本节主题：

DFS的应用

1. 是否有简单路径？

- 基础
  - **简单路径：**除开始点和结束点可以相同外，其余顶点均不相同的路径。
- 问题
  - 假设图G采用邻接表存储，设计一个算法，判断顶点u到v是否有简单路径。
- 策略
  - 从顶点u开始进行深度优先搜索
  - 当搜索到顶点v时，表明从顶点u到顶点v有路径，且一定为简单路径


```
int visited[MAXV];    //全局数组，调用前置为全0
void ExistPath(ALGraph *G,int u,int v, bool &has)
{
   int w;
   ArcNode *p;
   visited[u]=1;
   if(u==v)
   {
      has=true;
      return;
   }
   p=G->adjlist[u].firstarc;
   while (p!=NULL)
   {
      w=p->adjvex;
      if (visited[w]==0)
         ExistPath(G,w,v,has);
      p=p->nextarc;
   }
}
```

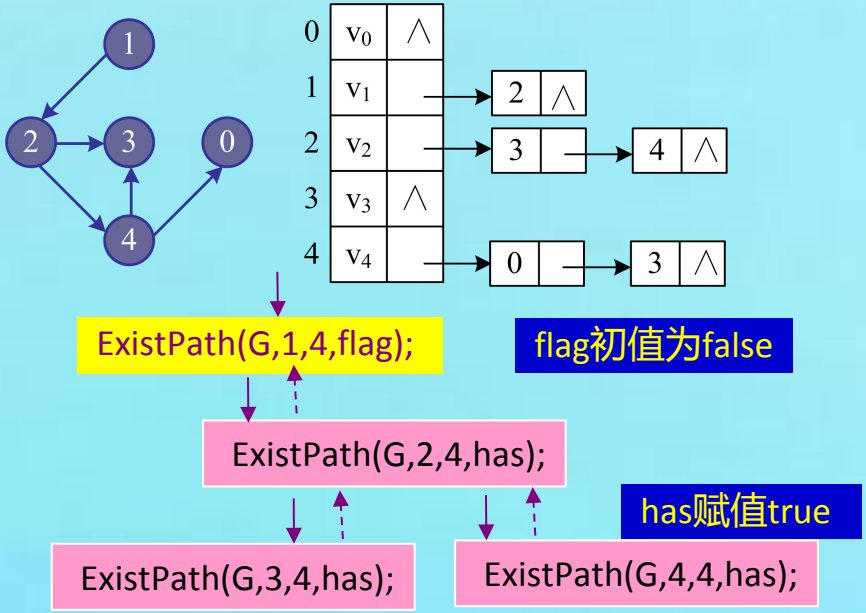
```
void HasPath(ALGraph *G,int u,int v)
{
   int i;
   bool flag = false;
   for (i=0; i<G->n; i++)
      visited[i]=0;
   ExistPath(G,u,v,flag);
   if(flag)
      printf("有\n");
   else
      printf("无\n");
}
```

“是否有简单路径”的算法执行过程

```
int visited[MAXV];    //全局数组，调用前置为全0
void ExistPath(ALGraph *G,int u,int v, bool &has)
{
   int w;
   ArcNode *p;
   visited[u]=1;
   if(u==v)
   {
      has=true;
      return;
   }
   p=G->adjlist[u].firstarc;
   while (p!=NULL)
   {
      w=p->adjvex;
      if (visited[w]==0)
         ExistPath(G,w,v,has);
      p=p->nextarc;
   }
}
```

int visited[N];

[0]	0
[1]	1
[2]	1
[3]	1
[4]	1



改进点：

- has一旦为true，即退出

2. 输出简单路径

- 问题
  - 假设图G采用邻接表存储，设计一个算法输出图G中从顶点u到v的一条简单路径（假设图G中从顶点u到v至少有一条简单路径）。
- 策略
  - 采用深度优先遍历的方法。
  - 在深度优先遍历算法的基础上增加形参
    - path存放顶点u到v的路径
    - d表示path中的路径长度，其初值为-1。
  - 当从顶点u遍历到顶点v后，输出path并返回。

```
int visited[MAXV];
void FindAPath(ALGraph *G,int u,int v,int path[],int d)
{
   int w,i;
   ArcNode *p;
   visited[u]=1;
   d++;
   path[d]=u;
   if (u==v)
   {
      //找到一条路径后输出并返回
   }
   p=G->adjlist[u].firstarc;
   while (p!=NULL)
   {
      w=p->adjvex;
      if (visited[w]==0)
         FindAPath(G,w,v,path,d);
      p=p->nextarc;
   }
}
```

```
printf("一条简单路径为:");
for (i=0; i<=d; i++)
   printf("%d ",path[i]);
printf("\n");
return;
```

```
void APath(ALGraph *G,int u,int v)
{
   int i;
   int path[MAXV];
   for (i=0; i<G->n; i++)
      visited[i]=0;
   FindAPath(G,u,v,path,-1);
}
```

3. 输出所有简单路径

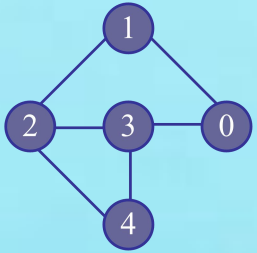
- 问题
  - 输出从顶点u到v的**所有**简单路径。
- 策略
  - 利用回溯的深度优先搜索方法。
  - 从顶点u开始进行深度优先搜索，在搜索过程中
    - 设立一个数组path保存走过的路径，把当前的搜索线路记录下来
    - 用d记录走过的路径长度。
  - 若当前扫描到的顶点u等于v时，表示找到了一条路径，则输出路径path。
  - 处理完一个顶点，将其标记为未访问，以寻找下一种可能。

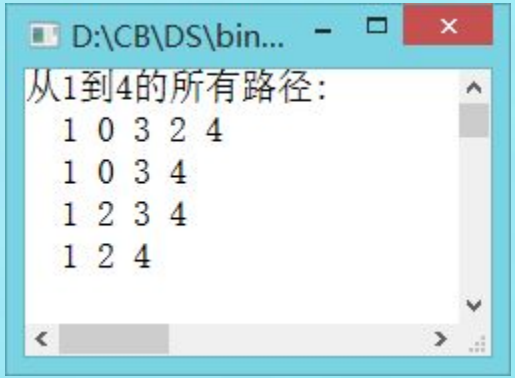
```
int visited[MAXV];
void FindPaths(ALGraph *G,int u,int v,int path[],int d)
{
   int w,i;
   ArcNode *p;
   visited[u]=1;
   d++;
   path[d]=u;
   if (u==v && d>1)
   {
      //输出找到的路径
   }
   p=G->adjlist[u].firstarc;
   while(p!=NULL)
   {
      w=p->adjvex;
      if (visited[w]==0)
         FindPaths(G,w,v,path,d);
      p=p->nextarc;
   }
   visited[u]=0;
}
```

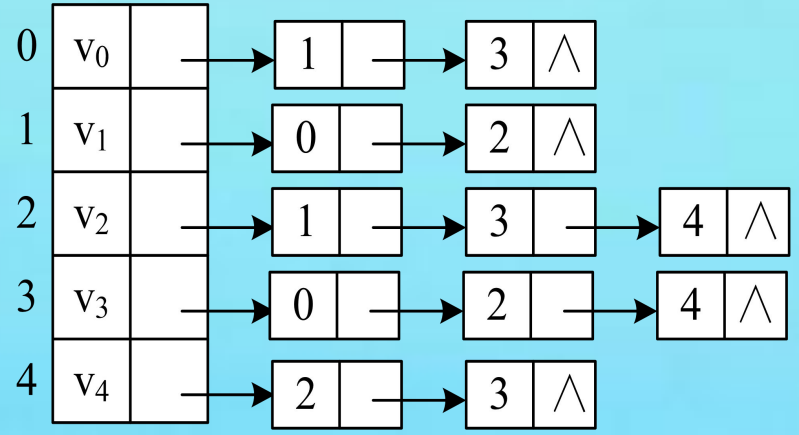
```
printf("  ");
for (i=0; i<=d; i++)
    printf("%d ",path[i]);
printf("\n");
```

体现回溯——这一条路径不算，继续找下一种可能

FindPaths(G,1,4,path,-1);








```
int visited[MAXV];
void FindPaths(ALGraph *G,int u,int v,int path[],int d)
{
   int w,i;
   ArcNode *p;
   visited[u]=1;
   d++;
   path[d]=u;
   if (u==v && d>1)
   {
      //输出找到的路径
   }
   p=G->adjlist[u].firstarc;
   while(p!=NULL)
   {
      w=p->adjvex;
      if (visited[w]==0)
         FindPaths(G,w,v,path,d);
      p=p->nextarc;
   }
   visited[u]=0;
}
```

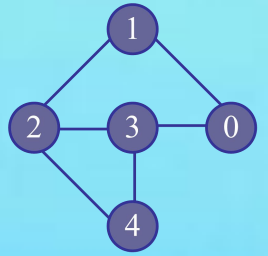
```
printf("  ");
for (i=0; i<=d; i++)
    printf("%d ",path[i]);
printf("\n");
```

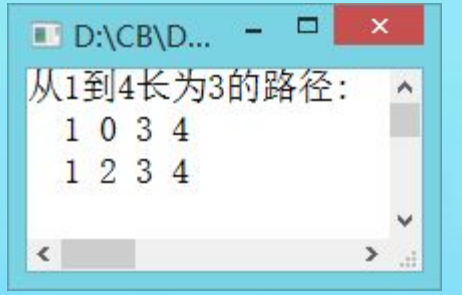
体现回溯——这一条路径不算，继续找下一种可能

4. 输出某些简单路径

- 问题
  - 输出图G中从顶点u到v的长度为s的所有简单路径。
- 策略
  - 完全同输出所有路径，只需在输出时限制长度

```
SomePaths(G,1,4,3,path,-1);
```



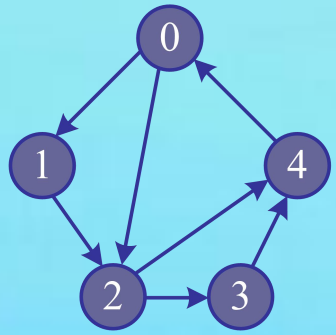


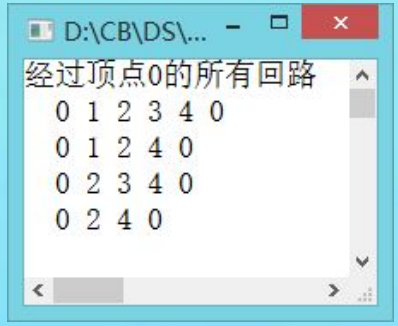

```
int visited[MAXV];
void SomePaths(ALGraph *G,int u,int v,int s,int path[],int d)
{
   int w,i;
   ArcNode *p;
   visited[u]=1;
   d++;
   path[d]=u;
   if (u==v && d==s)
   {
      //输出找到的路径
   }
   p=G->adjlist[u].firstarc;
   while(p!=NULL)
   {
      w=p->adjvex;
      if (visited[w]==0)
         SomePaths(G,w,v,s,path,d);
      p=p->nextarc;
   }
   visited[u]=0;
}
```

```
printf("  ");
for (i=0; i<=d; i++)
   printf("%d ",path[i]);
printf("\n");
```

5.输出过某点的所有回路

- 问题
  - 求图中通过某顶点k的所有简单回路（若存在）
- 策略
  - 发现下一个为指定点时再输出。






```
int visited[MAXV];
void DFSPath(ALGraph *G,int u,int v,int path[],int d)
{
   int w,i;
   ArcNode *p;
   visited[u]=1;
   d++;
   path[d]=u;
   p=G->adjlist[u].firstarc;
   while (p!=NULL)
   {
      w=p->adjvex;
      if (w==v && d>0)
      {
         //找到一个回路，输出之
      }
      if (visited[w]==0)
         DFSPath(G,w,v,path,d);
      p=p->nextarc;
   }
   visited[u]=0;
}
```

```
void FindCyclePath(ALGraph *G,int k)
{
   int path[MAXV],i;
   for (i=0; i<G->n; i++)
      visited[i]=0;
   printf("经顶点%d的所有回路\n",k);
   DFSPath(G,k,k,path,-1);
   printf("\n");
}
```

```
printf("  ");
for (i=0; i<=d; i++)
      printf("%d ",path[i]);
printf("%d \n",v);
```