主讲：胡燕

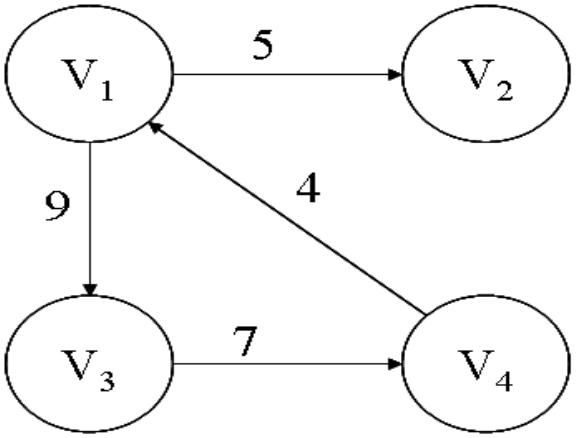
建立邻接表

（1）输入顶点个数和边的个数
（2）邻接表初始化

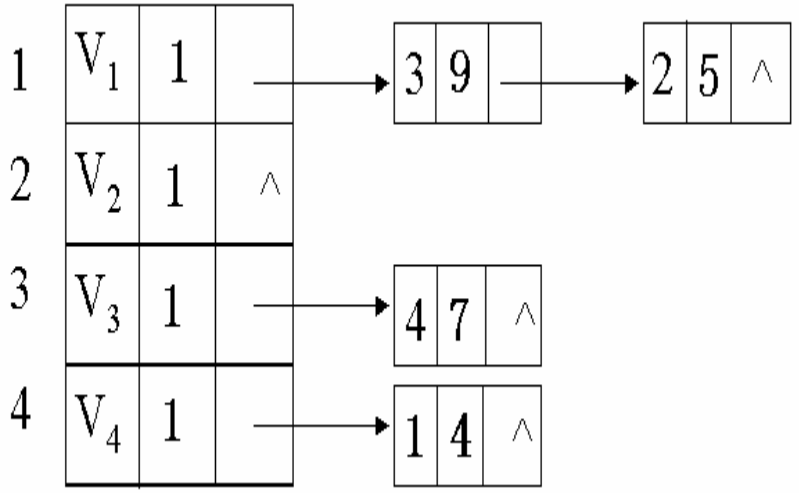
	顶点	入度	first
1	V_1	0	∧
2	V_2	0	∧
3	V_3	0	∧
4	V_4	0	∧

first：指向链表的第一个结点

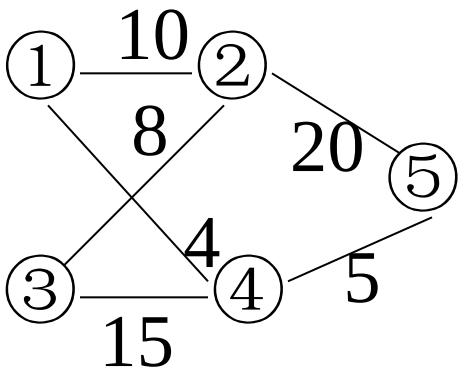
（3）输入e条边，构造邻接表



有向网络G2



建立邻接表(例)



(1)输入是否有向图
(2)输入顶点个数n
(3)邻接表的初始化

(4)循环输入各边并存入邻接表

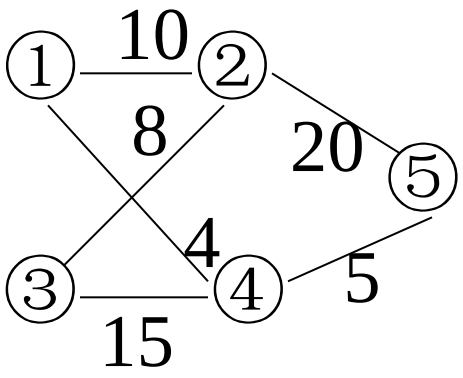
输入(i,j,w)
查询j结点是否已连上
建立新结点
插入j结点,j结点入度加1
若为无向图则插入i结点

1	0	^
2	0	^
3	0	^
4	0	^
5	0	^

10	2	∧

(1,2,10)

建立邻接表(例)



(1)输入是否有向图
(2)输入顶点个数n
(3)邻接表的初始化

(4)循环输入各边并存入邻接表

输入(i,j,w)
查询j结点是否已连上
建立新结点
插入j结点,j结点入度加1
若为无向图则插入i结点

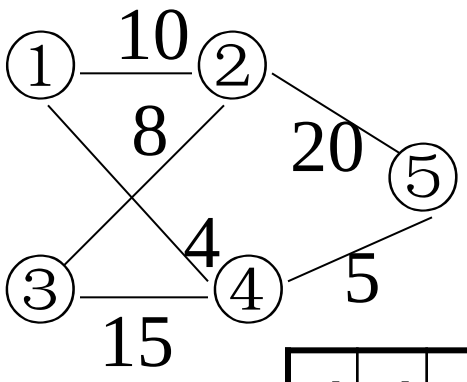
1	0	^
2	1	^
3	0	^
4	0	^
5	0	^

10	2	∧

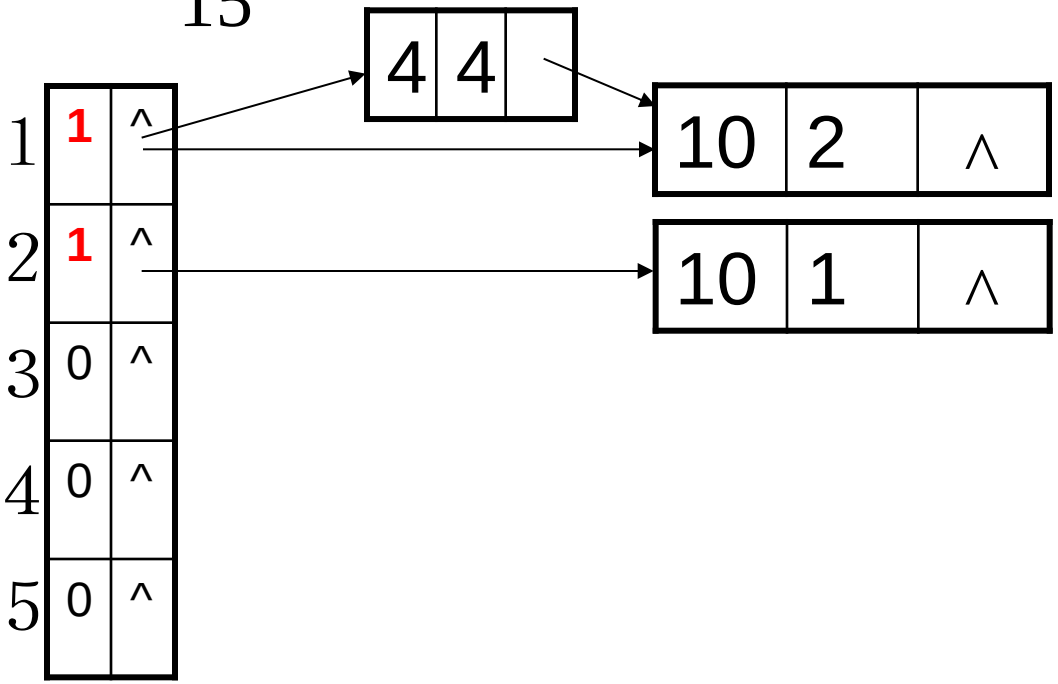
10	1	∧

(1,2,10)

建立邻接表(例)



(1)输入是否有向图
(2)输入顶点个数n
(3)邻接表的初始化



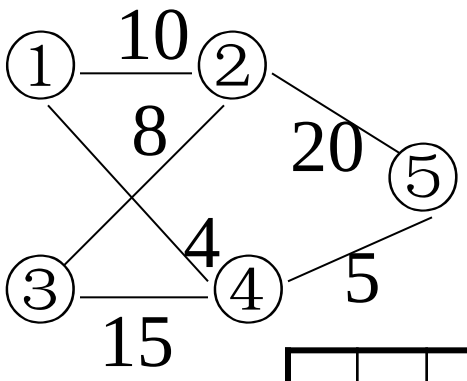
(4)循环输入各边并存入邻接表

输入(i,j,w)
查询j结点是否已连上
建立新结点
插入j结点,j结点入度加1
若为无向图则插入i结点

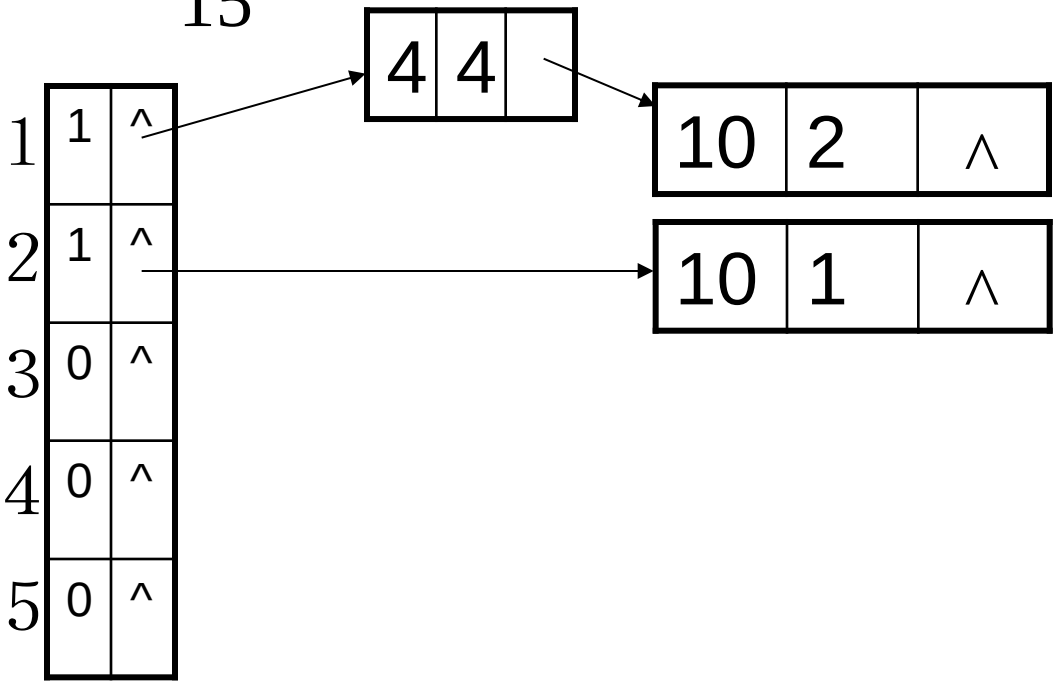
(1,2,10)

(1,4,4)

建立邻接表(例)



(1)输入是否有向图
(2)输入顶点个数n
(3)邻接表的初始化



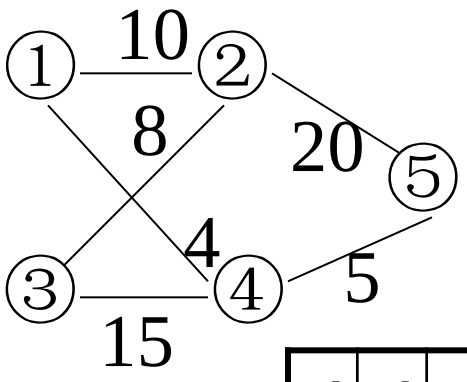
(4)循环输入各边并存入邻接表

输入(i,j,w)
查询j结点是否已连上
建立新结点
插入j结点,j结点入度加1
若为无向图则插入i结点

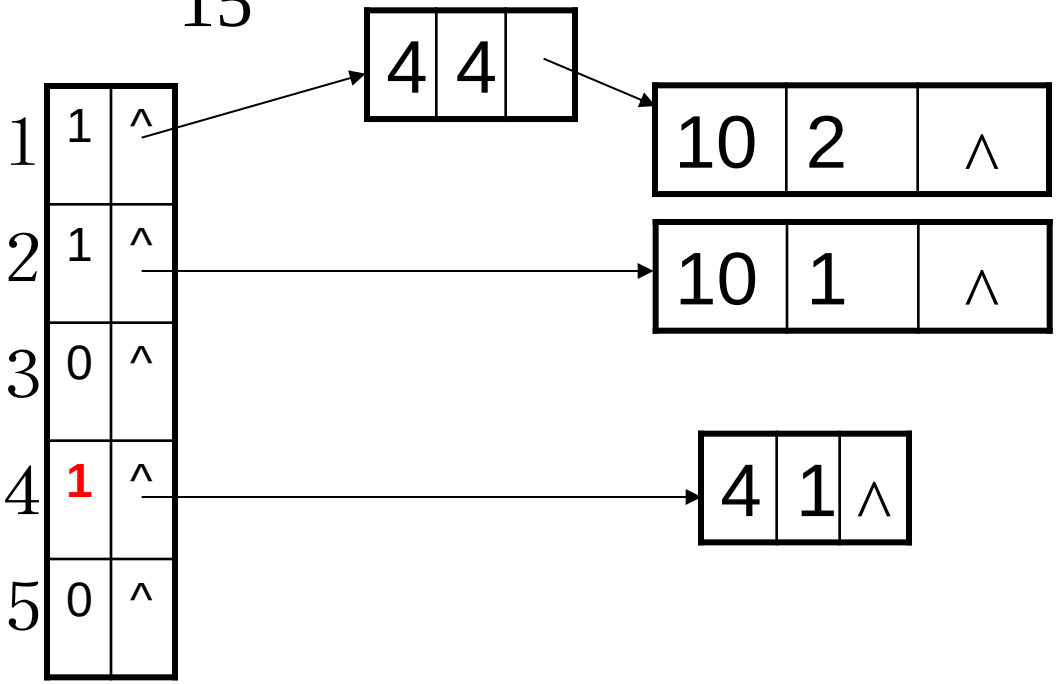
(1,2,10)

(1,4,4)

建立邻接表(例)



(1)输入是否有向图
(2)输入顶点个数n
(3)邻接表的初始化



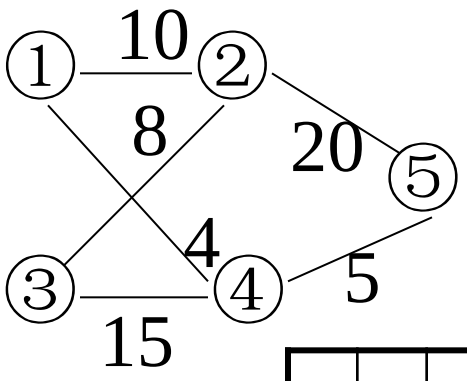
(4)循环输入各边并存入邻接表

输入(i,j,w)
查询j结点是否已连上
建立新结点
插入j结点,j结点入度加1
若为无向图则插入i结点

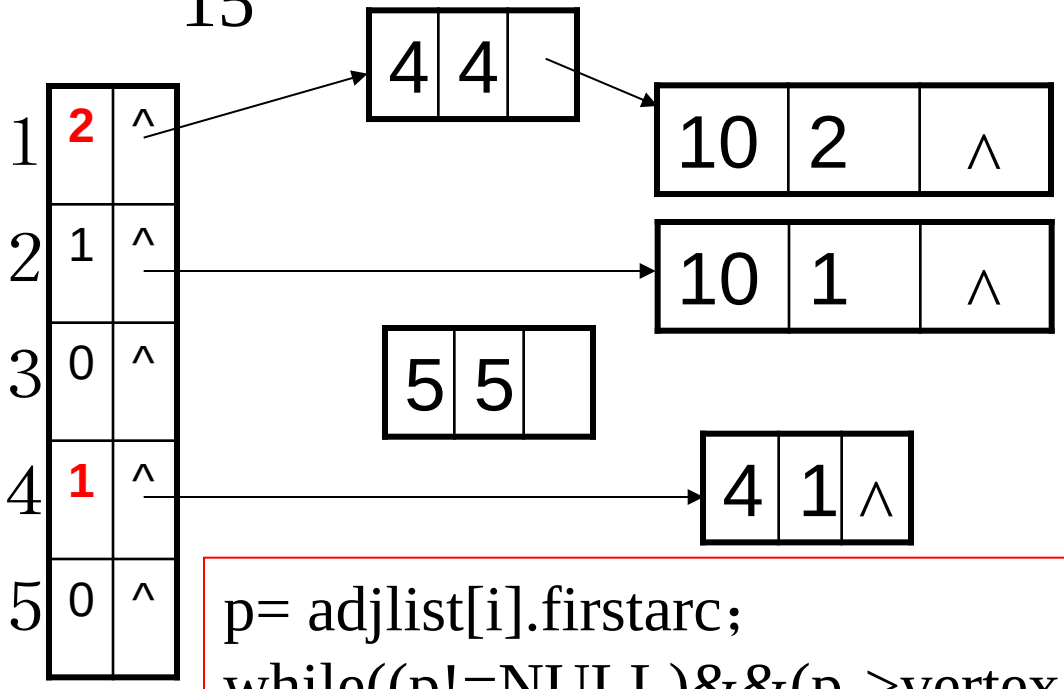
(1,2,10)

(1,4,4)

建立邻接表(例)



(1)输入是否有向图
(2)输入顶点个数n
(3)邻接表的初始化




```
p= adjlist[i].firstarc;
while((p!=NULL)&&(p->vertex!=j))
        p=p->next;
```

(4)循环输入各边并存入邻接表

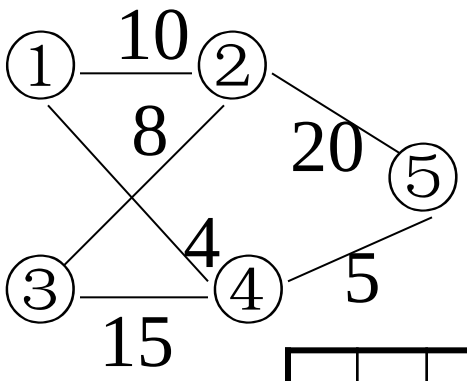
输入(i,j,w)
查询j结点是否已连上
建立新结点
插入j结点,j结点入度加1
若为无向图则插入i结点

(1,2,10)

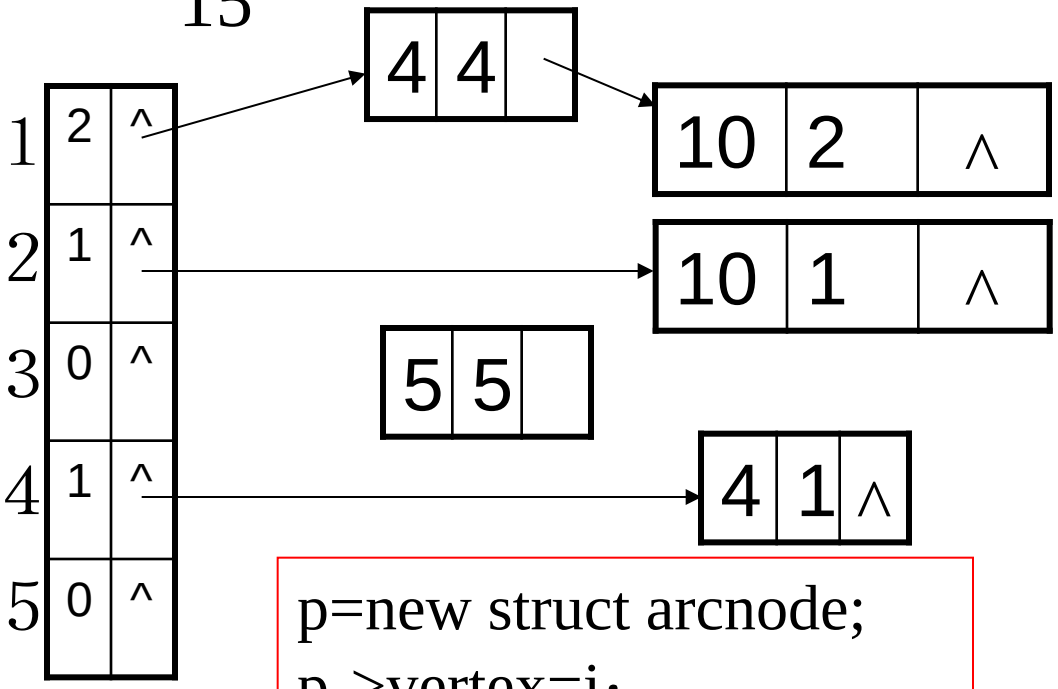
(1,4,4)

(4,5,5)

建立邻接表(例）



(1)输入是否有向图
(2)输入顶点个数n
(3)邻接表的初始化




```
p=new struct arcnode;
p->vertex=j;
p->info=w;
```

(4)循环输入各边并存入邻接表

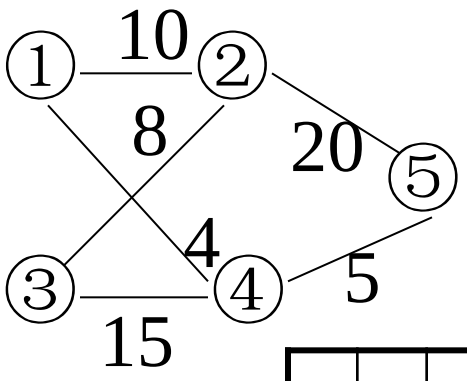
输入(i,j,w)
查询j结点是否已连上
建立新结点
插入j结点,j结点入度加1
若为无向图则插入i结点

(1,2,10)

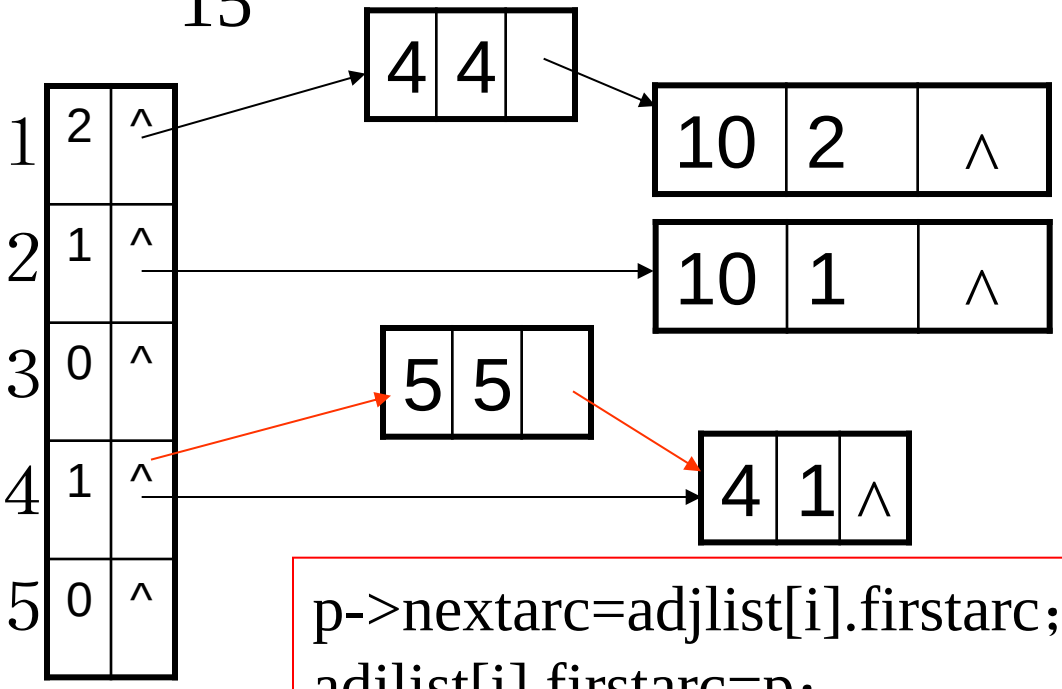
(1,4,4)

(4,5,5)

建立邻接表(例)



(1)输入是否有向图
(2)输入顶点个数n
(3)邻接表的初始化




```
p->nextarc=adjlist[i].firstarc;
adjlist[i].firstarc=p;
++(adjlist[j].degree
```

(4)循环输入各边并存入邻接表

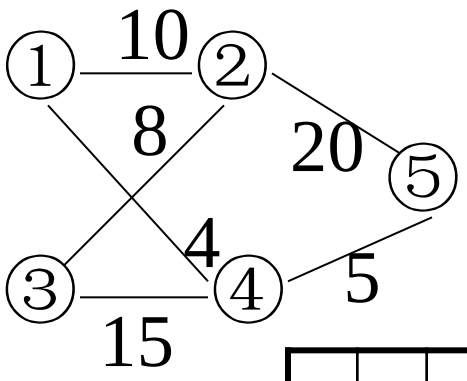
输入(i,j,w)
查询j结点是否已连上
建立新结点
插入j结点,j结点入度加1
若为无向图则插入i结点

(1,2,10)

(1,4,4)

(4,5,5)

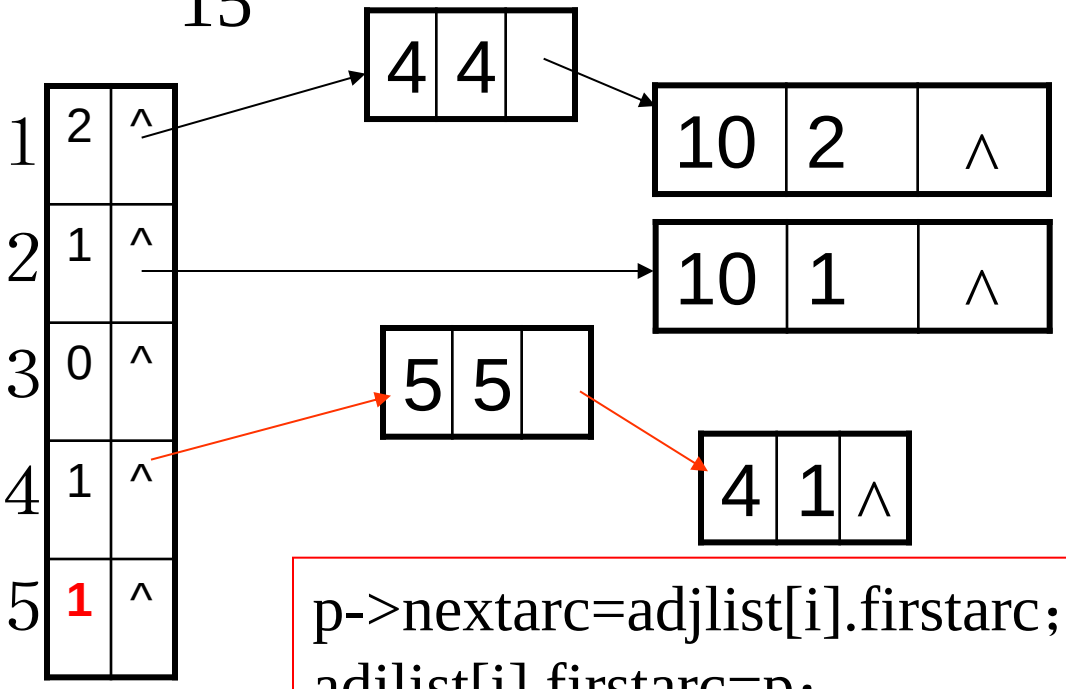
建立邻接表(例)



(1)输入是否有向图
(2)输入顶点个数n
(3)邻接表的初始化

(4)循环输入各边并存入邻接表

输入(i,j,w)
查询j结点是否已连上
建立新结点
插入j结点,j结点入度加1
若为无向图则插入i结点




```
p->nextarc=adjlist[i].firstarc;
adjlist[i].firstarc=p;
++(adjlist[j].degree
```

(1,2,10)

(1,4,4)

(4,5,5)

建立邻接表(例）

```
#define n0 100
struct arcnode {
    int vertex;
    int info;
    struct arcnode *nextarc;
    };
struct node{
    int degree;
    struct acrnode *firstarc;
   };
struct node adjlist[n0+1];
int n;
void setgraph()
{  int i，j，k，w;
    struct arcnode *p;
```

```
printf( “1:无向网 2:有向网” );
scanf(“%d”,&k);
printf( “顶点数” );
scanf(“%d”,&n);
for(i=l；i<=n；i++)
   { adjlist[i].degree=0；
     adjlist[i].first=NULL；
    };
printf( “输入边集合” );
scanf(“%d,%d,%d”,&i,&j,&w);
while((i>=1)&&(i<=n)
      &&(j>=1)&&(j<=n))
 {p= adjlist[i].firstarc；
  while((p!=NULL)&&(p->vertex!=j))
        p=p->nextarc；
```

建立邻接表(例）

```
if p==NULL)
   {    p=(struct arcnode*)malloc(sizeof(struct arcnode));
        p->vertex=j;
        p->info=w;
        p->nextarc=adjlist[i].firstarc;
        adjlist[i].firstarc=p;
       ++(adjlist[j].degree);
       if(k==l  )
         { p= (struct arcnode*)malloc(sizeof(struct arcnode));
           p->vertex=i;
           p->info=w;
           p->nextarc= adjlist[j].firstarc;
           adjlist[j].firstarc=p;
          ++(adjlist[i].degree);     }     }
scanf("%d,%d,%d",&i,&j,&w);   }    }
```