第7章 图

1．选择题

（1）在一个图中，所有顶点的度数之和等于图的边数的（ ）倍。（画个无向图）

A．1/2 B．1 C．2 D．4

答案：C

（2）在一个有向图中，所有顶点的入度之和等于所有顶点的出度之和的（ ）倍。

A．1/2 B．1 C．2 D．4

答案：B

解释：有向图所有顶点入度之和等于所有顶点出度之和。

（3）具有n个顶点的有向图最多有（ ）条边。

A．n B．n(n-1) C．n(n+1) D．n2

答案：B

解释：有向图的边有方向之分，即为从n个顶点中选取2个顶点有序排列，结果为n(n-1)。

（4）n个顶点的连通图用邻接距阵表示时，该距阵至少有（ ）个非零元素。

A．n B．2(n-1) C．n/2 D．n2

答案：B

（5）G是一个非连通无向图，共有28条边，则该图至少有（ ）个顶点。

A．7 B．8 C．9 D．10

答案：C

解释：8个顶点的无向图最多有8\*7/2=28条边，再添加一个点即构成非连通无向图，故至少有9个顶点。

（6）若从无向图的任意一个顶点出发进行一次深度优先搜索可以访问图中所有的顶点，则该图一定是（ ）图。

A．非连通 B．连通 C．强连通 D．有向

答案：B

解释：即从该无向图任意一个顶点出发有到各个顶点的路径，所以该无向图是连通图。

（7）下面（　）算法适合构造一个稠密图G的最小生成树。

A． Prim算法 B．Kruskal算法 C．Floyd算法 D．Dijkstra算法

答案：A

解释：Prim算法适合构造一个稠密图G的最小生成树，Kruskal算法适合构造一个稀疏图G的最小生成树。

（8）用邻接表表示图进行广度优先遍历时，通常借助（ ）来实现算法。

A．栈 B. 队列 C. 树 D．图

答案：B

解释：广度优先遍历通常借助队列来实现算法，深度优先遍历通常借助栈来实现算法。

（9）用邻接表表示图进行深度优先遍历时，通常借助（ ）来实现算法。

A．栈 B. 队列 C. 树 D．图

答案：A

解释：广度优先遍历通常借助队列来实现算法，深度优先遍历通常借助栈来实现算法。

（10）深度优先遍历类似于二叉树的（ ）。

A．先序遍历 B．中序遍历 C．后序遍历 D．层次遍历

答案：A

（11）广度优先遍历类似于二叉树的（ ）。

A．先序遍历 B．中序遍历 C．后序遍历 D．层次遍历

答案：D

（12）图的BFS生成树的树高比DFS生成树的树高（ ）。

A．小 B．相等 C．小或相等 D．大或相等

答案：C

解释：对于一些特殊的图，比如只有一个顶点的图，其BFS生成树的树高和DFS生成树的树高相等。一般的图，根据图的BFS生成树和DFS树的算法思想，BFS生成树的树高比DFS生成树的树高小。

（13）已知图的邻接矩阵如图6.30所示，则从顶点*v*0出发按深度优先遍历的结果是（ ）。

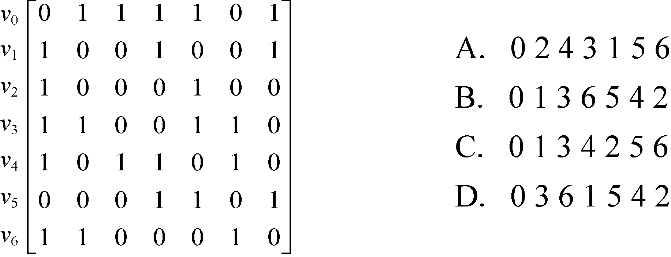


  图6.30 邻接矩阵

答案：C

（14）已知图的邻接表如图6.31所示，则从顶点*v*0出发按广度优先遍历的结果是（ ），按深度优先遍历的结果是（ ）。

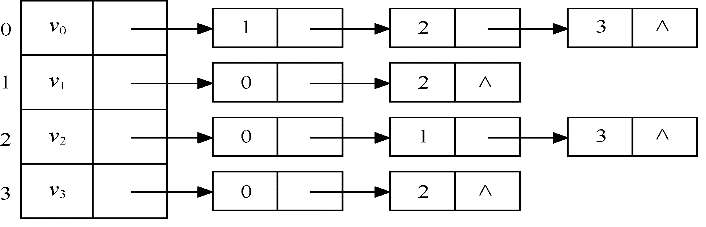


图6.31 邻接表

A．0 1 3 2 B．0 2 3 1 C．0 3 2 1 D．0 1 2 3

答案：D、D

（15）下面（ ）方法可以判断出一个有向图是否有环。

A．深度优先遍历 B．拓扑排序 C．求最短路径 D．求关键路径

答案：B

**2．应用题**

（1）已知图6.32所示的有向图，请给出：

① 每个顶点的入度和出度；

② 邻接矩阵；

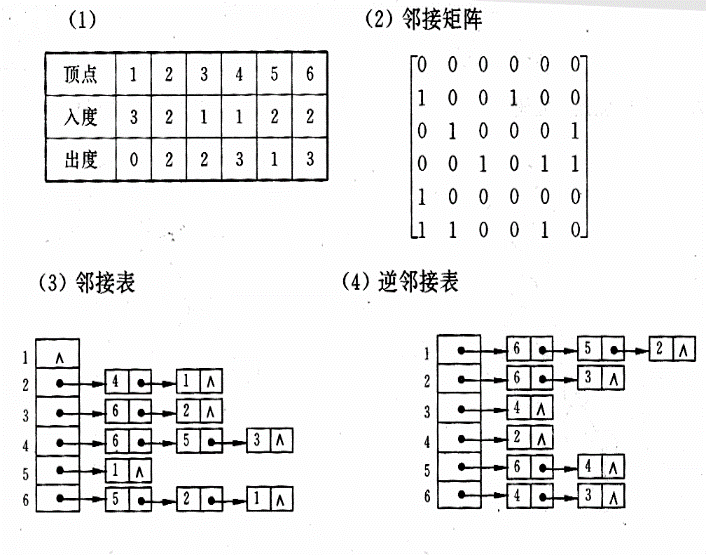
③ 邻接表；

④ 逆邻接表。

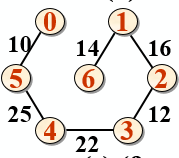


  图6.32 有向图 图6.33 无向网

答案：



2. 最小生成树为：



（3）已知图的邻接矩阵如图6.34所示。试分别画出自顶点1出发进行遍历所得的深度优先生成树和广度优先生成树。

**深度优先遍历序列为：**

**1-7-3-4-5-6-2-10-9-8**

**广度优先遍历序列为**

**1-7-9-3-10-5-4-8-6-2**

（4）有向网如图6.35所示，试用迪杰斯特拉算法求出从顶点a到其他各顶点间的最短路径，完成表6.9。

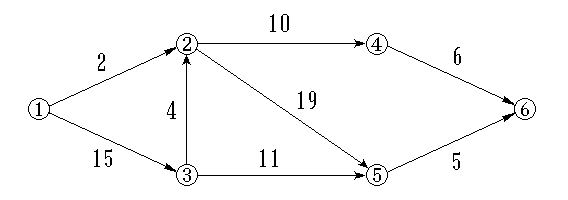
  图6.34 邻接矩阵 图6.35 有向网

表6.9

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| D  终点 | i=1 | i=2 | i=3 | i=4 | i=5 | i=6 |
| b | 15  (a,b) | 15  (a,b) | 15  (a,b) | 15  (a,b) | 15  (a,b) | **15**  **(a,b)** |
| c | **2**  **(a,c)** |  |  |  |  |  |
| d | 12  (a,d) | 12  (a,d) | 11  (a,c,f,d) | **11**  **(a,c,f,d)** |  |  |
| e | ∞ | 10  (a,c,e) | **10**  **(a,c,e)** |  |  |  |
| f | ∞ | **6**  **(a,c,f)** |  |  |  |  |
| g | ∞ | ∞ | 16  (a,c,f,g) | 16  (a,c,f,g) | **14**  **(a,c,f,d,g)** |  |
| S  终点集 | {a,c} | {a,c,f} | {a,c,f,e} | {a,c,f,e,d} | {a,c,f,e,d,g} | {a,c,f,e,d,g,b} |

（5）试对图6.36所示的AOE-网：

① 求这个工程最早可能在什么时间结束；

② 求每个活动的最早开始时间和最迟开始时间；

③ 确定哪些活动是关键活动

图6.36 AOE-网

答案：按拓扑有序的顺序计算各个顶点的最早可能开始时间*Ve*和最迟允许开始时间*Vl*。然后再计算各个活动的最早可能开始时间e和最迟允许开始时间*l*，根据*l - e* = 0? 来确定关键活动，从而确定关键路径。

1. 此工程最早完成时间为43。

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| *顶点* | 1 | 2 | 3 | 4 | 5 | 6 |
| *Ve* | 0 | 19 | 15 | 29 | 38 | 43 |
| *Vl* | 0 | 19 | 15 | 37 | 38 | 43 |

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 活动 | <1,2> | <1, 3> | <3, 2> | <2, 4> | <2, 5> | <3, 5> | <4, 6> | <5, 6> |
| *e* | 0 | 0 | 15 | 19 | 19 | 15 | 29 | 38 |
| *l* | 17 | 0 | 15 | 27 | 19 | 27 | 37 | 38 |
| *l-e* | 17 | 0 | 0 | 8 | 0 | 12 | 8 | 0 |

关键路径为<1, 3> <3, 2> <2, 5> <5, 6>

四 、编程题

int degree2(Graph ga, int numb)

//根据有向图的邻接矩阵求出序号为numb的顶点的度数

{ int i,j,d=0;

//求出顶点numb的出度

for(j=0; j<ga.vexnum; j++)

if(ga.arcs[numb][j]!=0 )

d++;

//求出顶点numb的入度

for(i=0; i<ga.vexnum; i++)

if(ga.arcs[i][numb]!=0 )

d++;

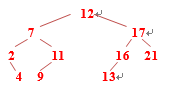
//返回顶点numb的度

return (d);

}

查找排序作业本习题

1. 在一棵空的二叉排序树中依次输入的关键字序列为12，7，17，11，16，2，13，9，21，4. 请画出所得到的二叉排序树，请计算在等概率情况下，查找成功的平均查找长度ASL；



**ASL =** =1/10\*( C1+C2+C3+C4+C5+c6+c7+c7+c8+c9+c10)

=1/10\*(1+2+2+3+3+3+3+4+4+4)=29/10=2.9

查找12 ，比较1次 c1=1

查找7，17 ，比较2次， c2=2,c3=2

查找2,11,16,21, 比较3次 ，C4=C5=C6=C7=3 ,

查找4，9，13 ，比较4次 ，c8=c9=c10=4

1. 设待排序的关键字序列为{12，2，16，30，28，10，16\*，20，6，18}，写出使用希尔排序方法每趟排序结束后关键字序列的状态。（增量5，3，1）

D=5 10 2 16 6 18 12 16\* 20 30 28

D=3 6 2 12 10 18 16 16\* 20 30 28

D=1 2 6 10 12 16 16\* 18 20 28 30

1. 直接插入排序程序

main( )

{ int a[10]={10,5,25,15,50,30,40,20,80,70};

int i,j,x;

for(i=1;i<10;++i)

{ if (a[i]<a[i-1] )

{

x=a[i];

for(j=i-1; j>=0&& x<a[j] ;--j)

a[j+1]=a[j]; ; // 记录后移

a[j+1] =x; // 插入到正确位置

}

}

printf("\n\n直接插入排序后的结果为：:\n");

for(i=0;i<10;i++)

printf("%5d",a[i]);

printf("\n");

}