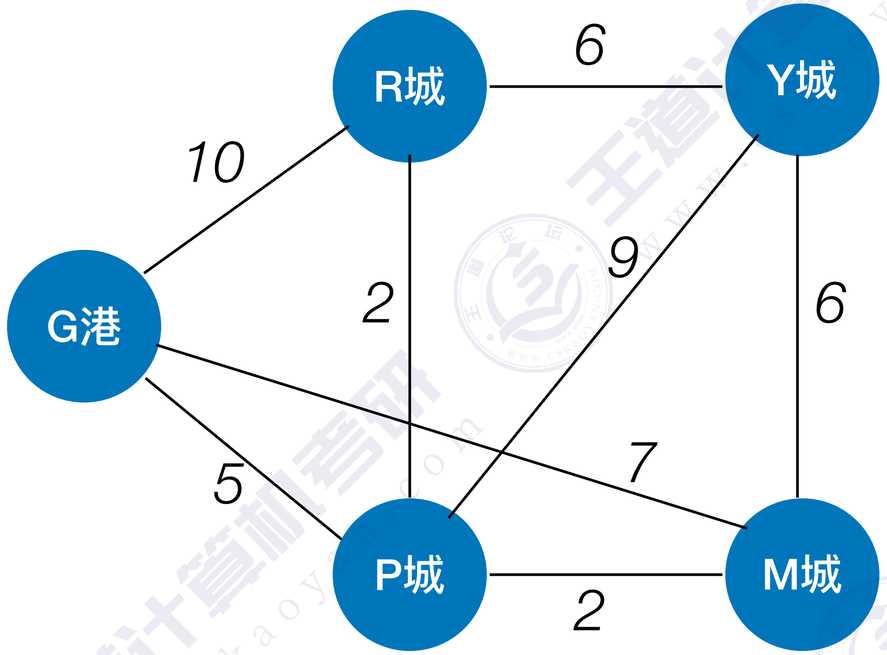
本节内容

最短路径

BFS算法

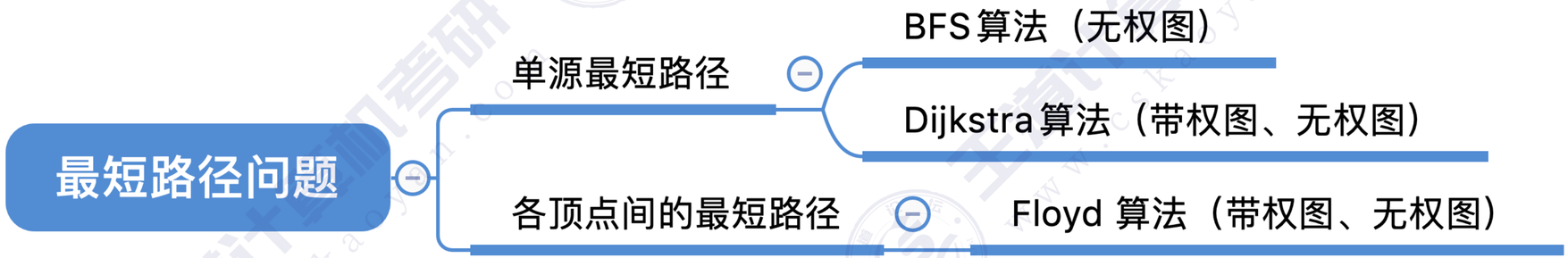
王道考研/CSKAOYAN.COM

最短路径问题



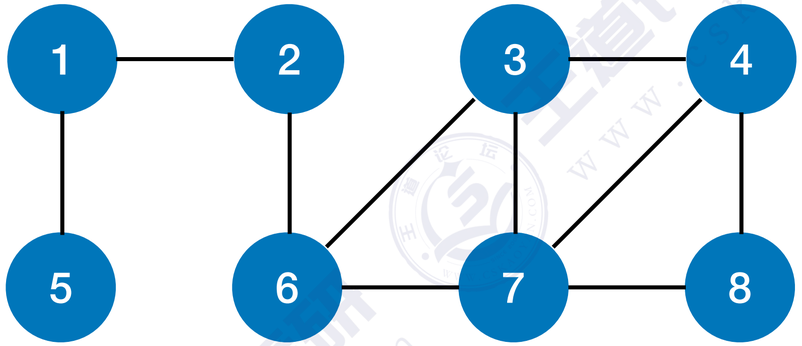
“G港”是个物流集散中心，经常需要往各个城市运东西，怎么运送距离最近？——单源最短路径问题

各个城市之间也需要互相往来，相互之间怎么走距离最近？——每对顶点间的最短路径



王道考研/CSKAOYAN.COM

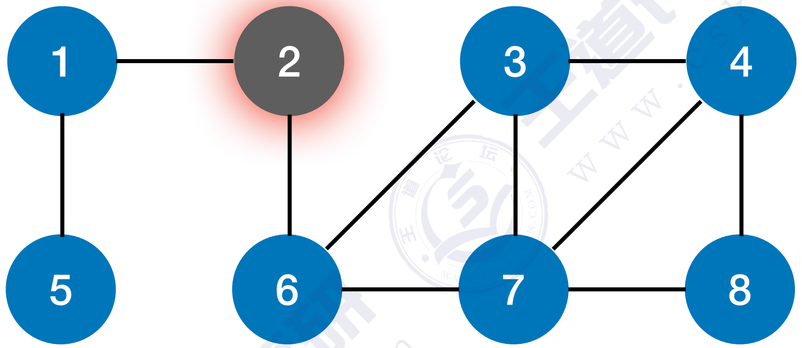
BFS求无权图的单源最短路径



注：无权图可以视为一种特殊的带权图，只是每条边的权值都为1

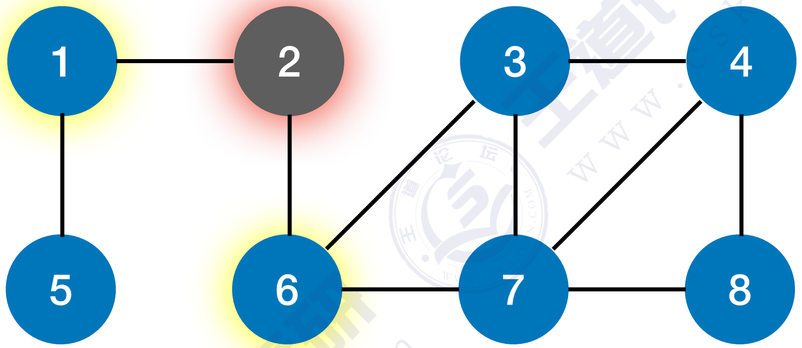
王道考研/CSKAOYAN.COM

BFS求无权图的单源最短路径



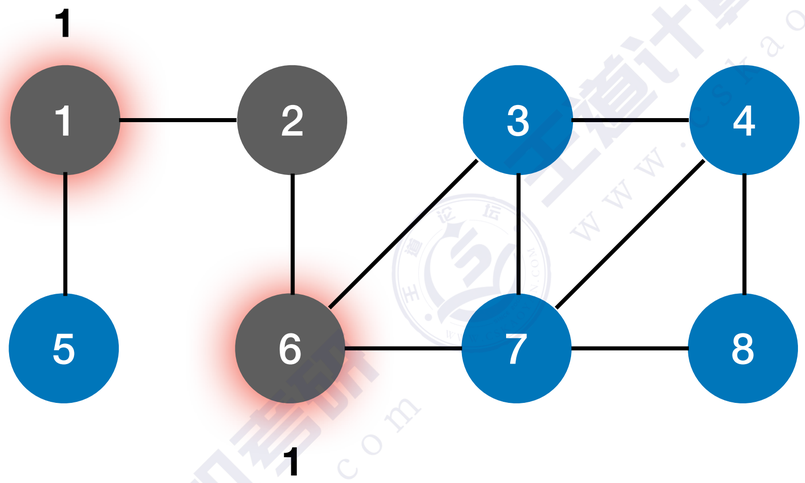
王道考研/CSKAOYAN.COM

BFS求无权图的单源最短路径



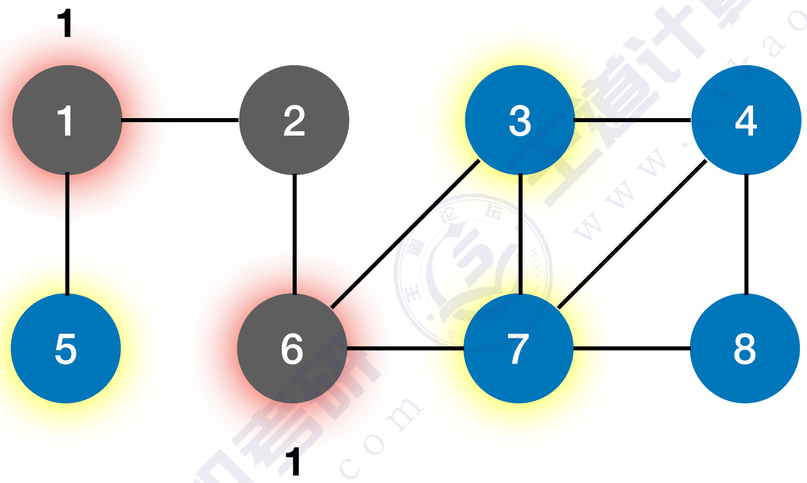
王道考研/CSKAOYAN.COM

BFS求无权图的单源最短路径



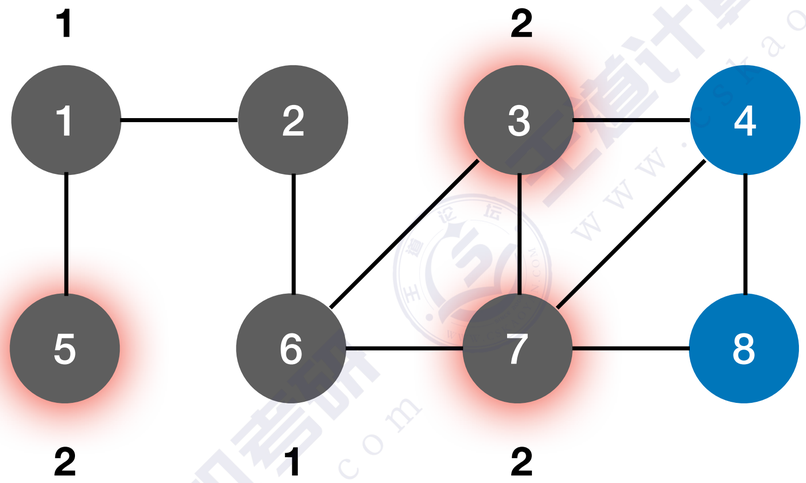
王道考研/CSKAOYAN.COM

BFS求无权图的单源最短路径



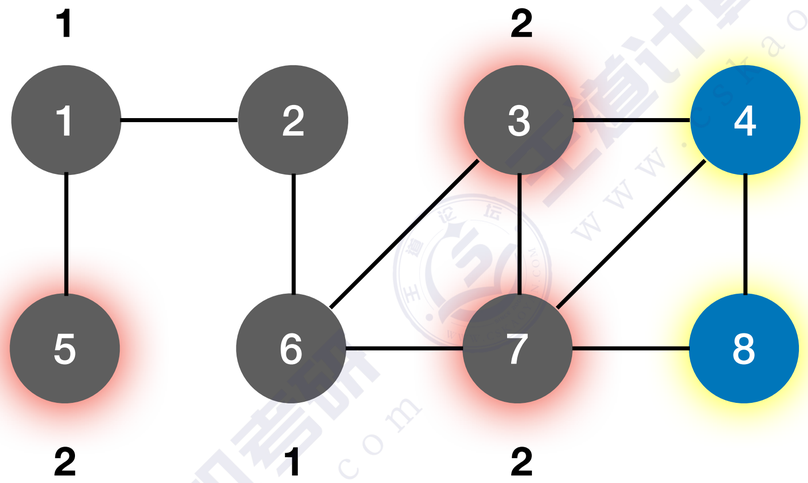
王道考研/CSKAOYAN.COM

BFS求无权图的单源最短路径



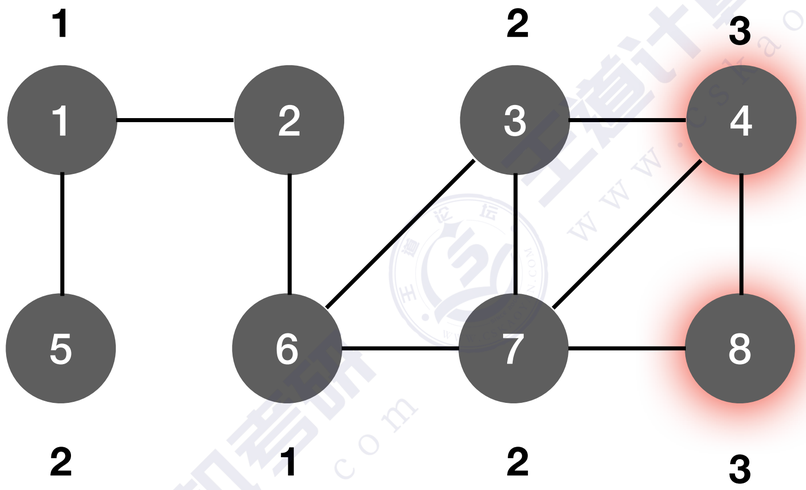
王道考研/CSKAOYAN.COM

BFS求无权图的单源最短路径



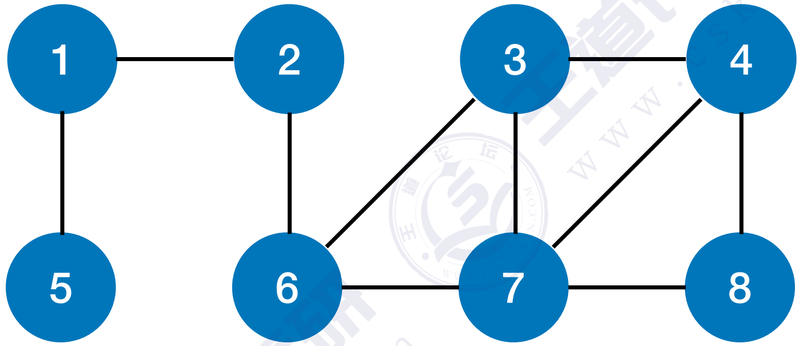
王道考研/CSKAOYAN.COM

BFS求无权图的单源最短路径



王道考研/CSKAOYAN.COM

代码实现



初始都为false

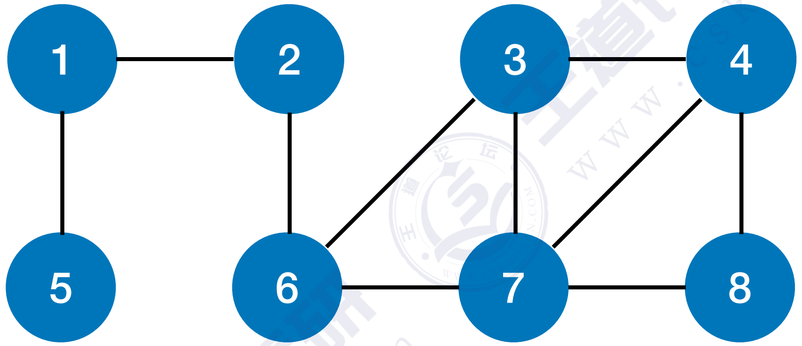
```
bool visited[MAX_VERTEX_NUM];    //访问标记数组

//广度优先遍历
void BFS(Graph G,int v){     //从顶点v出发，广度优先遍历图G
    visit(v);                    //访问初始顶点v
    visited[v]=TRUE;             //对v做已访问标记
    Enqueue(Q,v);                //顶点v入队列Q
    while(!isEmpty(Q)){
        DeQueue(Q,v);            //顶点v出队列
        for(w=FirstNeighbor(G,v);w>=0;w=NextNeighbor(G,v,w))
            //检测v所有邻接点
            if(!visited[w]){     //w为v的尚未访问的邻接顶点
                visit(w);        //访问顶点w
                visited[w]=TRUE;//对w做已访问标记
                EnQueue(Q,w);    //顶点w入队列
            }//if
    }//while
}
```

	1	2	3	4	5	6	7	8
visited	false	false	false	false	false	false	false	false

王道考研/CSKAOYAN.COM

代码实现



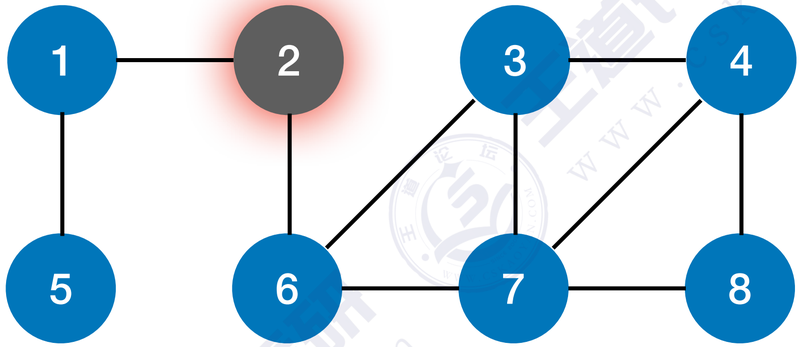

```
//求顶点 u 到其他顶点的最短路径
void BFS_MIN_Distance(Graph G,int u){
    //d[i]表示从u到i结点的最短路径
    for(i=0;i<G.vexnum;++i){
        d[i]=∞;       //初始化路径长度
        path[i]=-1; //最短路径从哪个顶点过来
    }
    d[u]=0;
    visited[u]=TRUE;
    EnQueue(Q,u);
    while(!isEmpty(Q)){                    //BFS算法主过程
        DeQueue(Q,u);                      //队头元素u出队
        for(w=FirstNeighbor(G,u);w>=0;w=NextNeighbor(G,u,w))
            if(!visited[w]){               //w为u的尚未访问的邻接顶点
                d[w]=d[u]+1;               //路径长度加1
                path[w]=u;                 //最短路径应从u到w
                visited[w]=TRUE;           //设已访问标记
                EnQueue(Q,w);              //顶点w入队
            }//if
    }//while
}
```

	1	2	3	4	5	6	7	8
d[ ]	∞	∞	∞	∞	∞	∞	∞	∞
path[ ]	-1	-1	-1	-1	-1	-1	-1	-1

	1	2	3	4	5	6	7	8
visited	false	false	false	false	false	false	false	false

王道考研/CSKAOYAN.COM

代码实现



2

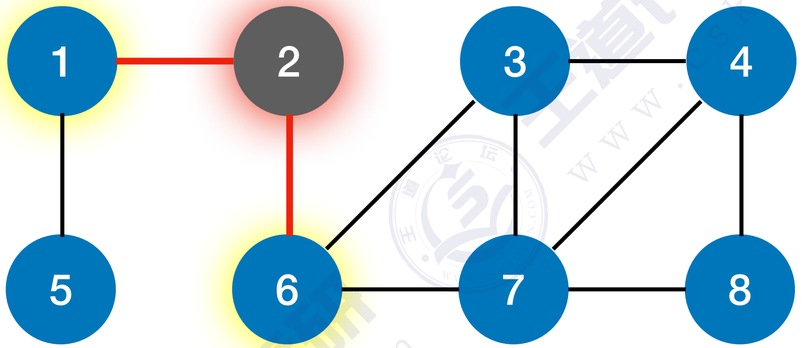
	1	2	3	4	5	6	7	8
d[ ]	∞	0	∞	∞	∞	∞	∞	∞
path[ ]	-1	-1	-1	-1	-1	-1	-1	-1

```
//求顶点 u 到其他顶点的最短路径
void BFS_MIN_Distance(Graph G,int u){
    //d[i]表示从u到i结点的最短路径
    for(i=0;i<G.vexnum;++i){
        d[i]=∞;       //初始化路径长度
        path[i]=-1; //最短路径从哪个顶点过来
    }
    d[u]=0;
    visited[u]=TRUE;
    EnQueue(Q,u);
    while(!isEmpty(Q)){                    //BFS算法主过程
        DeQueue(Q,u);                      //队头元素u出队
        for(w=FirstNeighbor(G,u);w>=0;w=NextNeighbor(G,u,w))
            if(!visited[w]){               //w为u的尚未访问的邻接顶点
                d[w]=d[u]+1;               //路径长度加1
                path[w]=u;                 //最短路径应从u到w
                visited[w]=TRUE;           //设已访问标记
                EnQueue(Q,w);              //顶点w入队
            }//if
    }//while
}
```

	1	2	3	4	5	6	7	8
visited	false	true	false	false	false	false	false	false

王道考研/CSKAOYAN.COM

代码实现



2

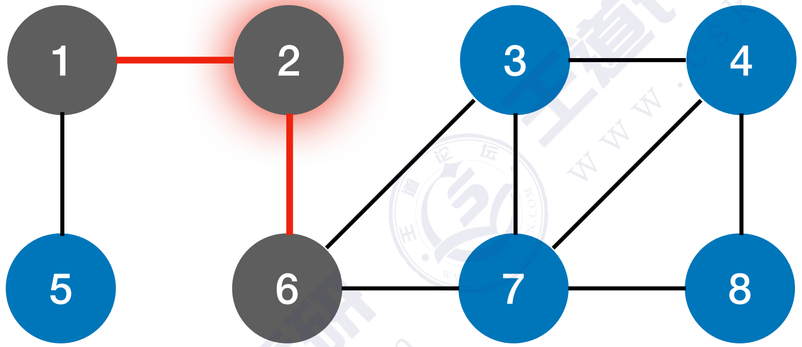
```
//求顶点 u 到其他顶点的最短路径
void BFS_MIN_Distance(Graph G,int u){
    //d[i]表示从u到i结点的最短路径
    for(i=0;i<G.vexnum;++i){
        d[i]=∞;       //初始化路径长度
        path[i]=-1; //最短路径从哪个顶点过来
    }
    d[u]=0;
    visited[u]=TRUE;
    EnQueue(Q,u);
    while(!isEmpty(Q)){                    //BFS算法主过程
        DeQueue(Q,u);                      //队头元素u出队
        for(w=FirstNeighbor(G,u);w>=0;w=NextNeighbor(G,u,w))
            if(!visited[w]){               //w为u的尚未访问的邻接顶点
                d[w]=d[u]+1;               //路径长度加1
                path[w]=u;                 //最短路径应从u到w
                visited[w]=TRUE;           //设已访问标记
                EnQueue(Q,w);              //顶点w入队
            }//if
    }//while
}
```

	1	2	3	4	5	6	7	8
d[ ]	∞	0	∞	∞	∞	∞	∞	∞
path[ ]	-1	-1	-1	-1	-1	-1	-1	-1

	1	2	3	4	5	6	7	8
visited	false	true	false	false	false	false	false	false

王道考研/CSKAOYAN.COM

代码实现



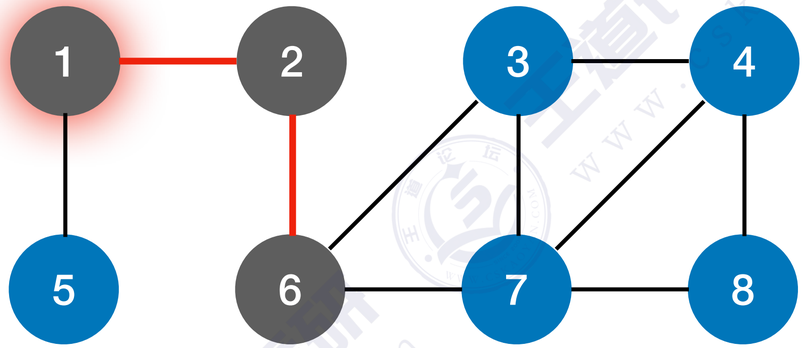
	1	2	3	4	5	6	7	8
d[ ]	1	0	∞	∞	∞	1	∞	∞
path[ ]	2	-1	-1	-1	-1	2	-1	-1

```
//求顶点 u 到其他顶点的最短路径
void BFS_MIN_Distance(Graph G,int u){
    //d[i]表示从u到i结点的最短路径
    for(i=0;i<G.vexnum;++i){
        d[i]=∞;       //初始化路径长度
        path[i]=-1; //最短路径从哪个顶点过来
    }
    d[u]=0;
    visited[u]=TRUE;
    EnQueue(Q,u);
    while(!isEmpty(Q)){                    //BFS算法主过程
        DeQueue(Q,u);                      //队头元素u出队
        for(w=FirstNeighbor(G,u);w>=0;w=NextNeighbor(G,u,w))
            if(!visited[w]){               //w为u的尚未访问的邻接顶点
                d[w]=d[u]+1;               //路径长度加1
                path[w]=u;                 //最短路径应从u到w
                visited[w]=TRUE;           //设已访问标记
                EnQueue(Q,w);              //顶点w入队
            }//if
    }//while
}
```

	1	2	3	4	5	6	7	8
visited	true	true	false	false	false	true	false	false

王道考研/CSKAOYAN.COM

代码实现



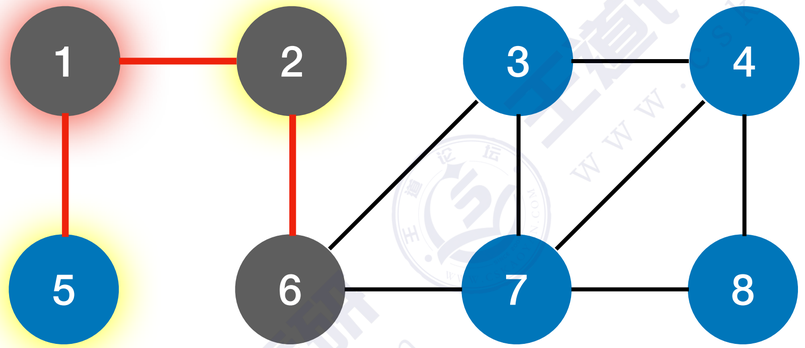
	1	2	3	4	5	6	7	8
d[ ]	1	0	∞	∞	∞	1	∞	∞
path[ ]	2	-1	-1	-1	-1	2	-1	-1

```
//求顶点 u 到其他顶点的最短路径
void BFS_MIN_Distance(Graph G,int u){
    //d[i]表示从u到i结点的最短路径
    for(i=0;i<G.vexnum;++i){
        d[i]=∞;       //初始化路径长度
        path[i]=-1; //最短路径从哪个顶点过来
    }
    d[u]=0;
    visited[u]=TRUE;
    EnQueue(Q,u);
    while(!isEmpty(Q)){                    //BFS算法主过程
        DeQueue(Q,u);                      //队头元素u出队
        for(w=FirstNeighbor(G,u);w>=0;w=NextNeighbor(G,u,w))
            if(!visited[w]){               //w为u的尚未访问的邻接顶点
                d[w]=d[u]+1;               //路径长度加1
                path[w]=u;                 //最短路径应从u到w
                visited[w]=TRUE;           //设已访问标记
                EnQueue(Q,w);              //顶点w入队
            }//if
    }//while
}
```

	1	2	3	4	5	6	7	8
visited	true	true	false	false	false	true	false	false

王道考研/CSKAOYAN.COM

代码实现



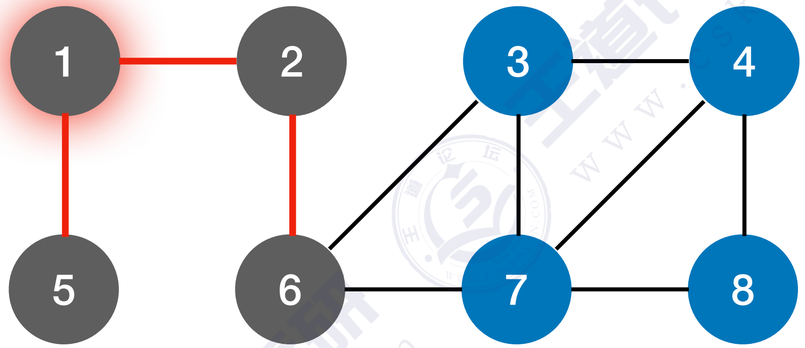
	1	2	3	4	5	6	7	8
d[ ]	1	0	∞	∞	∞	1	∞	∞
path[ ]	2	-1	-1	-1	-1	2	-1	-1

```
//求顶点 u 到其他顶点的最短路径
void BFS_MIN_Distance(Graph G,int u){
    //d[i]表示从u到i结点的最短路径
    for(i=0;i<G.vexnum;++i){
        d[i]=∞;       //初始化路径长度
        path[i]=-1; //最短路径从哪个顶点过来
    }
    d[u]=0;
    visited[u]=TRUE;
    EnQueue(Q,u);
    while(!isEmpty(Q)){                    //BFS算法主过程
        DeQueue(Q,u);                      //队头元素u出队
        for(w=FirstNeighbor(G,u);w>=0;w=NextNeighbor(G,u,w))
            if(!visited[w]){               //w为u的尚未访问的邻接顶点
                d[w]=d[u]+1;               //路径长度加1
                path[w]=u;                 //最短路径应从u到w
                visited[w]=TRUE;           //设已访问标记
                EnQueue(Q,w);              //顶点w入队
            }//if
    }//while
}
```

	1	2	3	4	5	6	7	8
visited	true	true	false	false	false	true	false	false

王道考研/CSKAOYAN.COM

代码实现



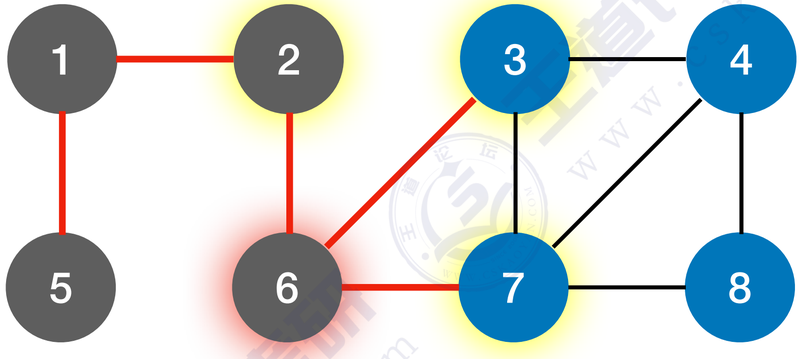
	1	2	3	4	5	6	7	8
d[ ]	1	0	∞	∞	2	1	∞	∞
path[ ]	2	-1	-1	-1	1	2	-1	-1

```
//求顶点 u 到其他顶点的最短路径
void BFS_MIN_Distance(Graph G,int u){
    //d[i]表示从u到i结点的最短路径
    for(i=0;i<G.vexnum;++i){
        d[i]=∞;       //初始化路径长度
        path[i]=-1; //最短路径从哪个顶点过来
    }
    d[u]=0;
    visited[u]=TRUE;
    EnQueue(Q,u);
    while(!isEmpty(Q)){                    //BFS算法主过程
        DeQueue(Q,u);                      //队头元素u出队
        for(w=FirstNeighbor(G,u);w>=0;w=NextNeighbor(G,u,w))
            if(!visited[w]){               //w为u的尚未访问的邻接顶点
                d[w]=d[u]+1;               //路径长度加1
                path[w]=u;                 //最短路径应从u到w
                visited[w]=TRUE;           //设已访问标记
                EnQueue(Q,w);              //顶点w入队
            }//if
    }//while
}
```

	1	2	3	4	5	6	7	8
visited	true	true	false	false	true	true	false	false

王道考研/CSKAOYAN.COM

代码实现



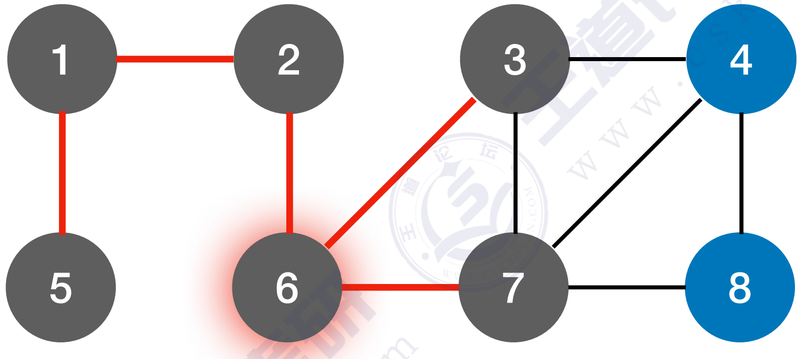
	1	2	3	4	5	6	7	8
d[ ]	1	0	∞	∞	2	1	∞	∞
path[ ]	2	-1	-1	-1	1	2	-1	-1

```
//求顶点 u 到其他顶点的最短路径
void BFS_MIN_Distance(Graph G,int u){
    //d[i]表示从u到i结点的最短路径
    for(i=0;i<G.vexnum;++i){
        d[i]=∞;       //初始化路径长度
        path[i]=-1; //最短路径从哪个顶点过来
    }
    d[u]=0;
    visited[u]=TRUE;
    EnQueue(Q,u);
    while(!isEmpty(Q)){                    //BFS算法主过程
        DeQueue(Q,u);                      //队头元素u出队
        for(w=FirstNeighbor(G,u);w>=0;w=NextNeighbor(G,u,w))
            if(!visited[w]){               //w为u的尚未访问的邻接顶点
                d[w]=d[u]+1;               //路径长度加1
                path[w]=u;                 //最短路径应从u到w
                visited[w]=TRUE;           //设已访问标记
                EnQueue(Q,w);              //顶点w入队
            }//if
    }//while
}
```

	1	2	3	4	5	6	7	8
visited	true	true	false	false	true	true	false	false

王道考研/CSKAOYAN.COM

代码实现



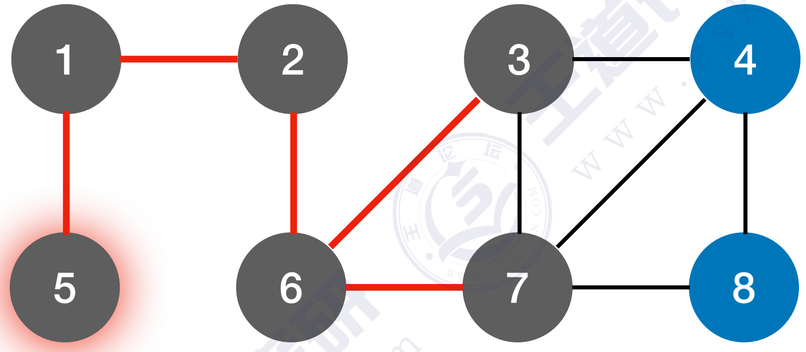
	1	2	3	4	5	6	7	8
d[ ]	1	0	2	∞	2	1	2	∞
path[ ]	2	-1	6	-1	1	2	6	-1

```
//求顶点 u 到其他顶点的最短路径
void BFS_MIN_Distance(Graph G,int u){
    //d[i]表示从u到i结点的最短路径
    for(i=0;i<G.vexnum;++i){
        d[i]=∞;       //初始化路径长度
        path[i]=-1; //最短路径从哪个顶点过来
    }
    d[u]=0;
    visited[u]=TRUE;
    EnQueue(Q,u);
    while(!isEmpty(Q)){                    //BFS算法主过程
        DeQueue(Q,u);                      //队头元素u出队
        for(w=FirstNeighbor(G,u);w>=0;w=NextNeighbor(G,u,w))
            if(!visited[w]){               //w为u的尚未访问的邻接顶点
                d[w]=d[u]+1;               //路径长度加1
                path[w]=u;                 //最短路径应从u到w
                visited[w]=TRUE;           //设已访问标记
                EnQueue(Q,w);              //顶点w入队
            }//if
    }//while
}
```

	1	2	3	4	5	6	7	8
visited	true	true	true	false	true	true	true	false

王道考研/CSKAOYAN.COM

代码实现



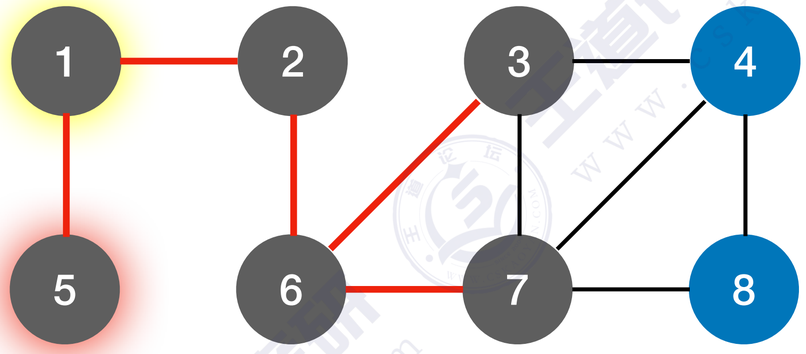
	1	2	3	4	5	6	7	8
d[ ]	1	0	2	∞	2	1	2	∞
path[ ]	2	-1	6	-1	1	2	6	-1

```
//求顶点 u 到其他顶点的最短路径
void BFS_MIN_Distance(Graph G,int u){
    //d[i]表示从u到i结点的最短路径
    for(i=0;i<G.vexnum;++i){
        d[i]=∞;       //初始化路径长度
        path[i]=-1; //最短路径从哪个顶点过来
    }
    d[u]=0;
    visited[u]=TRUE;
    EnQueue(Q,u);
    while(!isEmpty(Q)){                    //BFS算法主过程
        DeQueue(Q,u);                      //队头元素u出队
        for(w=FirstNeighbor(G,u);w>=0;w=NextNeighbor(G,u,w))
            if(!visited[w]){               //w为u的尚未访问的邻接顶点
                d[w]=d[u]+1;               //路径长度加1
                path[w]=u;                 //最短路径应从u到w
                visited[w]=TRUE;           //设已访问标记
                EnQueue(Q,w);              //顶点w入队
            }//if
    }//while
}
```

	1	2	3	4	5	6	7	8
visited	true	true	true	false	true	true	true	false

王道考研/CSKAOYAN.COM

代码实现



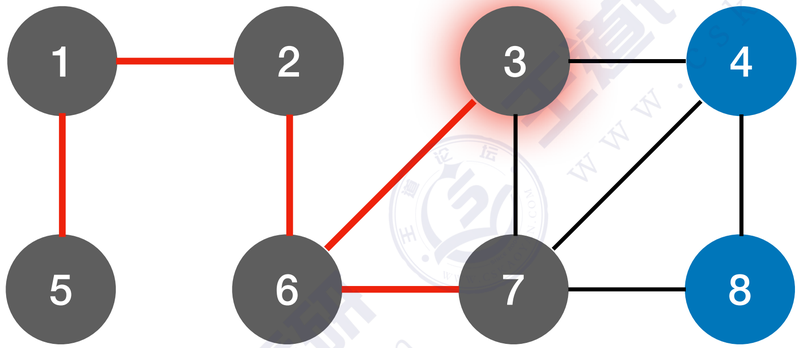
	1	2	3	4	5	6	7	8
d[ ]	1	0	2	∞	2	1	2	∞
path[ ]	2	-1	6	-1	1	2	6	-1

```
//求顶点 u 到其他顶点的最短路径
void BFS_MIN_Distance(Graph G,int u){
    //d[i]表示从u到i结点的最短路径
    for(i=0;i<G.vexnum;++i){
        d[i]=∞;       //初始化路径长度
        path[i]=-1; //最短路径从哪个顶点过来
    }
    d[u]=0;
    visited[u]=TRUE;
    EnQueue(Q,u);
    while(!isEmpty(Q)){                    //BFS算法主过程
        DeQueue(Q,u);                      //队头元素u出队
        for(w=FirstNeighbor(G,u);w>=0;w=NextNeighbor(G,u,w))
            if(!visited[w]){               //w为u的尚未访问的邻接顶点
                d[w]=d[u]+1;               //路径长度加1
                path[w]=u;                 //最短路径应从u到w
                visited[w]=TRUE;           //设已访问标记
                EnQueue(Q,w);              //顶点w入队
            }//if
    }//while
}
```

	1	2	3	4	5	6	7	8
visited	true	true	true	false	true	true	true	false

王道考研/CSKAOYAN.COM

代码实现



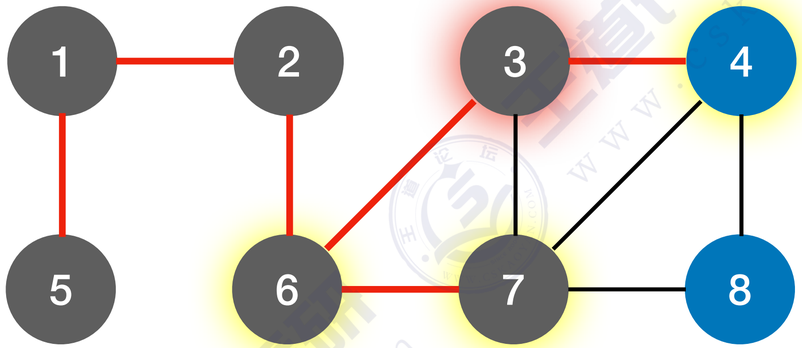
	1	2	3	4	5	6	7	8
d[ ]	1	0	2	∞	2	1	2	∞
path[ ]	2	-1	6	-1	1	2	6	-1

```
//求顶点 u 到其他顶点的最短路径
void BFS_MIN_Distance(Graph G,int u){
    //d[i]表示从u到i结点的最短路径
    for(i=0;i<G.vexnum;++i){
        d[i]=∞;       //初始化路径长度
        path[i]=-1; //最短路径从哪个顶点过来
    }
    d[u]=0;
    visited[u]=TRUE;
    EnQueue(Q,u);
    while(!isEmpty(Q)){                    //BFS算法主过程
        DeQueue(Q,u);                      //队头元素u出队
        for(w=FirstNeighbor(G,u);w>=0;w=NextNeighbor(G,u,w))
            if(!visited[w]){               //w为u的尚未访问的邻接顶点
                d[w]=d[u]+1;               //路径长度加1
                path[w]=u;                 //最短路径应从u到w
                visited[w]=TRUE;           //设已访问标记
                EnQueue(Q,w);              //顶点w入队
            }//if
    }//while
}
```

	1	2	3	4	5	6	7	8
visited	true	true	true	false	true	true	true	false

王道考研/CSKAOYAN.COM

代码实现



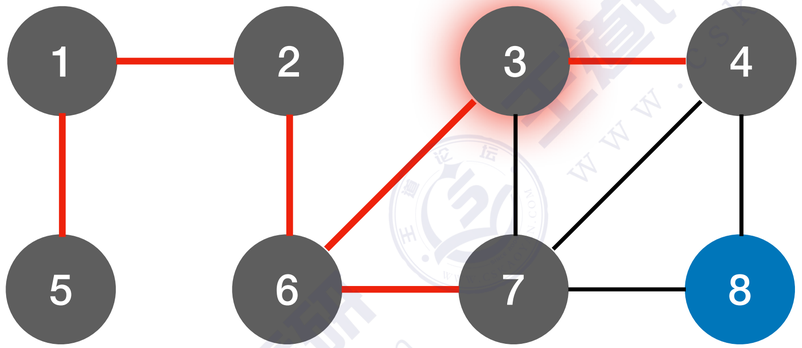
	1	2	3	4	5	6	7	8
d[ ]	1	0	2	∞	2	1	2	∞
path[ ]	2	-1	6	-1	1	2	6	-1

```
//求顶点 u 到其他顶点的最短路径
void BFS_MIN_Distance(Graph G,int u){
    //d[i]表示从u到i结点的最短路径
    for(i=0;i<G.vexnum;++i){
        d[i]=∞;       //初始化路径长度
        path[i]=-1; //最短路径从哪个顶点过来
    }
    d[u]=0;
    visited[u]=TRUE;
    EnQueue(Q,u);
    while(!isEmpty(Q)){                    //BFS算法主过程
        DeQueue(Q,u);                      //队头元素u出队
        for(w=FirstNeighbor(G,u);w>=0;w=NextNeighbor(G,u,w))
            if(!visited[w]){               //w为u的尚未访问的邻接顶点
                d[w]=d[u]+1;               //路径长度加1
                path[w]=u;                 //最短路径应从u到w
                visited[w]=TRUE;           //设已访问标记
                EnQueue(Q,w);              //顶点w入队
            }//if
    }//while
}
```

	1	2	3	4	5	6	7	8
visited	true	true	true	false	true	true	true	false

王道考研/CSKAOYAN.COM

代码实现



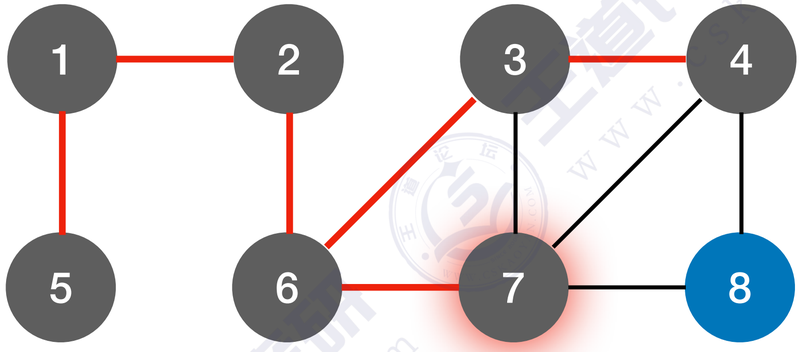

```
//求顶点 u 到其他顶点的最短路径
void BFS_MIN_Distance(Graph G,int u){
    //d[i]表示从u到i结点的最短路径
    for(i=0;i<G.vexnum;++i){
        d[i]=∞;       //初始化路径长度
        path[i]=-1; //最短路径从哪个顶点过来
    }
    d[u]=0;
    visited[u]=TRUE;
    EnQueue(Q,u);
    while(!isEmpty(Q)){                    //BFS算法主过程
        DeQueue(Q,u);                      //队头元素u出队
        for(w=FirstNeighbor(G,u);w>=0;w=NextNeighbor(G,u,w))
            if(!visited[w]){               //w为u的尚未访问的邻接顶点
                d[w]=d[u]+1;               //路径长度加1
                path[w]=u;                 //最短路径应从u到w
                visited[w]=TRUE;           //设已访问标记
                EnQueue(Q,w);              //顶点w入队
            }//if
    }//while
}
```

	1	2	3	4	5	6	7	8
d[ ]	1	0	2	3	2	1	2	∞
path[ ]	2	-1	6	3	1	2	6	-1

	1	2	3	4	5	6	7	8
visited	true	true	true	true	true	true	true	false

王道考研/CSKAOYAN.COM

代码实现



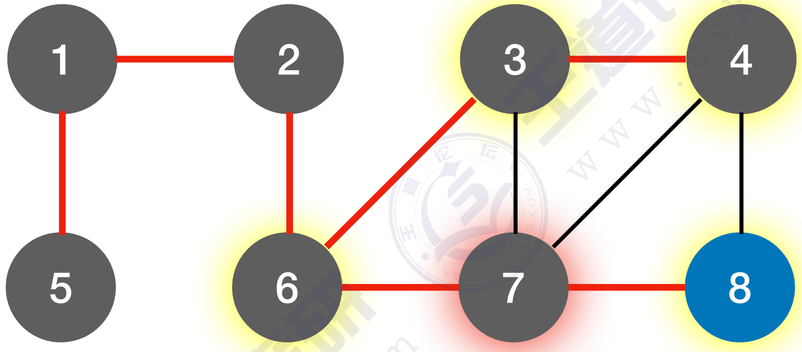

```
//求顶点 u 到其他顶点的最短路径
void BFS_MIN_Distance(Graph G,int u){
    //d[i]表示从u到i结点的最短路径
    for(i=0;i<G.vexnum;++i){
        d[i]=∞;       //初始化路径长度
        path[i]=-1; //最短路径从哪个顶点过来
    }
    d[u]=0;
    visited[u]=TRUE;
    EnQueue(Q,u);
    while(!isEmpty(Q)){                    //BFS算法主过程
        DeQueue(Q,u);                      //队头元素u出队
        for(w=FirstNeighbor(G,u);w>=0;w=NextNeighbor(G,u,w))
            if(!visited[w]){               //w为u的尚未访问的邻接顶点
                d[w]=d[u]+1;               //路径长度加1
                path[w]=u;                 //最短路径应从u到w
                visited[w]=TRUE;           //设已访问标记
                EnQueue(Q,w);              //顶点w入队
            }//if
    }//while
}
```

	1	2	3	4	5	6	7	8
d[ ]	1	0	2	3	2	1	2	∞
path[ ]	2	-1	6	3	1	2	6	-1

	1	2	3	4	5	6	7	8
visited	true	true	true	true	true	true	true	false

王道考研/CSKAOYAN.COM

代码实现



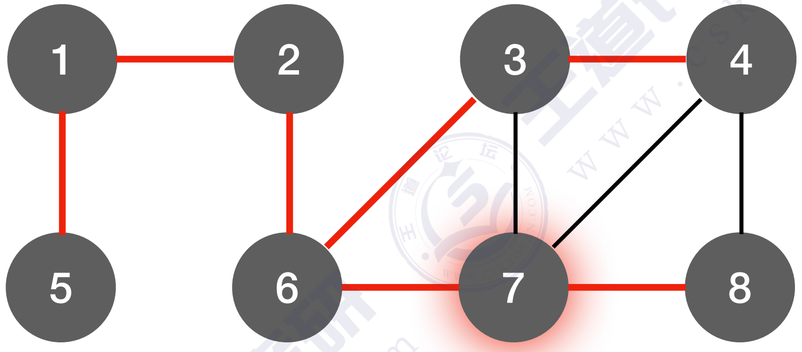

```
//求顶点 u 到其他顶点的最短路径
void BFS_MIN_Distance(Graph G,int u){
    //d[i]表示从u到i结点的最短路径
    for(i=0;i<G.vexnum;++i){
        d[i]=∞;       //初始化路径长度
        path[i]=-1; //最短路径从哪个顶点过来
    }
    d[u]=0;
    visited[u]=TRUE;
    EnQueue(Q,u);
    while(!isEmpty(Q)){                    //BFS算法主过程
        DeQueue(Q,u);                      //队头元素u出队
        for(w=FirstNeighbor(G,u);w>=0;w=NextNeighbor(G,u,w))
            if(!visited[w]){               //w为u的尚未访问的邻接顶点
                d[w]=d[u]+1;               //路径长度加1
                path[w]=u;                 //最短路径应从u到w
                visited[w]=TRUE;           //设已访问标记
                EnQueue(Q,w);              //顶点w入队
            }//if
    }//while
}
```

	1	2	3	4	5	6	7	8
d[ ]	1	0	2	3	2	1	2	∞
path[ ]	2	-1	6	3	1	2	6	-1

	1	2	3	4	5	6	7	8
visited	true	true	true	true	true	true	true	false

王道考研/CSKAOYAN.COM

代码实现



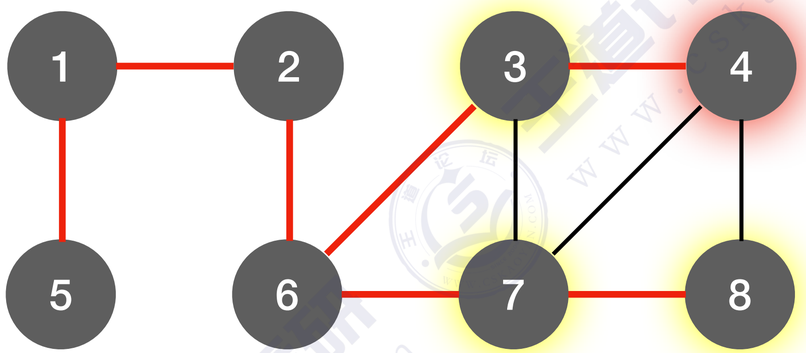

```
//求顶点 u 到其他顶点的最短路径
void BFS_MIN_Distance(Graph G,int u){
    //d[i]表示从u到i结点的最短路径
    for(i=0;i<G.vexnum;++i){
        d[i]=∞;       //初始化路径长度
        path[i]=-1; //最短路径从哪个顶点过来
    }
    d[u]=0;
    visited[u]=TRUE;
    EnQueue(Q,u);
    while(!isEmpty(Q)){                    //BFS算法主过程
        DeQueue(Q,u);                      //队头元素u出队
        for(w=FirstNeighbor(G,u);w>=0;w=NextNeighbor(G,u,w))
            if(!visited[w]){               //w为u的尚未访问的邻接顶点
                d[w]=d[u]+1;               //路径长度加1
                path[w]=u;                 //最短路径应从u到w
                visited[w]=TRUE;           //设已访问标记
                EnQueue(Q,w);              //顶点w入队
            }//if
    }//while
}
```

	1	2	3	4	5	6	7	8
d[ ]	1	0	2	3	2	1	2	3
path[ ]	2	-1	6	3	1	2	6	7

	1	2	3	4	5	6	7	8
visited	true	true	true	true	true	true	true	true

王道考研/CSKAOYAN.COM

代码实现



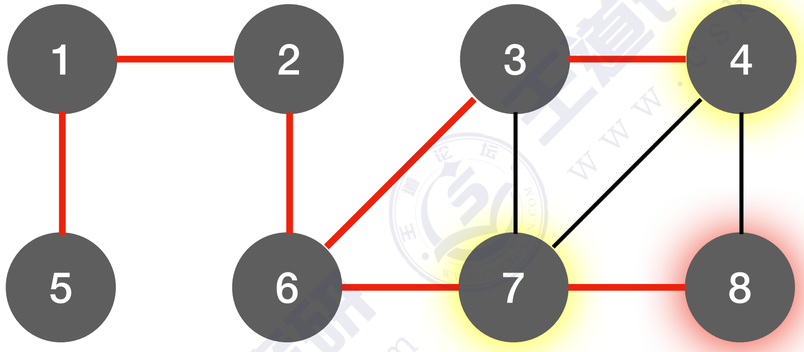

```
//求顶点 u 到其他顶点的最短路径
void BFS_MIN_Distance(Graph G,int u){
    //d[i]表示从u到i结点的最短路径
    for(i=0;i<G.vexnum;++i){
        d[i]=∞;       //初始化路径长度
        path[i]=-1; //最短路径从哪个顶点过来
    }
    d[u]=0;
    visited[u]=TRUE;
    EnQueue(Q,u);
    while(!isEmpty(Q)){                    //BFS算法主过程
        DeQueue(Q,u);                      //队头元素u出队
        for(w=FirstNeighbor(G,u);w>=0;w=NextNeighbor(G,u,w))
            if(!visited[w]){               //w为u的尚未访问的邻接顶点
                d[w]=d[u]+1;               //路径长度加1
                path[w]=u;                 //最短路径应从u到w
                visited[w]=TRUE;           //设已访问标记
                EnQueue(Q,w);              //顶点w入队
            }//if
    }//while
}
```

	1	2	3	4	5	6	7	8
d[ ]	1	0	2	3	2	1	2	3
path[ ]	2	-1	6	3	1	2	6	7

	1	2	3	4	5	6	7	8
visited	true	true	true	true	true	true	true	true

王道考研/CSKAOYAN.COM

代码实现



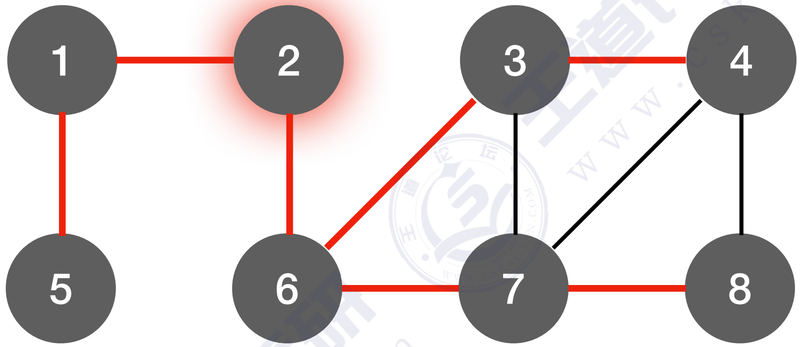

```
//求顶点 u 到其他顶点的最短路径
void BFS_MIN_Distance(Graph G,int u){
    //d[i]表示从u到i结点的最短路径
    for(i=0;i<G.vexnum;++i){
        d[i]=∞;       //初始化路径长度
        path[i]=-1; //最短路径从哪个顶点过来
    }
    d[u]=0;
    visited[u]=TRUE;
    EnQueue(Q,u);
    while(!isEmpty(Q)){                    //BFS算法主过程
        DeQueue(Q,u);                      //队头元素u出队
        for(w=FirstNeighbor(G,u);w>=0;w=NextNeighbor(G,u,w))
            if(!visited[w]){               //w为u的尚未访问的邻接顶点
                d[w]=d[u]+1;               //路径长度加1
                path[w]=u;                 //最短路径应从u到w
                visited[w]=TRUE;           //设已访问标记
                EnQueue(Q,w);              //顶点w入队
            }//if
    }//while
}
```

	1	2	3	4	5	6	7	8
d[ ]	1	0	2	3	2	1	2	3
path[ ]	2	-1	6	3	1	2	6	7

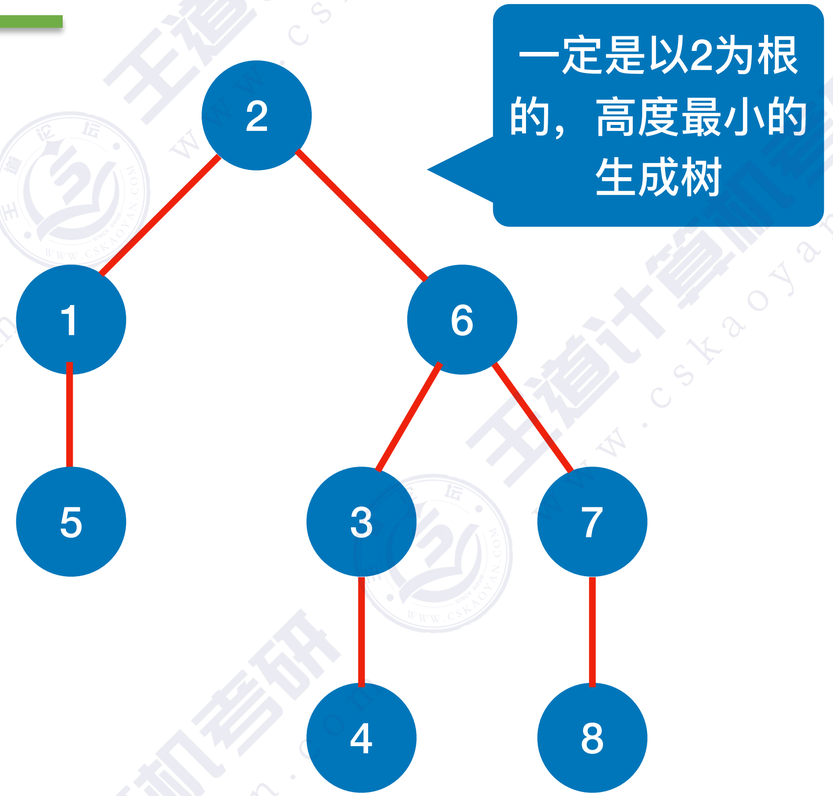
	1	2	3	4	5	6	7	8
visited	true	true	true	true	true	true	true	true

王道考研/CSKAOYAN.COM

知识点回顾与重要考点



广度优先生成树



一定是以2为根的，高度最小的生成树

就是对BFS的小修改，在visit一个顶点时，修改其最短路径长度 d[ ] 并在 path[ ] 记录前驱结点

	1	2	3	4	5	6	7	8
d[ ]	1	0	2	3	2	1	2	3
path[ ]	2	-1	6	3	1	2	6	7

2到8的最短路径长度 = d[8] = 3

通过path数组可知，2到8的最短路径为：8 <— 7 <— 6 <— 2

王道考研/CSKAOYAN.COM