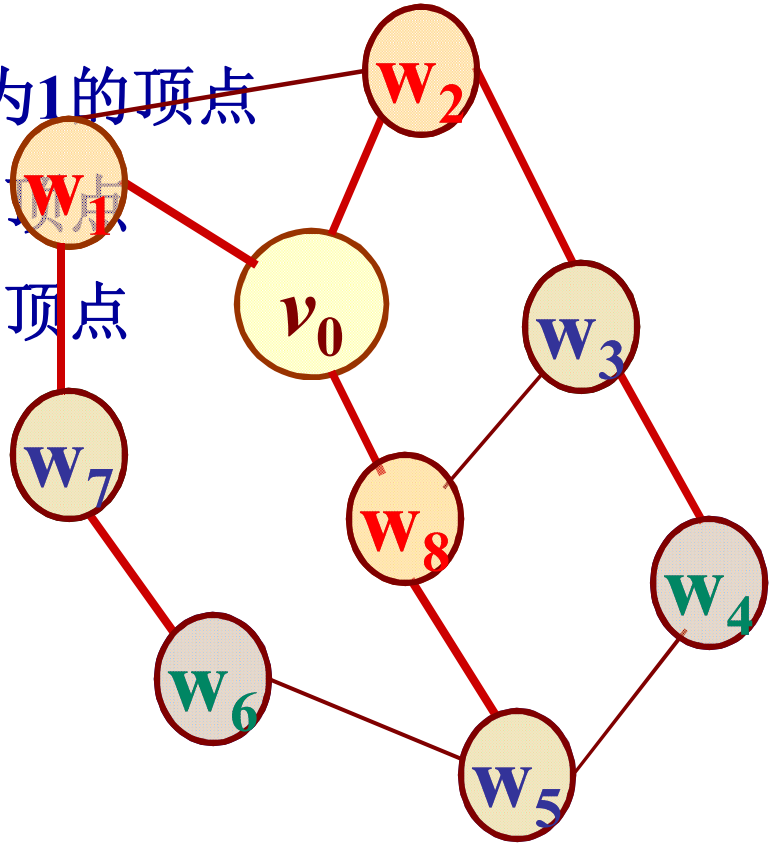
7.3 图的遍历--广度优先搜索

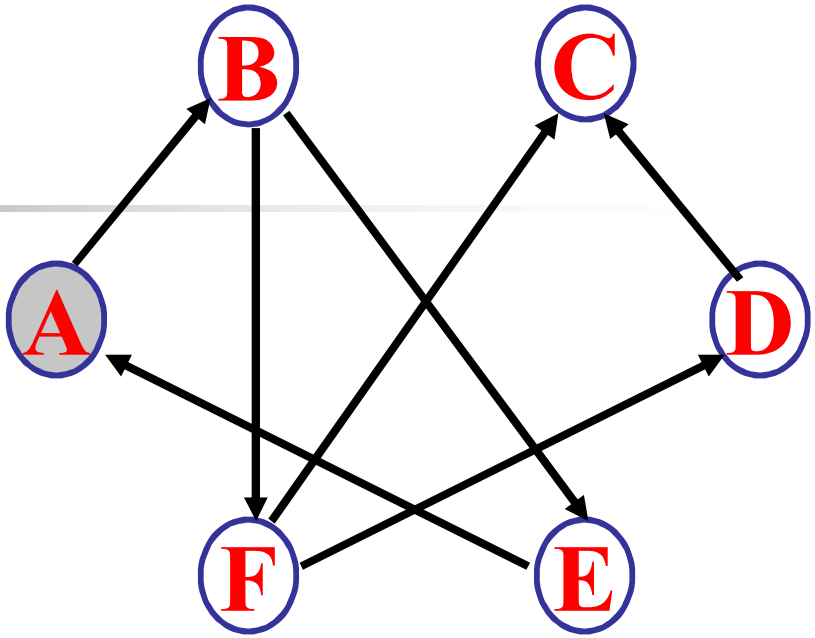
- 从图中的某个顶点v_0出发，并在访问此顶点之后依次访问v_0的所有未被访问过的邻接点，之后按这些顶点被访问的先后次序依次访问它们的邻接点，直至图中所有和v_0有路径相通的顶点都被访问到。
- 若此时图中尚有顶点未被访问，则另选图中一个未曾被访问的顶点作起始点，重复上述过程，直至图中所有顶点都被访问到为止。

7.3 图的遍历--广度优先搜索

- 图的广度优先搜索：按照与出发点v_0路径长度递增的顺序访问顶点----
  - 首先访问与出发点v_0路径长度为1的顶点
  - 访问与出发点v_0路径长度为2的顶点
  - 访问与出发点v_0路径长度为3的顶点
  - ……

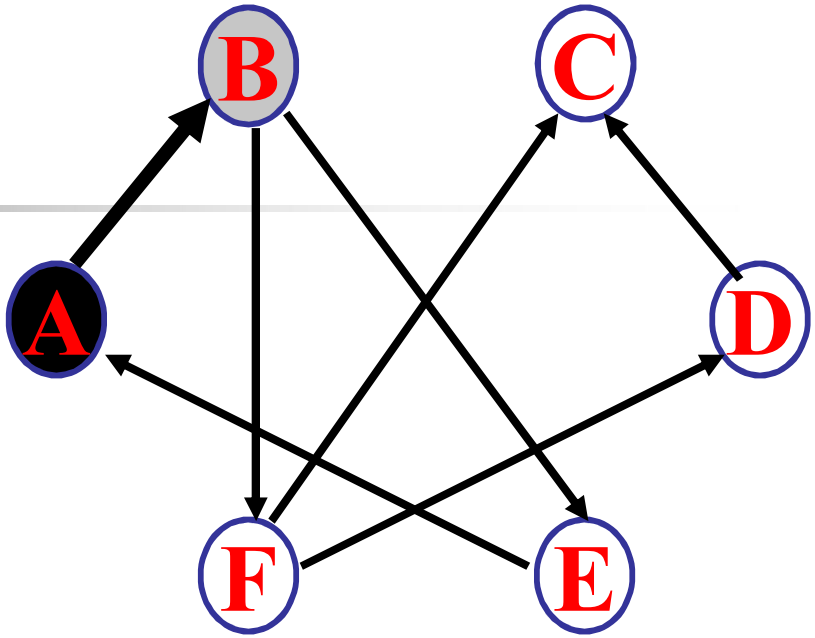


广度优先搜索



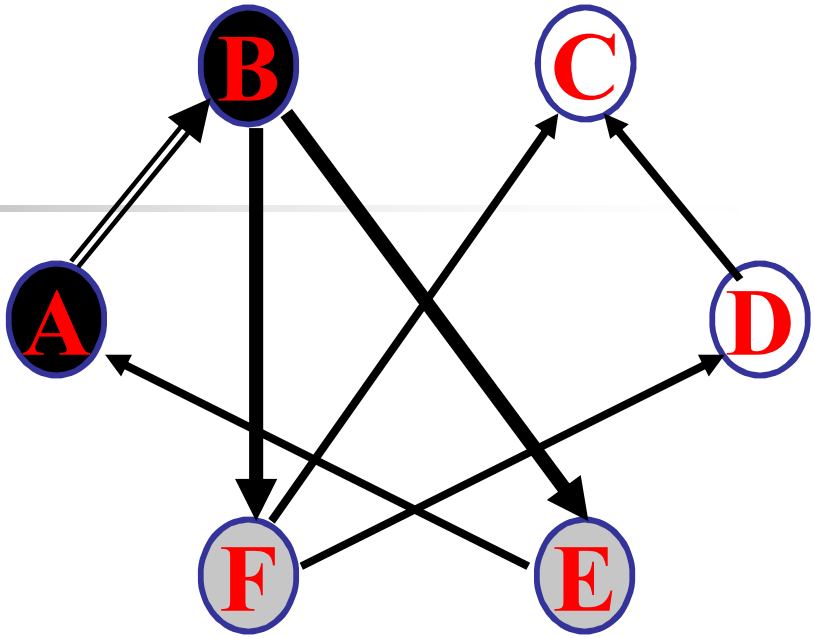
- A

广度优先搜索



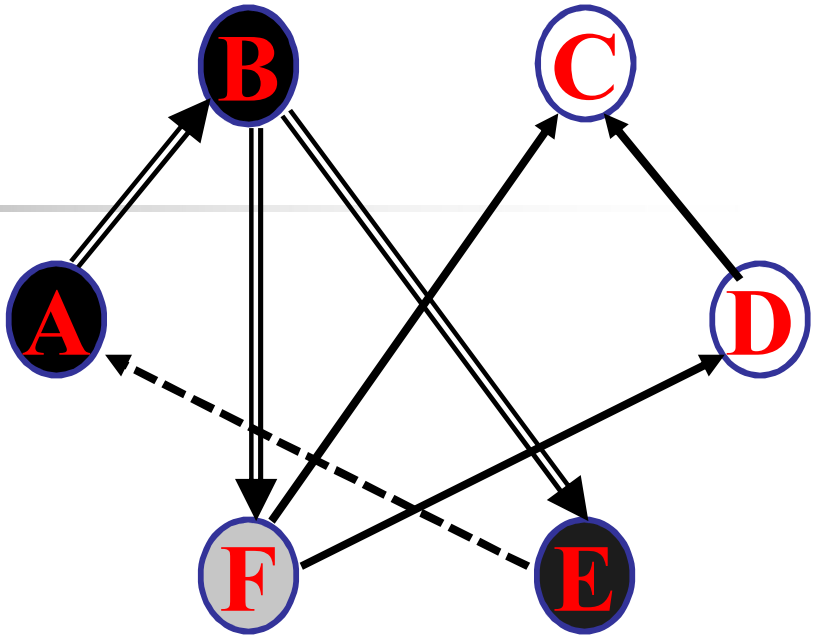
- A,B

广度优先搜索



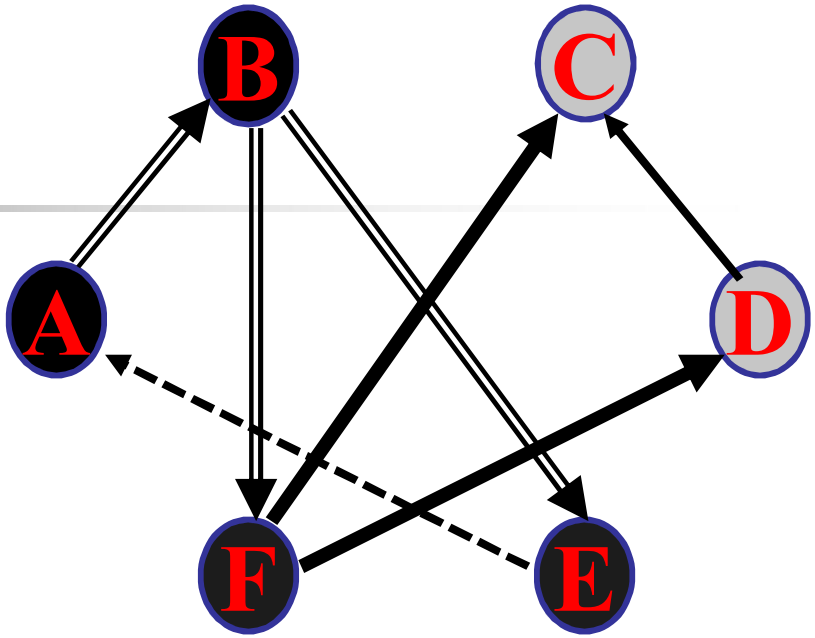
- A,B,E,F

广度优先搜索



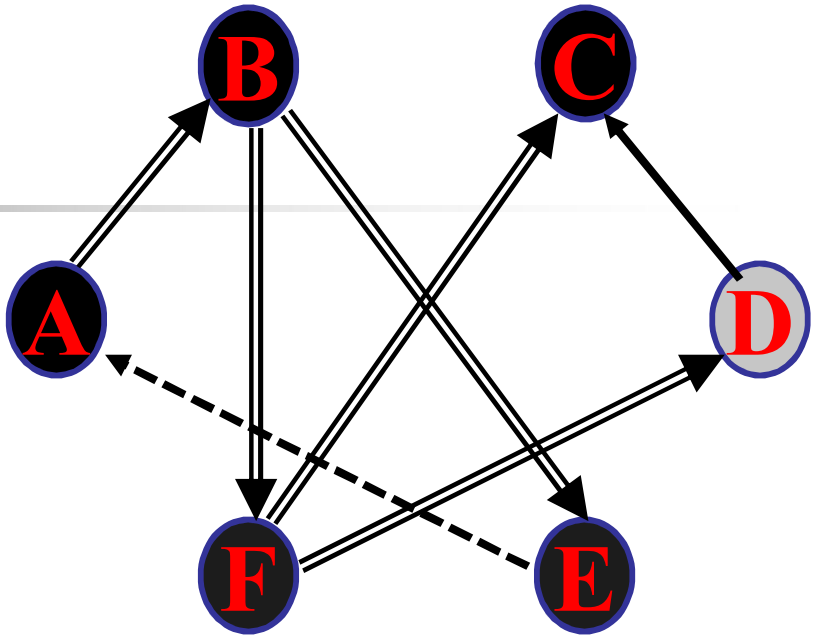
- A,B,E,F

广度优先搜索



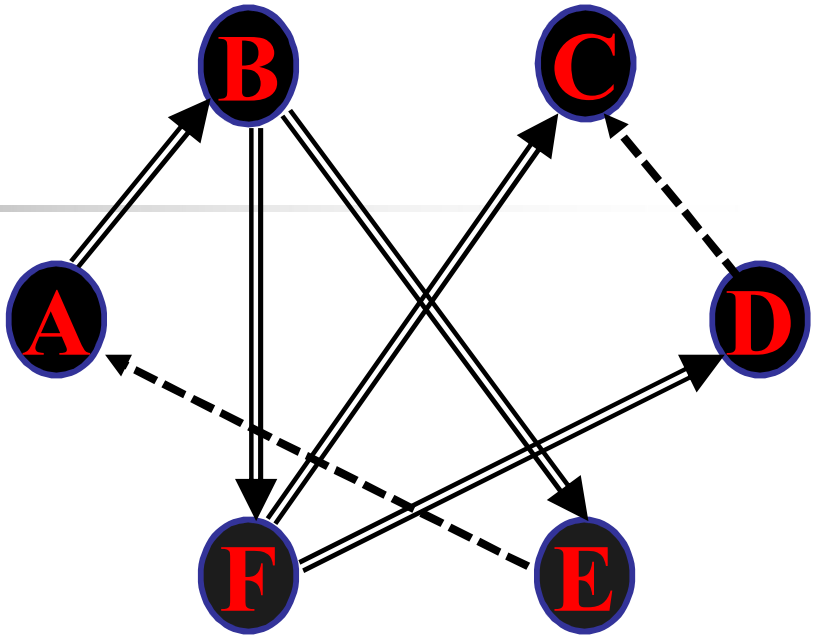
- A,B,E,F,C,D

广度优先搜索



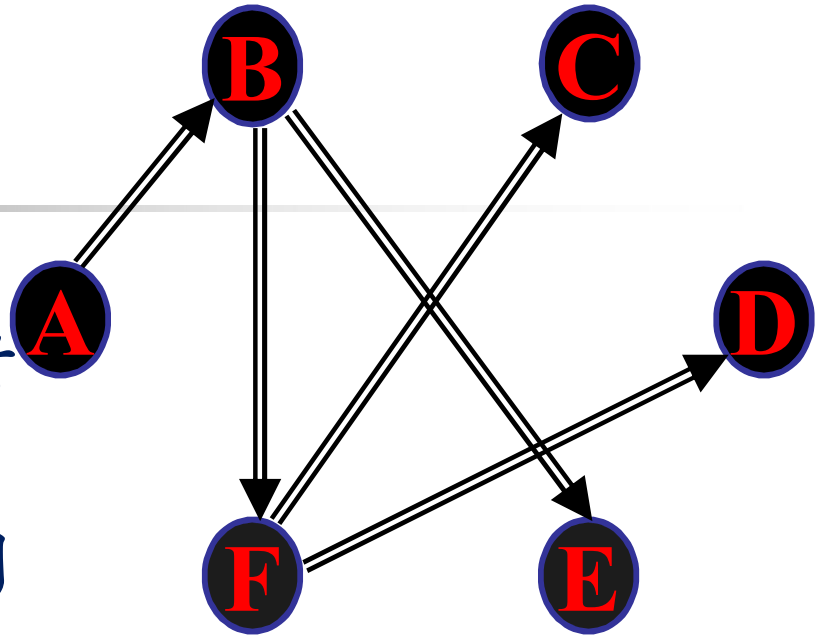
- A,B,E,F,C,D

广度优先搜索



- A,B,E,F,C,D

广度优先搜索

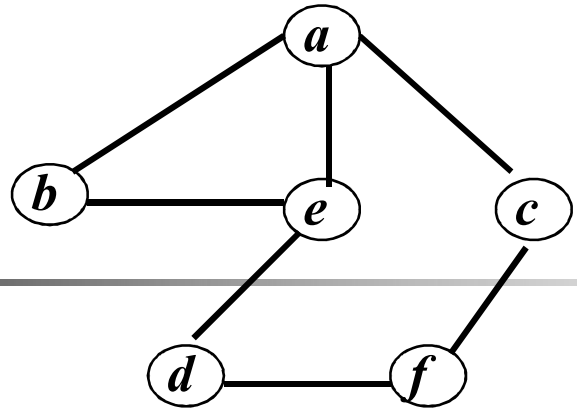


- **广度优先搜索生成树：** 访问时经过的顶点和边构成的子图
- **广度优先搜索生成森林：** 选用多个出发点做广度优先搜索，会产生多棵广度优先搜索生成树—构成广度优先搜索生成森林
- 对连通图，从起始点v到其余各顶点必定存在路径。按此路径长度递增次序访问

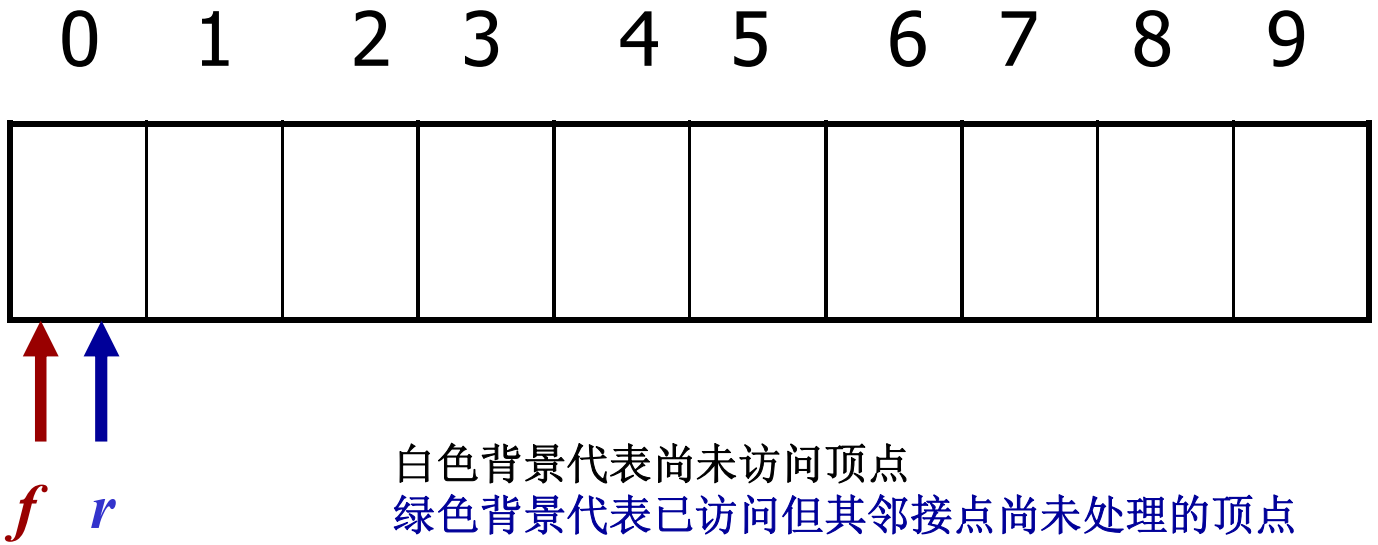
7.3 图的遍历--广度优先搜索

```
void BFSTraverse(Graphs G)
{
  for (v=0; v<G.vexnum; ++v)
        visited[v] = 0;
  for ( v=0;  v<G.vexnum;  ++v )
    if ( !visited[v]) BFS(G,v);
} // BFSTraverse
```

BFS(G,v):队列存放访问过但邻接点未处理的顶点

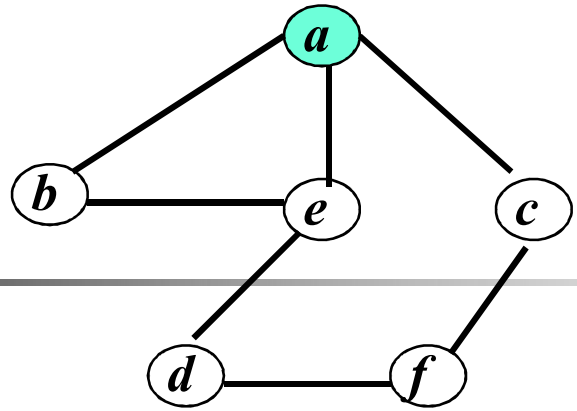


从a出发做广度优先搜索



白色背景代表尚未访问顶点
绿色背景代表已访问但其邻接点尚未处理的顶点
紫色背景代表正在处理其邻接点的顶点
黄色背景代表邻接点已经处理结束顶点

BFS(G,v):队列存放访问过但邻接点未处理的顶点



a

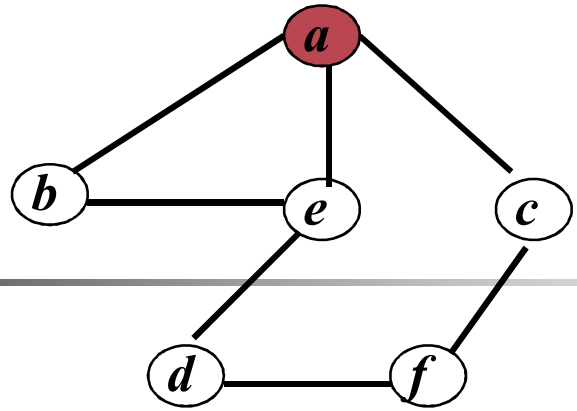
0	1	2	3	4	5	6	7	8	9
a									

f *r*

a

白色背景代表尚未访问顶点
绿色背景代表已访问但其邻接点尚未处理的顶点
紫色背景代表正在处理其邻接点的顶点
黄色背景代表邻接点已经处理结束顶点

BFS(G,v):队列存放访问过但邻接点未处理的顶点



a

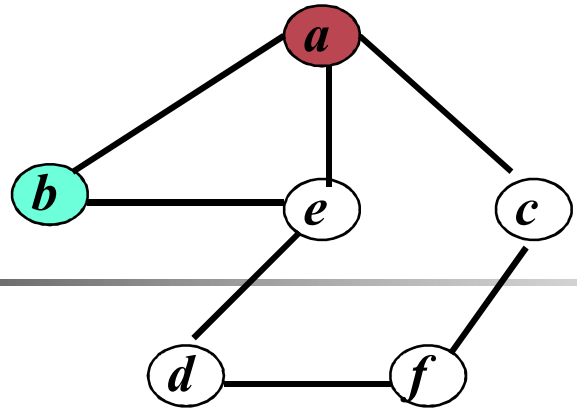
0	1	2	3	4	5	6	7	8	9
a									

f *r*

a

白色背景代表尚未访问顶点
绿色背景代表已访问但其邻接点尚未处理的顶点
紫色背景代表正在处理其邻接点的顶点
黄色背景代表邻接点已经处理结束顶点

BFS(G,v):队列存放访问过但邻接点未处理的顶点



ab

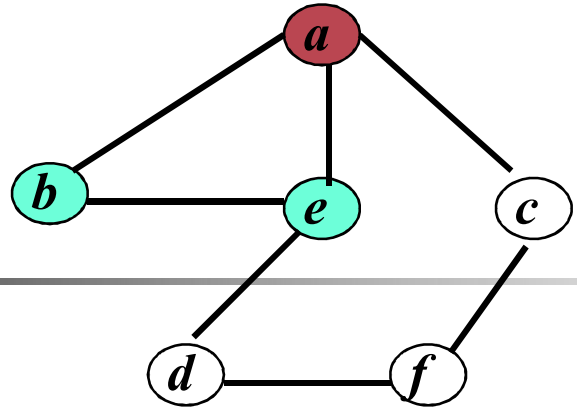
0	1	2	3	4	5	6	7	8	9
a	*b*								

f *r*

ab

白色背景代表尚未访问顶点
绿色背景代表已访问但其邻接点尚未处理的顶点
紫色背景代表正在处理其邻接点的顶点
黄色背景代表邻接点已经处理结束顶点

BFS(G,v):队列存放访问过但邻接点未处理的顶点



abe

0	1	2	3	4	5	6	7	8	9
a	*b*	*e*							

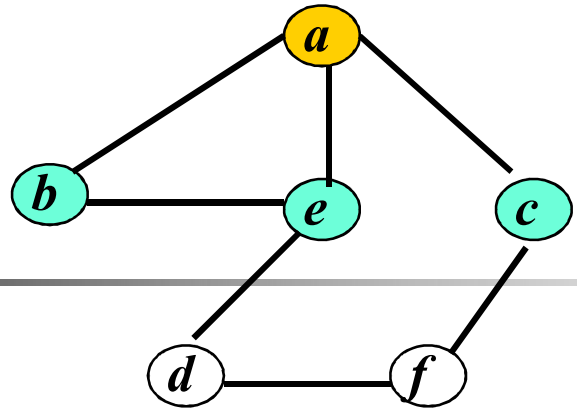
f

abe

r

白色背景代表尚未访问顶点
绿色背景代表已访问但其邻接点尚未处理的顶点
紫色背景代表正在处理其邻接点的顶点
黄色背景代表邻接点已经处理结束顶点

BFS(G,v):队列存放访问过但邻接点未处理的顶点



abec

0	1	2	3	4	5	6	7	8	9
a	*b*	*e*	*c*						

f

abec

r

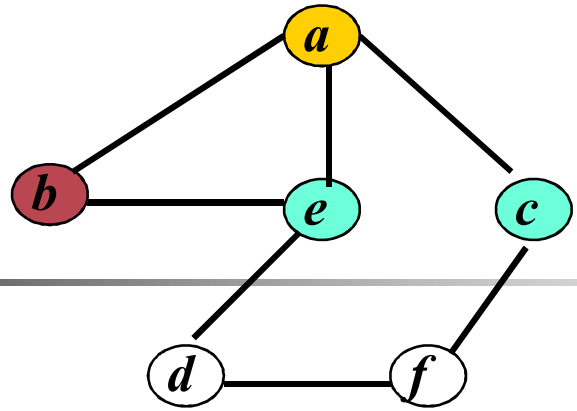
白色背景代表尚未访问顶点

绿色背景代表已访问但其邻接点尚未处理的顶点

紫色背景代表正在处理其邻接点的顶点

黄色背景代表邻接点已经处理结束顶点

BFS(G,v):队列存放访问过但邻接点未处理的顶点



abec

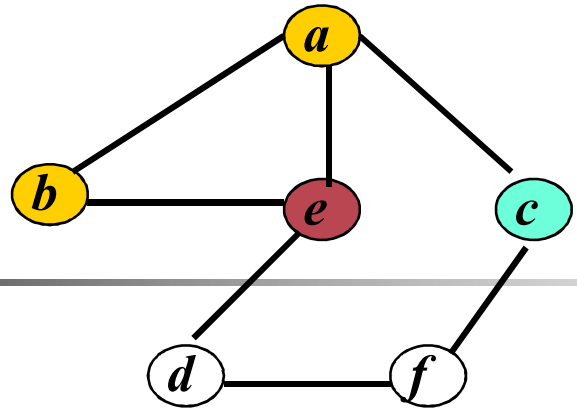
0	1	2	3	4	5	6	7	8	9
a	*b*	*e*	*c*						

f *r*

abec

白色背景代表尚未访问顶点
绿色背景代表已访问但其邻接点尚未处理的顶点
紫色背景代表正在处理其邻接点的顶点
黄色背景代表邻接点已经处理结束顶点

BFS(G,v):队列存放访问过但邻接点未处理的顶点



abec

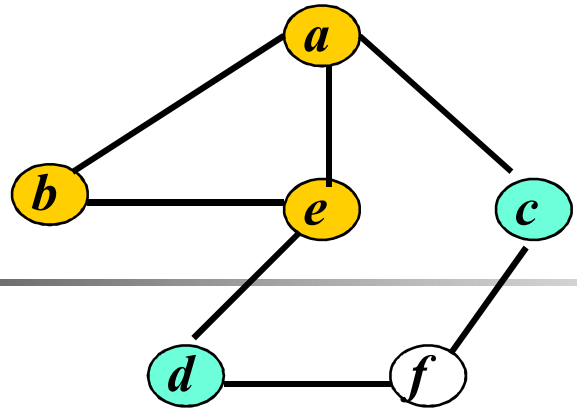
0	1	2	3	4	5	6	7	8	9
a	*b*	*e*	*c*						

f *r*

abec

白色背景代表尚未访问顶点
绿色背景代表已访问但其邻接点尚未处理的顶点
紫色背景代表正在处理其邻接点的顶点
黄色背景代表邻接点已经处理结束顶点

BFS(G,v):队列存放访问过但邻接点未处理的顶点



abecd

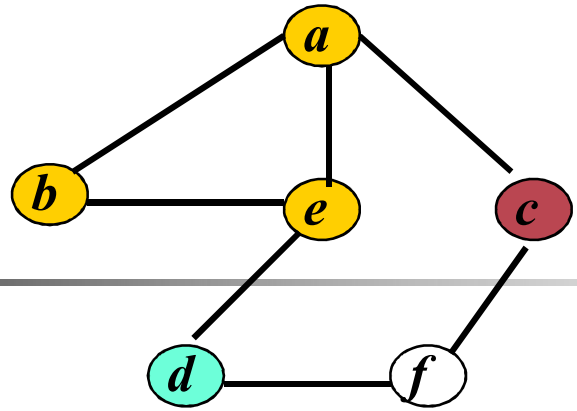
0	1	2	3	4	5	6	7	8	9
a	*b*	*e*	*c*	*d*					

f *r*

abecd

白色背景代表尚未访问顶点
绿色背景代表已访问但其邻接点尚未处理的顶点
紫色背景代表正在处理其邻接点的顶点
黄色背景代表邻接点已经处理结束顶点

BFS(G,v):队列存放访问过但邻接点未处理的顶点



abecd

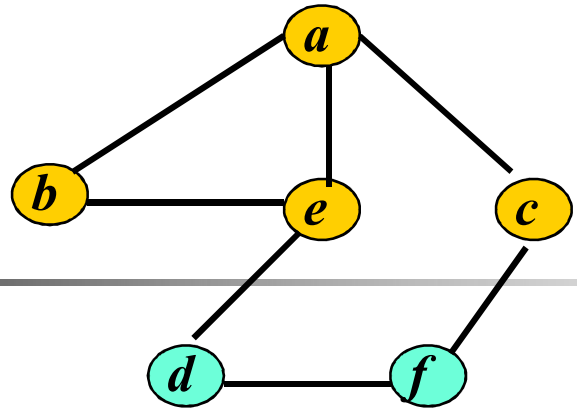
0	1	2	3	4	5	6	7	8	9
a	*b*	*e*	*c*	*d*					

f *r*

abecd

白色背景代表尚未访问顶点
绿色背景代表已访问但其邻接点尚未处理的顶点
紫色背景代表正在处理其邻接点的顶点
黄色背景代表邻接点已经处理结束顶点

BFS(G,v):队列存放访问过但邻接点未处理的顶点



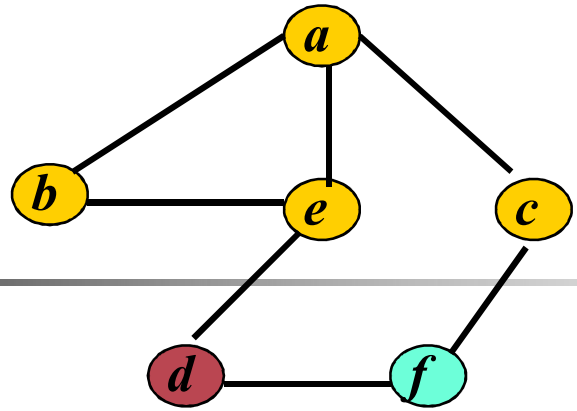
abecdf

0	1	2	3	4	5	6	7	8	9
a	*b*	*e*	*c*	*d*	*f*				

f *r*

abecdf

白色背景代表尚未访问顶点
绿色背景代表已访问但其邻接点尚未处理的顶点
紫色背景代表正在处理其邻接点的顶点
黄色背景代表邻接点已经处理结束顶点



abecdf

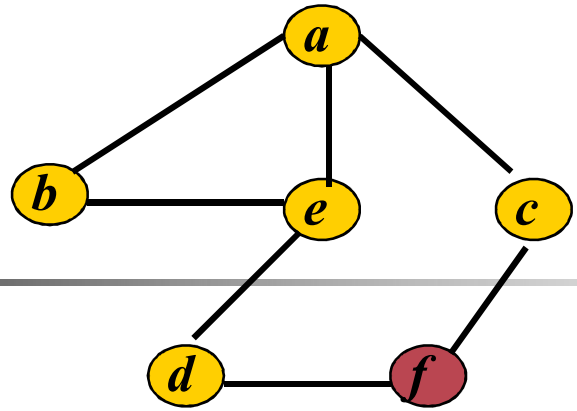
0	1	2	3	4	5	6	7	8	9
a	*b*	*e*	*c*	*d*	*f*				

f *r*

abecdf

白色背景代表尚未访问顶点
绿色背景代表已访问但其邻接点尚未处理的顶点
紫色背景代表正在处理其邻接点的顶点
黄色背景代表邻接点已经处理结束顶点

BFS(G,v):队列存放访问过但邻接点未处理的顶点



abecdf

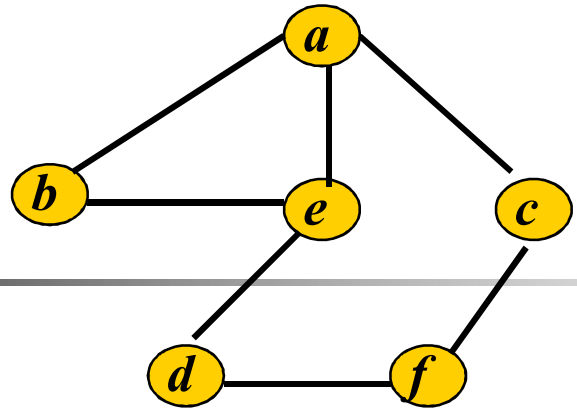
0	1	2	3	4	5	6	7	8	9
a	*b*	*e*	*c*	*d*	*f*				

f r

abecdf

白色背景代表尚未访问顶点
绿色背景代表已访问但其邻接点尚未处理的顶点
紫色背景代表正在处理其邻接点的顶点
黄色背景代表邻接点已经处理结束顶点

BFS(G,v):队列存放访问过但邻接点未处理的顶点



abecdf

0	1	2	3	4	5	6	7	8	9
a	*b*	*e*	*c*	*d*	*f*				

f r

abecdf

白色背景代表尚未访问顶点
绿色背景代表已访问但其邻接点尚未处理的顶点
紫色背景代表正在处理其邻接点的顶点
黄色背景代表邻接点已经处理结束顶点

```
void BFS(Graphs G,int v)
{ int q[MAXSIZE]; int f=r=0;
  visited[v] = 1; printf("%d",v);
  q[r++]=v;
  while(f!=r)
  {
      w=q[f++];
      for(p=G.arc[w].firstarc;p!=NULL;p=p->link)
    {  k=p->vex;
       if(!visit[k]){visit[k]=TRUE;  printf("%d",k);
                        if(r==MAXSIZE) exit(-2);
                        else q[r++]=k;}
    }
  }
} // BFSTraverse
```

将此算法改成循环队列，同时入队时判断是否溢出