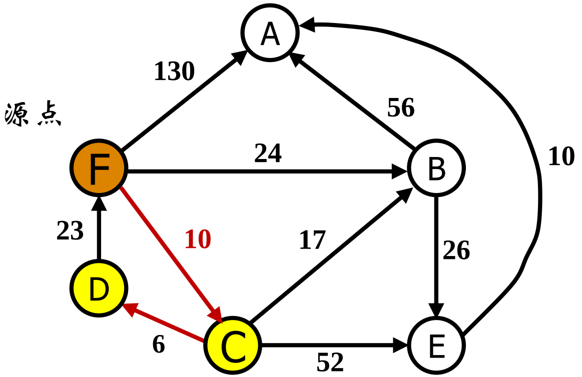
生长点	A	B	C	D	E
F	P(A)=FA D(A)=130	P(B)=FB D(B)=24	**P(C)=FC D(C)=10**	P(D)=空 D(D)=∞	P(E)=空 D(E)=∞
C	P(A)=FA D(A)=130	P(B)=FB D(B)=24		P(D)=FCD D(D)=16	P(E)=FCE D(E)=62

Example



生长点	A	B	C	D	E
F	P(A)=FA D(A)=130	P(B)=FB D(B)=24	**P(C)=FC D(C)=10**	P(D)=空 D(D)=∞	P(E)=空 D(E)=∞
C	P(A)=FA D(A)=130	P(B)=FB D(B)=24		P(D)=FCD D(D)=16	P(E)=FCE D(E)=62

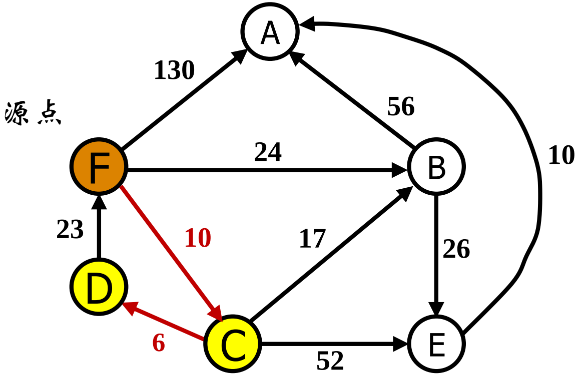
Example



生长点	A	B	C	D	E
F	P(A)=FA D(A)=130	P(B)=FB D(B)=24	**P(C)=FC D(C)=10**	P(D)=空 D(D)=∞	P(E)=空 D(E)=∞
C	P(A)=FA D(A)=130	P(B)=FB D(B)=24		P(D)=FCD D(D)=16	P(E)=FCE D(E)=62

Dijkstra算法示例

Example

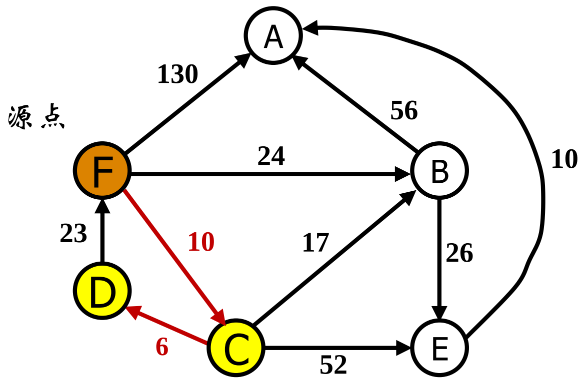


选定D

生长点	A	B	C	D	E
F	P(A)=FA D(A)=130	P(B)=FB D(B)=24	**P(C)=FC D(C)=10**	P(D)=空 D(D)=∞	P(E)=空 D(E)=∞
C	P(A)=FA D(A)=130	P(B)=FB D(B)=24		**P(D)=FCD D(D)=16**	P(E)=FCE D(E)=62
D					

Dijkstra算法示例

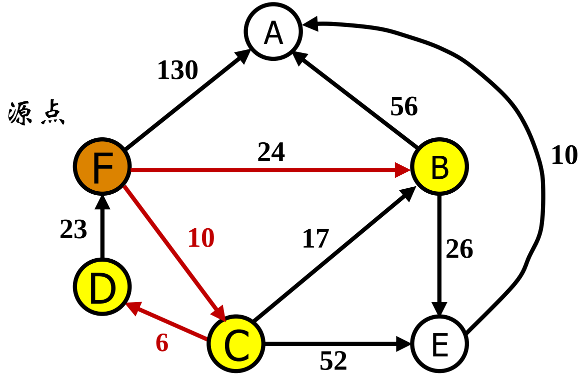
Example



生长点	A	B	C	D	E
F	P(A)=FA D(A)=130	P(B)=FB D(B)=24	**P(C)=FC** **D(C)=10**	P(D)=空 D(D)=∞	P(E)=空 D(E)=∞
C	P(A)=FA D(A)=130	P(B)=FB D(B)=24		**P(D)=FCD** **D(D)=16**	P(E)=FCE D(E)=62
D	P(A)=FA D(A)=130	P(B)=FB D(B)=24			P(E)=FCE D(E)=62

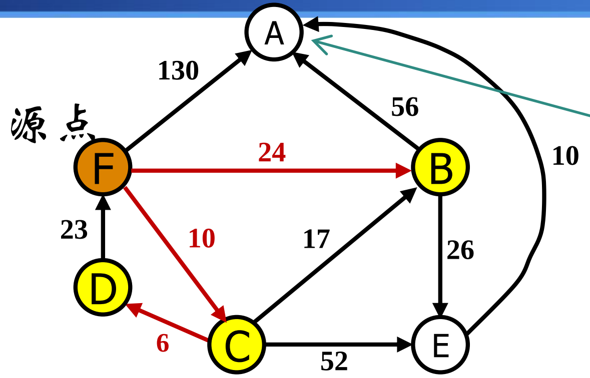
选定D

Example



生长点	A	B	C	D	E
F	P(A)=FA D(A)=130	P(B)=FB D(B)=24	**P(C)=FC** **D(C)=10**	P(D)=空 D(D)=∞	P(E)=空 D(E)=∞
C	P(A)=FA D(A)=130	P(B)=FB D(B)=24		**P(D)=FCD** **D(D)=16**	P(E)=FCE D(E)=62
D	P(A)=FA D(A)=130	P(B)=FB D(B)=24			P(E)=FCE D(E)=62

Example



D(A)>(D(B)+C<B,A>)
即：130>(24+56)=80

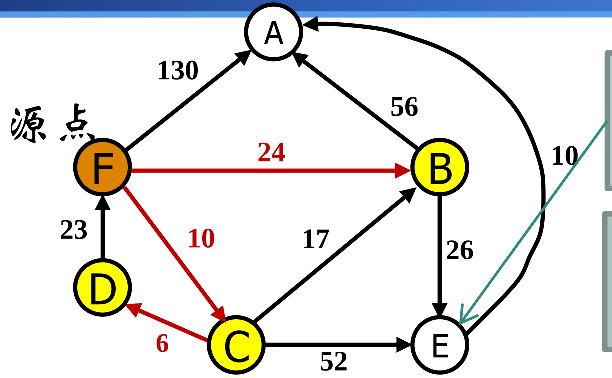
P(A)=P(B)U{A}
={FBA}

选定B

生长点	A	B	C	D	E
F	P(A)=FA D(A)=130	P(B)=FB D(B)=24	**P(C)=FC** **D(C)=10**	P(D)=空 D(D)=∞	P(E)=空 D(E)=∞
C	P(A)=FA D(A)=130	P(B)=FB D(B)=24		**P(D)=FCD** **D(D)=16**	P(E)=FCE D(E)=62
D	P(A)=FA D(A)=130	**P(B)=FB** **D(B)=24**			P(E)=FCE D(E)=62
B	P(A)=FBA D(A)=80				

Dijkstra算法示例

Example



D(E)>(D(B)+C<B,E>)

即：62>(24+26)=50

P(E)=P(B)U{E}

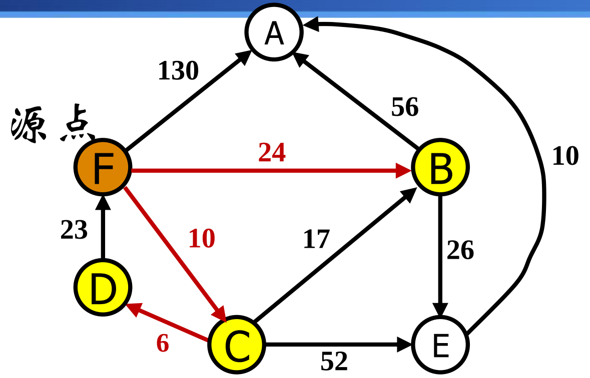
={FBE}

选定B

生长点	A	B	C	D	E
F	P(A)=FA D(A)=130	P(B)=FB D(B)=24	**P(C)=FC D(C)=10**	P(D)=空 D(D)=∞	P(E)=空 D(E)=∞
C	P(A)=FA D(A)=130	P(B)=FB D(B)=24		**P(D)=FCD D(D)=16**	P(E)=FCE D(E)=62
D	P(A)=FA D(A)=130	**P(B)=FB D(B)=24**			P(E)=FCE D(E)=62
B	P(A)=FBA D(A)=80				P(E)=FBE D(E)=50

Dijkstra算法示例

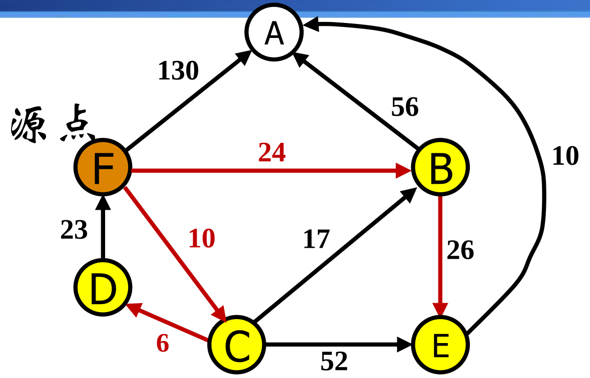
Example



生长点	A	B	C	D	E
F	P(A)=FA D(A)=130	P(B)=FB D(B)=24	**P(C)=FC** **D(C)=10**	P(D)=空 D(D)=∞	P(E)=空 D(E)=∞
C	P(A)=FA D(A)=130	P(B)=FB D(B)=24		**P(D)=FCD** **D(D)=16**	P(E)=FCE D(E)=62
D	P(A)=FA D(A)=130	**P(B)=FB** **D(B)=24**			P(E)=FCE D(E)=62
B	P(A)=FBA D(A)=80				P(E)=FBE D(E)=50

Dijkstra算法示例

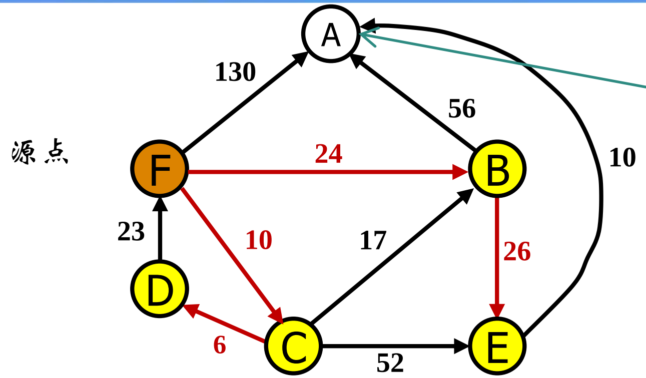
Example



生长点	A	B	C	D	E
F	P(A)=FA D(A)=130	P(B)=FB D(B)=24	**P(C)=FC D(C)=10**	P(D)=空 D(D)=∞	P(E)=空 D(E)=∞
C	P(A)=FA D(A)=130	P(B)=FB D(B)=24		**P(D)=FCD D(D)=16**	P(E)=FCE D(E)=62
D	P(A)=FA D(A)=130	**P(B)=FB D(B)=24**			P(E)=FCE D(E)=62
B	P(A)=FBA D(A)=80				P(E)=FBE D(E)=50

Dijkstra算法示例

Example



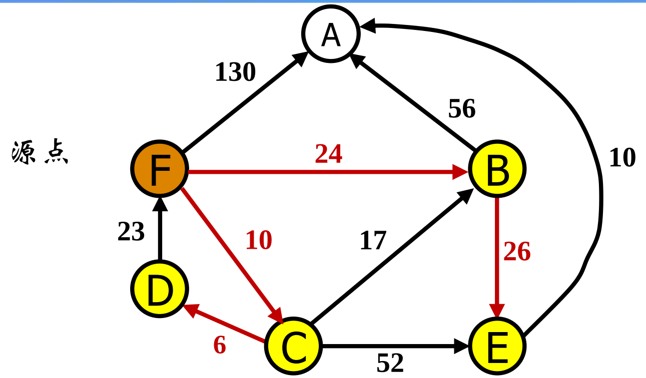
D(A)>(D(E)+C<E,A>)
即：80>(50+10)=60

P(A)=P(E)U{A}
={FBEA}

生长点	A	B	C	D	E
F	P(A)=FA D(A)=130	P(B)=FB D(B)=24	**P(C)=FC** **D(C)=10**	P(D)=空 D(D)=∞	P(E)=空 D(E)=∞
C	P(A)=FA D(A)=130	P(B)=FB D(B)=24		**P(D)=FCD** **D(D)=16**	P(E)=FCE D(E)=62
D	P(A)=FA D(A)=130	**P(B)=FB** **D(B)=24**			P(E)=FCE D(E)=62
B	P(A)=FBA D(A)=80				**P(E)=FBE** **D(E)=50**
选定E → E	P(A)=FBEA D(A)=60				

Dijkstra算法示例

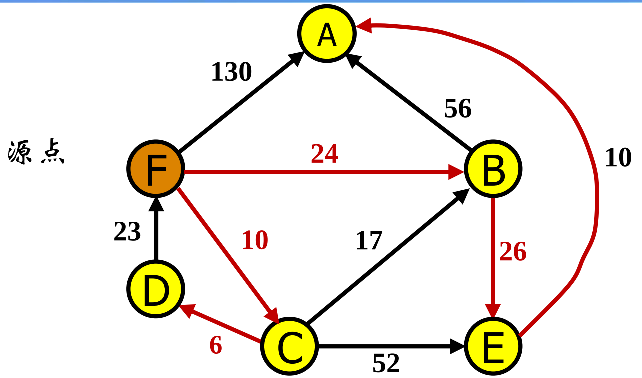
Example



生长点	A	B	C	D	E
F	P(A)=FA D(A)=130	P(B)=FB D(B)=24	**P(C)=FC D(C)=10**	P(D)=空 D(D)=∞	P(E)=空 D(E)=∞
C	P(A)=FA D(A)=130	P(B)=FB D(B)=24		**P(D)=FCD D(D)=16**	P(E)=FCE D(E)=62
D	P(A)=FA D(A)=130	**P(B)=FB D(B)=24**			P(E)=FCE D(E)=62
B	P(A)=FBA D(A)=80				**P(E)=FBE D(E)=50**
E	P(A)=FBEA D(A)=60				

Dijkstra算法示例

Example

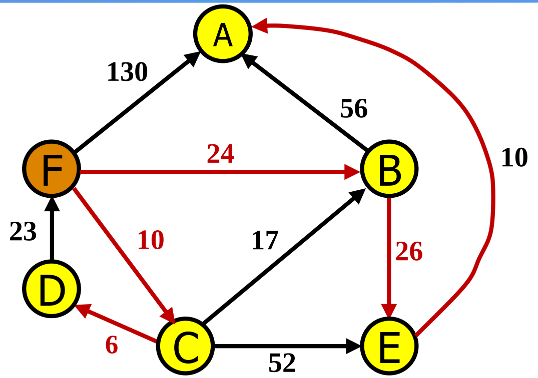


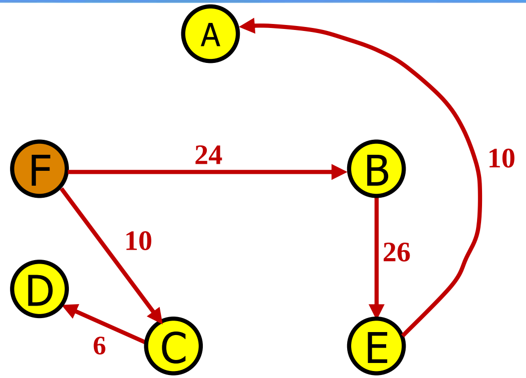
生长点	A	B	C	D	E
F	P(A)=FA D(A)=130	P(B)=FB D(B)=24	**P(C)=FC D(C)=10**	P(D)=空 D(D)=∞	P(E)=空 D(E)=∞
C	P(A)=FA D(A)=130	P(B)=FB D(B)=24		**P(D)=FCD D(D)=16**	P(E)=FCE D(E)=62
D	P(A)=FA D(A)=130	**P(B)=FB D(B)=24**			P(E)=FCE D(E)=62
B	P(A)=FBA D(A)=80				**P(E)=FBE D(E)=50**
E	$\boxed{\text{P(A)=FBEA D(A)=60}}$				

Dijkstra算法示例

Example

源点





生长点	A	B	C	D	E
F	P(A)=FA D(A)=130	P(B)=FB D(B)=24	**P(C)=FC D(C)=10**	P(D)=空 D(D)=∞	P(E)=空 D(E)=∞
C	P(A)=FA D(A)=130	P(B)=FB D(B)=24		**P(D)=FCD D(D)=16**	P(E)=FCE D(E)=62
D	P(A)=FA D(A)=130	**P(B)=FB D(B)=24**			P(E)=FCE D(E)=62
B	P(A)=FBA D(A)=80				**P(E)=FBE D(E)=50**
E	P(A)=FBEA D(A)=60				

Dijkstra算法的实现

采用邻接表存储有向图

邻接点域　边长度域　指向下一个边结点的指针域

边结点结构

adjvex	cost	next

顶点结点结构

name	firstarc	path	dist	mark

path：路径序列，代表P(v)

dist：最短路径长度,代替路径长度D(v)

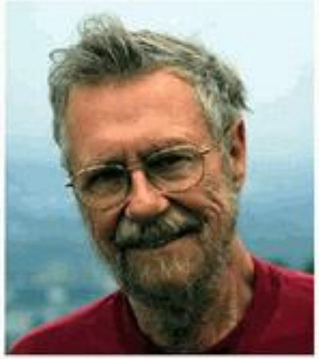
mark：0代表非生长点，1代表生长点

Dijkstra算法的实现

```
void shortpath_dijkstra(L[M], int n, int s)     //n顶点数，s源点号
{
1.    for(i=0; i<n; i++) { L[i].path = L[s].name; L[i].dist = MAX; L[i].mark = 0; } //设置待定路径初值
2.    L[s].mark = 1; L[s].dist = 0;  //设置s为源点
3.    p = L[s].firstarc;    //因为存在边<s,w>，修改s的邻接点w对应的初始待选路径
4.    while  (p!=NULL)
        {    w = p->adjvex; L[w].dist = p->cost;  p = p->next;    }
5.     for(i=0; i<n-2; i++)  //选择n-2次
6.         {     k = -1; d = MAX;
7.               for(j=0; j<n; j++)   //选择待选路径中的最短路径
8.                     {   if(L[j].dist<d&&L[j].mark==0)  { k=j; d=L[j].dist; }  }
9.               L[k].mark = 1;    //k为新的生长点
10.               p = L[k].firstarc;   //顺着k的邻接表修改邻接点的待定路径
11.               while(p!=NULL)
12.                  {     w = p->adjvex;
13.                        if(L[w].mark==0 &&L[w].dist>L[k].dist+p->cost)   //修改非生长点w的待定路径
14.                         { L[w].path = L[k].path+L[k].name; L[w].dist = L[k].dist+p->cost; }
15.                        p = p->next;    }
         }
   }
```

算法的时间复杂性为$O(n^2)$

关于狄杰斯特拉



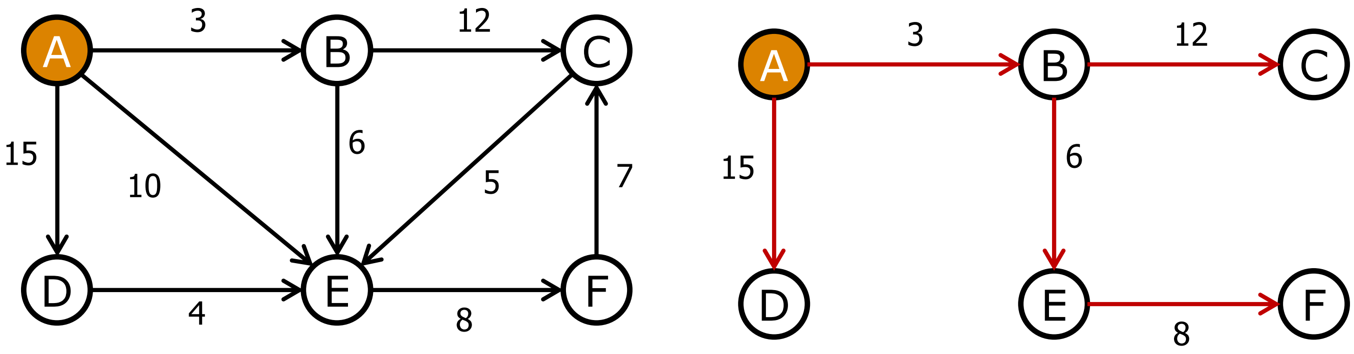
狄杰斯特拉

德加·狄杰斯特拉（Edsger Dijkstra，1930－2002）

- ◆荷兰人，Amsterdam 大学博士学位。
- ◆1972年图灵奖获得者
- ◆主要贡献：
  - ◆最早指出“goto有害”
  - ◆首创结构化程序设计
- ◆1983年，Dijkstra有两篇论文入选ACM的25篇里程碑意义的论文。
- ◆*1956年*，设计Dijkstra算法——运动路径规划问题。

Dijkstra算法本质

从源点逐步生长，类似于Prim算法。
贪心法原理。



Dijkstra算法的应用场景

- 计算机网络路由选择问题。**OSPF（open shortest path first，**开放最短路径优先）算法是**Dijkstra**算法在网络路由中的一个具体实现。
- 物流配送方案的优化设计问题。在电子商务中，运送商品，物流公司选择最短路径，降低配送服务价格。
- 求次短路径问题。例如，百度地图中的线路推荐方案。

智能推荐方案

最短路程方案

Dijkstra算法的局限性

- 图的权值只能是正实数。

改进Dijkstra算法的实现

```
void shortpath_dijkstra(L[M], int n, int s)    //n顶点数，s源点号
{
    for(i=0; i<n; i++) { L[i].path = L[s].name; L[i].dist = MAX; L[i].mark = 0; } //设置待定路径初值
    L[s].mark = 1; L[s].dist = 0;  //设置s为源点
    p = L[s].firstarc;   //因为存在边<s,w>，修改s的邻接点w对应的初始待选路径
    while  (p!=NULL)
     {   w = p->adjvex; L[w].dist = p->cost;  p = p->next;   }
     for(i=0; i<n-2; i++)  //选择n-2次
       {   k = -1; d = MAX;
           for(j=0; j<n; j++)   //选择待选路径中的最短路径
                 {  if(L[j].dist<d&&L[j].mark==0)  { k=j; d=L[j].dist; }  }
           L[k].mark = 1;    //k为新的生长点
            p = L[k].firstarc;   //顺着k的邻接表修改邻接点的待定路径
             while(p!=NULL)
               {    w = p->adjvex;
                    if(L[w].mark==0 &&L[w].dist>L[k].dist+p->cost)   //修改非生长点w的待定路径
                     { L[w].path = L[k].path+L[k].name; L[w].dist = L[k].dist+p->cost; }
                    p = p->next;    }
       }
  }
```

穷举法效率低

路径序列空间浪费大

思考

- 如何利用**Dijkstra**算法求次短路径？
- 查资料，了解可以求解含有负实数边的算法，以及使用该算法的前提条件是什么？

The End, Thank You!