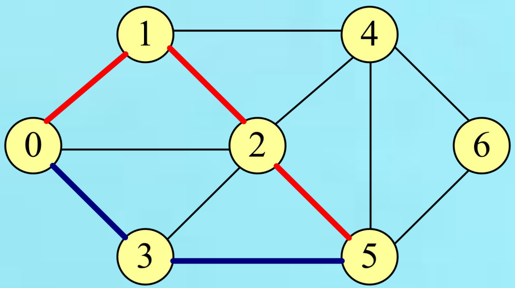
本节主题：

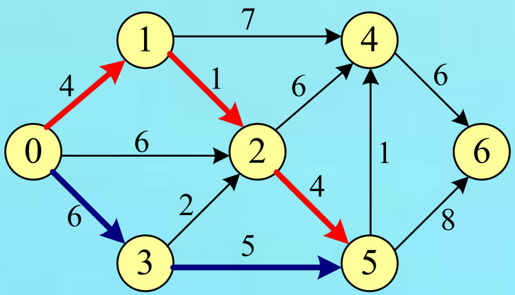
从一个顶点到其余各顶点的最短路径

最短路径

- 对无权图
  - 若从一顶点到另一顶点存在着一条路径，则称该**路径长度**为该路径上所经过的边的数目，它等于该路径上的顶点数减1。
  - 由于从一顶点到另一顶点可能存在着多条路径，每条路径上所经过的边数可能不同，即路径长度不同，我们把路径长度最短（即经过的边数最少）的那条路径叫做**最短路径**，其路径长度叫做**最短路径长度**或**最短距离**。

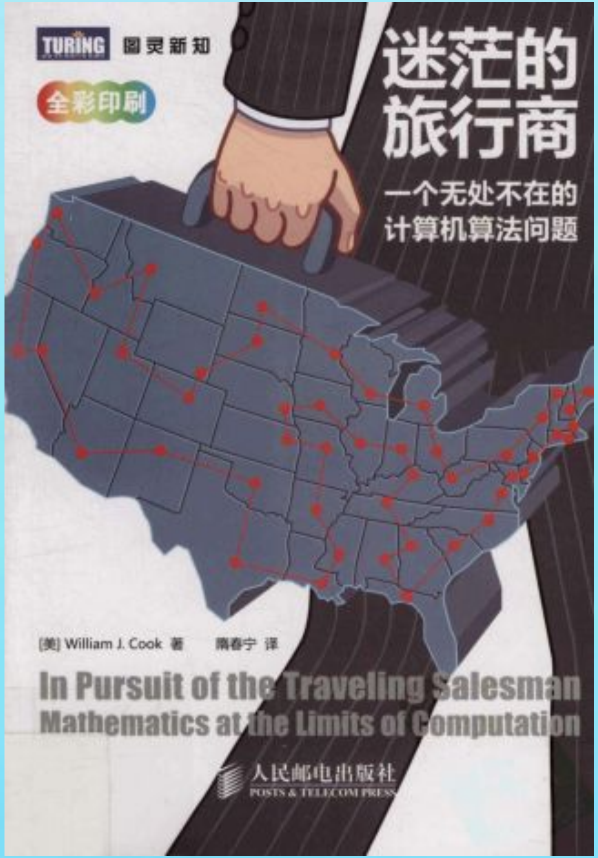


- 对带权图
  - 对于带权的图，考虑路径上各边上的权值，则通常把一条路径上所经边的权值之和定义为该路径的**路径长度**或称**带权路径长度**。
  - 从源点到终点可能不止一条路径，把带权路径长度最短的那条路径称为**最短路径**，其路径长度（权值之和）称为**最短路径长度**或者**最短距离**。



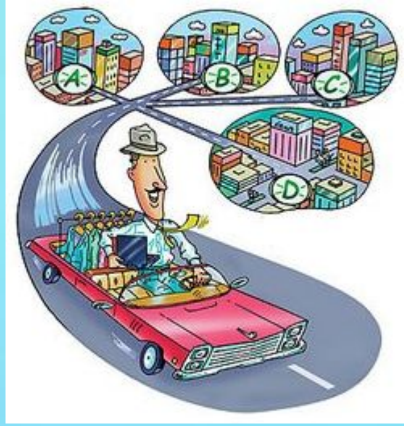
- 穷举法求解？

广泛应用

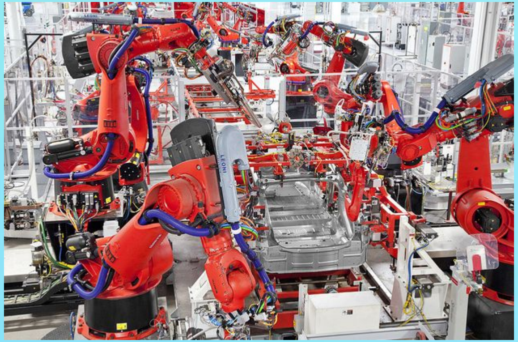


导航中的路径规划


货郎担问题

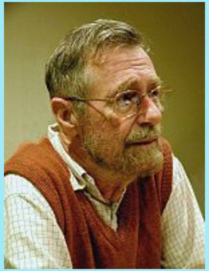


旅行商问题

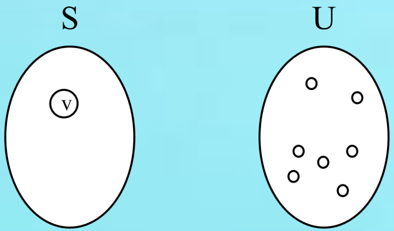

机械臂路径

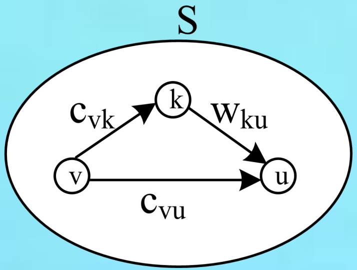
从一个顶点到其余各顶点的最短路径

- 问题
  - 给定一个带权有向图G与源点v，求从v到G中其他顶点的最短路径，并限定各边上的权值大于或等于0。
- 狄克斯特拉（Dijkstra）算法基本思想
  - 设G=(V，E)是一个带权有向图，把图中顶点集合V分成两组
    - 第一组为已求出最短路径的顶点集合，用S表示，初值为{v}，渐增至V。
    - 第二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依次把第二组的顶点加入S中
    - 在加入的过程中，总保持从源点v到S中各顶点的最短路径长度不大于从源点v到U中任何顶点的最短路径长度。



E.W. Dijkstra

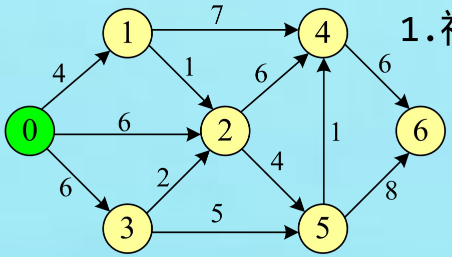




MIN($c_{vk}+w_{ku}$，c_{vu})

算法过程(动态规划法)

辅助存储
- s-是否已经加入S；
- dist - 距离；
- path - 记录路径



1.初值

	[0]	[1]	[2]	[3]	[4]	[5]	[6]
s	1	0	0	0	0	0	0
dist	0	4	6	6	∞	∞	∞
path	0	0	0	0	-1	-1	-1

2．加入①--s[1]=1，并修正①到达的点
dist[2]=min{dist[2],dist[1]+1}=5, path[2]=1
dist[4]=min{dist[4],dist[1]+7}=11, path[4]=1

	[0]	[1]	[2]	[3]	[4]	[5]	[6]
s	1	1	0	0	0	0	0
dist	0	4	5	6	11	∞	∞
path	0	0	1	0	1	-1	-1

3．加入②--s[2]=1，并修正②到达的点
dist[4]=min{dist[4],dist[2]+6}=11, path[2]不变
dist[5]=min{dist[5],dist[2]+4}=9, path[5]=2

	[0]	[1]	[2]	[3]	[4]	[5]	[6]
s	1	1	1	0	0	0	0
dist	0	4	5	6	11	9	∞
path	0	0	1	0	1	2	-1

4．加入③--s[3]=1，并修正③到达的点
dist[2]=min{dist[2],dist[3]+2}=5, path[2]不变
dist[5]=min{dist[5],dist[3]+5}=9, path[5]不变

	[0]	[1]	[2]	[3]	[4]	[5]	[6]
s	1	1	1	1	0	0	0
dist	0	4	5	6	11	9	∞
path	0	0	1	0	1	2	-1

5．加入⑤--s[5]=1，并修正⑤到达的点
dist[4]=min{dist[4],dist[5]+1}=10, path[4]=5
dist[6]=min{dist[6],dist[5]+8}=17, path[5]=5

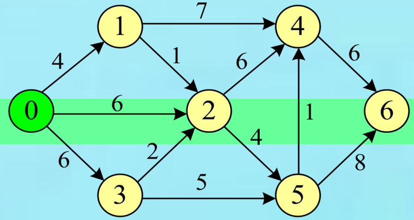
	[0]	[1]	[2]	[3]	[4]	[5]	[6]
s	1	1	1	1	0	1	0
dist	0	4	5	6	10	9	17
path	0	0	1	0	5	2	5

6．加入④--s[4]=1，并修正④到达的点
dist[6]=min{dist[6],dist[4]+6}=16, path[6]=4

	[0]	[1]	[2]	[3]	[4]	[5]	[6]
s	1	1	1	1	4	1	1
dist	0	4	5	6	10	9	16
path	0	0	1	0	5	2	4

狄克斯特拉算法实现

```
void Dijkstra(MGraph g,int v)
{
   //定义辅助存储及赋初值
   for (i=0; i<g.n; i++)
   {
      //选取不在s中且具有最小距离的顶点u
      s[u]=1;   //顶点u加入s中
      //修改不在s中的顶点的距离
   }
   Dispath(...);  //输出最短路径
}
```



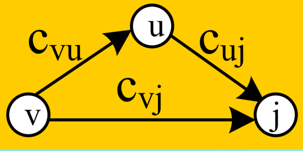
$$\begin{bmatrix} 0 & 4 & 6 & 6 & \infty & \infty & \infty \\ \infty & 0 & 1 & \infty & 7 & \infty & \infty \\ \infty & \infty & 0 & \infty & 6 & 4 & \infty \\ \infty & \infty & 2 & 0 & \infty & 5 & \infty \\ \infty & \infty & \infty & \infty & 0 & \infty & 6 \\ \infty & \infty & \infty & \infty & 1 & 0 & 8 \\ \infty & \infty & \infty & \infty & \infty & \infty & 0 \end{bmatrix}$$

```
int mindis,i,j,u;
int s[MAXV];
int dist[MAXV],path[MAXV];
for (i=0; i<g.n; i++)
{
   s[i]=0;
   dist[i]=g.edges[v][i];
   if (g.edges[v][i]<INF)
      path[i]=v;
   else
      path[i]=-1;
}
s[v]=1;
path[v]=0;
```

```
mindis=INF;
for (j=0; j<g.n; j++)
   if (s[j]==0 && dist[j]<mindis)
   {
      u=j;
      mindis=dist[j];
   }
```

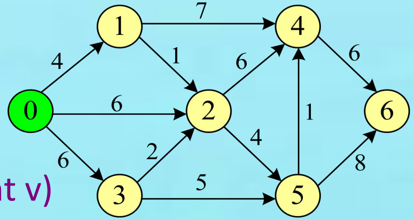
时间复杂度: $O(n^2)$

```
for (j=0; j<g.n; j++)
   if (s[j]==0)
      if (g.edges[u][j]<INF && dist[u]+g.edges[u][j]<dist[j])
      {
         dist[j]=dist[u]+g.edges[u][j];
         path[j]=u;
      }
```



	[0]	[1]	[2]	[3]	[4]	[5]	[6]
s	1	1	1	1	1	1	1
dist	0	4	5	6	10	9	16
path	0	0	1	0	5	2	4

输出结果



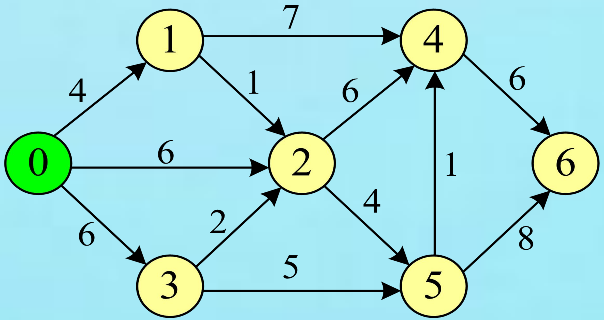

```
void Dispath(int dist[],int path[],int s[],int n,int v)
{
   int i;
   for (i=0; i<n; i++)
      if (s[i]==1)
      {
         printf(" 从%d到%d的最短路径长度为:%d\t路径为:",v,i,dist[i]);
         printf("%d,",v);
         Ppath(path,i,v);
         printf("%d\n",i);
      }
      else
         printf("从%d到%d不存在路径\n",v,i);
}
```

	[0]	[1]	[2]	[3]	[4]	[5]	[6]
path	0	0	1	0	5	2	4

```
void Ppath(int path[],int i,int v)
{
   int k;
   k=path[i];
   if (k==v)
      return;
   Ppath(path,k,v);
   printf("%d,",k);
}
```


D:\CB\DS\bin\Debug\ds.exe

```
从0到0的最短路径长度为:0      路径为:0,0
从0到1的最短路径长度为:4      路径为:0,1
从0到2的最短路径长度为:5      路径为:0,1,2
从0到3的最短路径长度为:6      路径为:0,3
从0到4的最短路径长度为:10     路径为:0,1,2,5,4
从0到5的最短路径长度为:9      路径为:0,1,2,5
从0到6的最短路径长度为:16     路径为:0,1,2,5,4,6
```



狄克斯特拉算法的过程

S	U	dist[]	path[]
		0 1 2 3 4 5 6	0 1 2 3 4 5 6
{0}	{1,2,3,4,5,6}	{0, 4, 6, 6, ∞, ∞,∞}	{0, 0, 0, 0, -1, -1, -1}
{0,1}	{2,3,4,5,6}	{0, 4, 5, 6, 11, ∞,∞}	{0, 0, 1, 0, 1, -1, -1}
{0,1,2}	{3,4,5,6}	{0, 4, 5, 6, 11, 9, ∞}	{0, 0, 1, 0, 1, 2, -1}
{0,1,2,3}	{4,5,6}	{0, 4, 5, 6, 11, 9, ∞}	{0, 0, 1, 0, 1, 2, -1}
{0,1,2,3,5}	{4,6}	{0, 4, 5, 6, 10, 9, 17}	{0, 0, 1, 0, 5, 2, 5}
{0,1,2,3,5,4}	{6}	{0, 4, 5, 6, 10, 9, 16}	{0, 0, 1, 0, 5, 2, 4}
{0,1,2,3,5,4,6}	{}	{0, 4, 5, 6, 10, 9, 16}	{0, 0, 1, 0, 5, 2, 4}

s[]={1,1,1,1,0,1,0}