第 5 章 图

5.1 本 章 导 学

5.1.1 知识结构图

本章有两条主线：一条主线是图的逻辑结构和存储结构，重点是图的两种遍历（深度优先和广度优先）的执行过程以及以遍历为核心的其他操作，例如求生成树、连通分量等，标☆的知识点为扩展与提高内容；另一条主线是图的经典应用，这些经典应用是本章的重点和难点，首先把握算法的基本思想，其次描述算法的执行过程和顶层伪代码，再次分析算法采用的存储结构，最后才能复现具体的算法。本章的知识结构如图 5-1 所示。

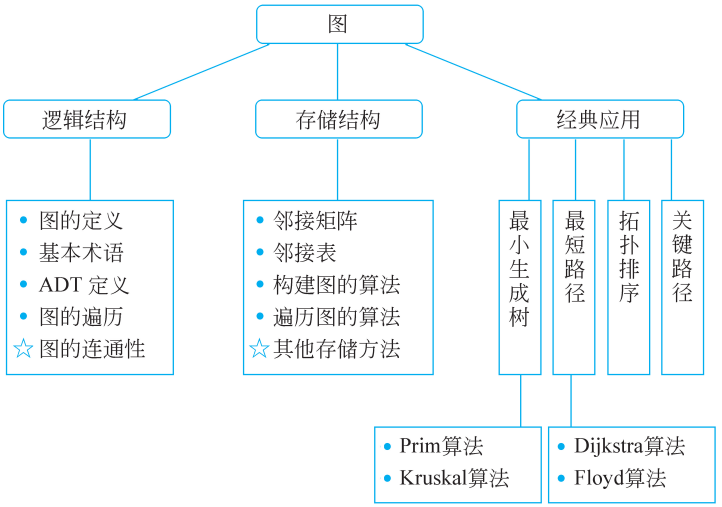


图 5-1 第 5 章的知识结构

5.1.2 重点整理

1. 图由顶点集合和顶点之间边的集合组成。如果任意两个顶点之间的边都是无向边，则称该图为无向图；否则称该图为有向图。边上带权的图称为网图。

2. 在无向图中，对于任意顶点 v_i 和 v_j，若存在边(v_i, v_j)，则称顶点 v_i 和 v_j 互为邻接点。在有向图中，对于任意顶点 v_i 和 v_j，若存在弧$<v_i, v_j>$，则称顶点 v_j 是 v_i 的邻接点，称顶点 v_i 是 v_j 的逆邻接点。

3. 含有 n 个顶点的无向完全图共有 $n(n-1)/2$ 条边。含有 n 个顶点的有向完全图共有 $n(n-1)$条边。

4. 在无向图中，顶点 v 的度是依附于该顶点的边的个数。在有向图中，顶点 v 的入度是以该顶点为弧头的弧的个数，顶点 v 的出度是以该顶点为弧尾的弧的个数。

5. 在无向图 $G=(V,E)$ 中，顶点 v_p 到 v_q 之间的路径是一个顶点序列 $v_p=v_{i0},v_{i1},\cdots,v_{im}=v_q$，其中，$(v_{ij-1},v_{ij})\in E(1\leqslant j\leqslant m)$。如果 G 是有向图，则 $<v_{ij-1},v_{ij}>\in E(1\leqslant j\leqslant m)$。路径上边的数目称为路径长度，第一个顶点和最后一个顶点相同的路径称为回路或环。

6. 在无向图中，若任意顶点 v_i 和 v_j 之间均有路径，则称该图是连通图。非连通图的极大连通子图称为连通分量。在有向图中，对任意顶点 v_i 和 v_j，若从顶点 v_i 到 v_j 和从顶点 v_j 到 v_i 均有路径，则称该有向图是强连通图。非强连通图的极大强连通子图称为强连通分量。

7. 图的遍历次序通常有深度优先和广度优先两种方式。深度优先遍历以递归方式进行，采用栈保存已访问的顶点；广度优先遍历以层次方式进行，采用队列保存已访问的顶点。

8. 图的基本存储结构有邻接矩阵、邻接表等。图的邻接矩阵用一个一维数组存储顶点的信息(顶点表)，用一个二维数组存储边的信息(邻接矩阵)。图的邻接表由边表和顶点表组成，每个顶点的所有邻接点构成一个边表，所有边表的头指针和存储顶点信息的一维数组构成顶点表。

9. 在图的存储结构中，用一维数组存储顶点信息，同时确定了顶点在数组中的下标，即顶点的编号。图的其他存储结构还有边集数组、邻接多重表、十字链表等。

10. 连通图的生成树是包含图中全部顶点的一个极小连通子图。连通图的生成树可以在遍历过程中得到。

11. 最小生成树是无向连通网代价最小的生成树，Prim 算法和 Kruskal 算法是构造最小生成树的两个经典算法。Prim 算法采用最近顶点策略，时间复杂度为 $O(n^2)$，适用于求稠密网的最小生成树。Kruskal 算法采用最短链接策略，时间复杂度为 $O(e\log_2 e)$，适用于求稀疏网的最小生成树。

12. 在网图中，最短路径是两个顶点之间经历的边上权值之和最小的路径。Dijkstra 算法按路径长度递增的次序求得单源点最短路径，时间复杂度为 $O(n^2)$。Floyd 算法采用迭代的方式求得每一对顶点之间的最短路径，时间复杂度为 $O(n^3)$。

13. AOV 网是用顶点表示活动，用弧表示活动之间优先关系的有向图。AOE 网是用顶点表示事件，用有向边表示活动，用边上的权值表示活动持续时间的有向图。AOV 网和 AOE 网是工程建模的两种常用图结构。

14. 顶点序列 $v_0,v_1,\cdots,v_{n-1}$ 称为一个拓扑序列。当且仅当从顶点 v_i 到 $v_j(i\neq j)$ 有一条路径时，在顶点序列中顶点 v_i 必在 v_j 之前。判断 AOV 网是否存在回路的方法是对 AOV 网进行拓扑排序。

15. 工程的最短工期是从源点到终点的最大路径长度，具有最大路径长度的路径称为关键路径。计算工程的最短工期，找出关键活动的方法是对 AOE 网求关键路径，根据每个活动的最早开始时间和最晚开始时间判定该活动是否为关键活动。

5.2 重点难点释疑

5.2.1 深度优先遍历算法的非递归实现

深度优先遍历算法的非递归实现需要按照深度优先遍历的执行过程设置一个工作栈模拟递归实现的系统栈。首先访问起始顶点并将其入栈，然后访问栈顶顶点的未被访问的邻接点并将其入栈，如果栈顶顶点没有未被访问的邻接点，则执行出栈操作。假设图采用邻接矩阵作为存储结构，图的顶点个数不超过1000，算法如下：

```
void DFTraverse(MGraph * G, int v)
{
  int S[1000], top = -1, j;                    //顺序栈初始化
  int visited[1000] = {0};
  printf("%5c", G->vertex[v]); visited[v] = 1; S[++top] = v;
  while (top != -1)
  {
    v = S[top];
    for (j = 0; j < G->vertexNum; j++)
      if (G->edge[v][j] == 1 && visited[j] == 0)
      {
        printf("%5c", G->vertex[v]);
        visited[j] = 1; S[++top] = j;
        break;
      }
    if (j == G->vertexNum) top--;
  }
}
```

5.2.2 基于图遍历的算法设计技巧

图的遍历是图最基本的操作。遍历算法中对顶点的访问可以是任意操作，根据遍历算法的框架，适当修改访问操作，可以派生出很多关于图应用的算法。

例 5-1 设计算法求无向图的深度优先生成树。

【解答】 从连通图 $G=(V,E)$ 中任一顶点出发进行深度优先遍历，将边集 E 分成两个集合 T 和 B，其中 T 是遍历过程中经历边的集合，B 是剩余边的集合。显然，T 和 V 构成连通图 G 的一棵深度优先生成树。可以修改深度优先遍历算法，在访问某顶点的未访问的邻接点时，输出(或保存)这两个顶点之间的边。假设无向图采用邻接矩阵存储，标志数组 visited[n]设为全局变量并已初始化为0，算法如下：

```
void DFTraverse(MGraph * G, int v)
{
  visited[v] = 1;
  for (int j = 0; j < G->vertexNum; j++)
```

```
    if (G->edge[v][j] == 1 && visited[j] == 0)
    {
      printf("(%3c, %3c)", G->vertex[v], G->vertex[j]);
      DFTraverse(&G, j);
    }
}
```

例 5-2 在无向图 $G=(V,E)$ 中，对于给定的顶点 $v\in V$，求距离顶点 v 最远的一个顶点。

【解答】 利用广度优先遍历算法的层次性，从顶点 v 出发进行广度优先遍历，最后一层的顶点距离顶点 v 最远。由于题目只要求输出一个顶点，可以将最后一个出队的顶点作为结果。假设图采用邻接矩阵存储，图的顶点个数不超过 1000，算法如下：

```
int MaxDist(MGraph * G, int v)
{
  int Q[1000], front = -1, rear = -1;          //顺序队列 Q 初始化
  int visited[1000] = {0}, j;
  Q[++rear] = v; visited[v] = 1;
  while (front != rear)
  {
    v = Q[++front];
    for (j = 0; j < G->vertexNum; j++)
      if (G->edge[v][j] == 1 && visited[j] == 0)
      {
        Q[++rear] = j; visited[j] = 1;
      }
  }
  return v;                                    //v 是最后一个访问的顶点
}
```

5.2.3 有向图的强连通分量

强连通分量是非强连通图的极大强连通子图。深度优先遍历是求强连通分量的一个有效方法，求解步骤如下：

(1) 从某个顶点出发进行深度优先遍历，按其所有邻接点均已访问(即出栈)的顺序将顶点排列起来。

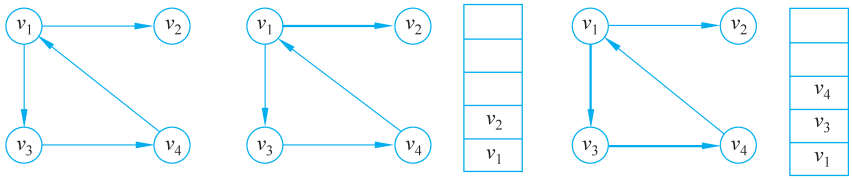
(2) 从最后访问的顶点出发，沿着以该顶点为头的弧作逆向的深度优先遍历。若此次遍历不能访问到有向图中所有顶点，则从余下的顶点中最后访问的顶点出发，继续作逆向深度优先遍历，直至有向图中所有顶点都被访问到。

(3) 每一次逆向深度优先遍历访问的顶点集便是该有向图的一个强连通分量的顶点集。若仅作一次逆向深度优先遍历就能访问图中的所有顶点，则该有向图是强连通图。

例 5-3 对于图 5-2(a)所示的有向图，求它的强连通分量。

【解答】 从顶点 v_1 出发作深度优先遍历。在访问顶点 v_2 后，顶点 v_2 不存在未访问

的邻接点，将 v_2 出栈，如图 5-2(b)所示。再从顶点 v_1 出发，在访问顶点 v_3 和 v_4 后，顶点 v_4 不存在未访问的邻接点，将 v_4 出栈；顶点 v_3 不存在未访问的邻接点，将 v_3 出栈；顶点 v_1 不存在未访问的邻接点，将 v_1 出栈。以上过程如图 5-2(c)所示，得到出栈的顶点序列 v_2,v_4,v_3,v_1。接下来从最后一个出栈的顶点 v_1 出发作逆向深度优先遍历，得到顶点集$\{v_1,v_4,v_3\}$；再从顶点 v_2 出发作逆向深度优先遍历，得到顶点集$\{v_2\}$，如图 5-2(d)所示。最终得到该有向图的两个强连通分量如图 5-2(e)所示。



(a) 一个有向图　(b) 将v_2出栈　(c) 将v_4、v_3、v_1出栈

(d) 分别从v_1和v_2出发作逆向深度优先遍历　(e) 得到两个强连通分量

图 5-2　有向图强连通分量的求解过程

5.3　习 题 解 析

一、单项选择题

1. 具有 n 个顶点的有向完全图共有(　　)条边。

A. $n(n-1)/2$　　B. $n(n-1)$　　C. $n(n+1)/2$　　D. n^2

【解答】 B

【分析】 在有向完全图中，任意两个顶点之间存在方向相反的两条弧。

2. 具有 n 个顶点的强连通图至少有(　　)条边。

A. n　　B. $n+1$　　C. $n-1$　　D. $n(n-1)$

【解答】 A

【分析】 边数最少的强连通图是一个环状有向图。

3. 在 n 个顶点的连通图中，任意一条简单路径的长度都不可能超过(　　)。

A. 1　　B. $n/2$　　C. $n-1$　　D. n

【解答】 C

【分析】 若路径长度超过 $n-1$，则路径中必存在重复的顶点。

4. 无向图 G 有 16 条边，度为 4 的顶点有 3 个，度为 3 的顶点有 4 个，其余顶点的度均小于 3，则图 G 至少有(　　)个顶点。

A. 10　　B. 11　　C. 12　　D. 13

【解答】 B

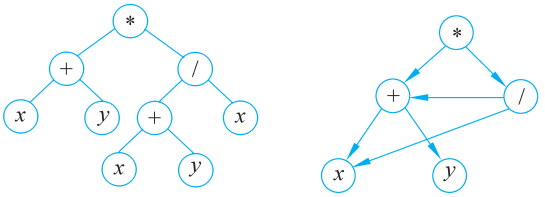
【分析】 根据顶点的度数之和与边数之间的关系，可以列出不等式：$3\times4+4\times3+(x-3-4)\times2\geqslant16\times2$，解得 x 至少为 11。

5. 用有向无环图描述表达式$(x+y)((x+y)/x)$，需要的顶点个数至少是(　　)。

A. 5　　B. 6　　C. 8　　D. 9

【解答】 A

【分析】 将算术表达式表示为二叉树，再合并相同子树和相同结点，将二叉树的父子关系修改为有向图的邻接关系，如图 5-3 所示。



(a) 二叉表示树　　(b) 有向无环图

图 5-3　第 5 题的分析

6. 有向图的邻接矩阵是一个(　　)。

A. 上三角矩阵　　B. 下三角矩阵

C. 对称矩阵　　D. 无规律的矩阵

【解答】 D

【分析】 有向图的邻接矩阵不一定对称。由于图中任意两个顶点之间都可能存在关系，因此，邻接矩阵的元素分布没有规律。

7. 具有 n 个顶点、e 条边的无向图采用邻接矩阵存储，邻接矩阵中零元素的个数为(　　)。

A. e　　B. $2e$　　C. n^2-e　　D. n^2-2e

【解答】 D

【分析】 无向图的邻接矩阵共有 n^2 个元素，每条边在邻接矩阵中以对称位置的两个非零元素表示，即非零元素的个数为 $2e$，则零元素的个数为 n^2-2e。

8. 具有 n 个顶点的无向图采用邻接表存储，邻接表中最多有(　　)个边表结点。

A. n^2　　B. $n(n-1)$　　C. $n(n+1)$　　D. $n(n-1)/2$

【解答】 B

【分析】 无向图最多有 $n(n-1)/2$ 条边，在邻接表中每条边对应两个边表结点。

9. 在有向图的邻接表存储结构中，顶点 v 在边表中出现的次数是(　　)。

A. 顶点 v 的度　　B. 顶点 v 的出度

C. 顶点 v 的入度　　D. 依附于顶点 v 的边数

【解答】 C

【分析】 在有向图的邻接表存储结构中，顶点 v 的出度是该顶点出边表的结点个数，顶点 v 的入度是该顶点在所有边表中出现的次数。

10. 设图的邻接矩阵存储如图 5-4 所示，各顶点的度依次是(　　)。

A. 1、2、1、2　　B. 2、2、1、1　　C. 3、4、2、3　　D. 4、4、2、2

【解答】 C

【分析】 邻接矩阵是非对称矩阵，说明此图是有向图，各顶点的度等于该顶点的出度与入度之和。各顶点的出度等于邻接矩阵中各行的非 0 元素个数，入度等于各列的非 0 元素个数，因此各顶点的出度依次是 2、2、1、1，入度依次是 1、2、1、2，则各顶点的度依次是 3、4、2、3。

$$\begin{pmatrix} 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{pmatrix}$$

图 5-4　第 10 题图

11. 假设非连通无向图 G 有 28 条边，则该图至少有(　　)个顶点。

A. 6　　B. 7　　C. 8　　D. 9

【解答】 D

【分析】 含有 n 个顶点的无向图，边数 $e \leqslant n(n-1)/2$，将 $e=28$ 代入得 $n \geqslant 8$，已知无向图 G 非连通，则 $n=9$。

12. 设无向图 $G=(V, E)$ 和 $G'=(V', E')$，如果 G' 是 G 的生成树，下列说法中错误的是(　　)。

A. G' 为 G 的极小连通子图　　B. G' 为 G 的连通分量

C. $|V'|=|V|$ 且 $|E'| \leqslant |E|$　　D. G' 为 G 的无环子图

【解答】 B

【分析】 连通分量是无向图的极大连通子图，其中极大的含义是包括依附于连通分量中顶点的所有边，所以连通分量可能存在回路。

13. 设有向图 $G=(V,E)$，顶点集 $V=\{v_0, v_1, v_2, v_3\}$，边集 $E=\{<v_0, v_1>, <v_0, v_2>, <v_0, v_3>, <v_1, v_3>\}$。从顶点 v_0 开始对图进行深度优先遍历，得到的遍历序列个数是(　　)。

A. 2　　B. 3　　C. 4　　D. 5

【解答】 D

【分析】 根据顶点集 V 和边集 E 画出的有向图如图 5-5 所示，深度优先遍历序列是 $v_0v_1v_3v_2$、$v_0v_2v_3v_1$、$v_0v_2v_1v_3$、$v_0v_3v_1v_2$、$v_0v_3v_2v_1$。

14. 对如图 5-6 所示的有向图从顶点 a 出发进行深度优先遍历，不可能得到的遍历序列是(　　)。

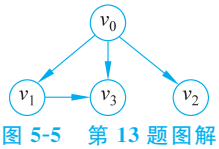


图 5-5　第 13 题图解

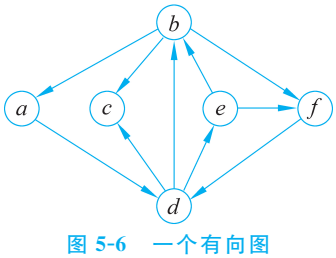


图 5-6　一个有向图

A. $adbefc$　　B. $adcefb$　　C. $adcbfe$　　D. $adefbc$

【解答】 A

【分析】 对于选项 A，访问顶点 adb 后可以访问顶点 c 和 f，但不能访问顶点 e。

15. 设无向连通图 G 含有 n 个顶点，下列说法中正确的是(　　)。

A. 只要图 G 中没有权值相同的边，其最小生成树一定唯一

B. 只要图 G 中有权值相同的边，其最小生成树一定不唯一

C. 从图 G 中选取 $n-1$ 条权值最小的边，即可构成最小生成树

D. 含有 n 个顶点 $n-1$ 条边的子图一定是图 G 的生成树

【解答】 A

【分析】 如果无向连通网中权值相同的边都不在最小生成树中(例如相同的权值都很大)，其最小生成树是唯一的。从图 G 中选取 $n-1$ 条权值最小的边构成子图，如果该子图中没有包含图 G 的全部顶点，或者选取的 $n-1$ 条边不能使子图连通，则不能构成生成树。

16. 在如图 5-7 所示的无向连通网图中，从顶点 d 开始用 Prim 算法构造最小生成树，在构造过程中加入最小生成树的前 4 条边依次是(　　)。

A. $(d,f)4$，$(f,e)2$，$(f,b)3$，$(b,a)5$

B. $(f,e)2$，$(f,b)3$，$(a,c)3$，$(f,d)4$

C. $(d,f)4$，$(f,e)2$，$(a,c)3$，$(b,a)5$

D. $(d,f)4$，$(d,b)5$，$(f,e)2$，$(b,a)5$

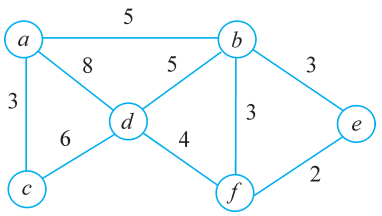


图 5-7　一个无向连通网

【解答】 A

【分析】 需要执行 Prim 算法，简便方法如下：先将顶点 d 涂黑，然后选取一个顶点涂黑而另一个顶点未涂黑的边中权值最小的边，为$(d,f)4$；将顶点 f 涂黑，选取一个顶点涂黑而另一个顶点未涂黑的边中权值最小的边，为$(f,e)2$；将顶点 e 涂黑，选取一个顶点涂黑而另一个顶点未涂黑的边中权值最小的边，为$(f,b)3$ 或$(e,b)3$；将顶点 b 涂黑，选取一个顶点涂黑而另一个顶点未涂黑的边中权值最小的边，为$(b,a)5$。

17. 设图采用邻接表存储，Prim 算法的时间复杂度为(　　)。

A. $O(n)$　　B. $O(n+e)$　　C. $O(n^2)$　　D. $O(n^3)$

【解答】 C

【分析】 Prim 算法采用邻接矩阵和邻接表作为存储结构的时间复杂度均为 $O(n^2)$，但是采用邻接矩阵存储能够快速读取任意两个顶点之间边的权值。

18. Kruskal 算法适合于求(　　)的最小生成树。

A. 稀疏图　　B. 稠密图　　C. 连通图　　D. 有向图

【解答】 A

【分析】 Kruskal 算法采用边集数组作为存储结构，时间复杂度为 $O(e\ \log_2 e)$，因此适合求边数较少(即稀疏图)的最小生成树。

19. 设图采用邻接矩阵存储，Dijkstra 算法的时间复杂度为(　　)。

A. $O(n^3)$　　B. $O(n+e)$　　C. $O(n^2)$　　D. $O(ne)$

【解答】 C

【分析】 Dijkstra 算法需要进行 $n-1$ 次迭代，每次迭代需要在辅助数组中求最小

值，时间复杂度为 $O(n^2)$。

20. 下列说法中正确的是(　　)。

A. 最短路径一定是简单路径

B. Dijkstra 算法不适用于有回路的网图

C. Dijkstra 算法不适用于求任意两个顶点间的最短路径

D. 在 Floyd 算法的求解过程中，$Path_{k-1}[i][j]$一定是 $Path_k[i][j]$的子集

【解答】 A

【分析】 最短路径如果不是简单路径，则把路径上的回路去掉可得到路径长度更短的路径。有向网图中存在回路对 Dijkstra 算法没有影响。可以将图的 n 个顶点分别作为源点调用 Dijkstra 算法，因此，Dijkstra 算法也适用于求任意两个顶点间的最短路径。用 Floyd 算法求任意两个顶点间的最短路径，$Path_k[i][j]$在 $Path_{k-1}[i][j]$的基础上进行迭代，通常 $Path_{k-1}[i][j]$不是 $Path_k[i][j]$的子集。

21. 若有向图的全部顶点不能形成一个拓扑序列，则可断定该有向图(　　)。

A. 是有根有向图　　B. 是强连通图

C. 含有多个入度为 0 的顶点　　D. 含有顶点数大于 1 的强连通分量

【解答】 D

【分析】 强连通分量一定存在回路，但存在回路的不一定是强连通图。

22. 对于如图 5-8 所示的 AOE 网，事件 v_4 的最早开始时间是(　　)，最迟开始时间是(　　)，该 AOE 网的关键路径有(　　)条。

A. 11　　B. 12　　C. 13　　D. 14

E. 1　　F. 2　　G. 3　　H. 4

【解答】 C，C，F

【分析】 事件 v_4 的最早开始时间是从顶点 v_1 到 v_4 的最长路径长度 13，事件 v_4 的最迟开始时间＝min{事件 v_6 的最迟开始时间－11，事件 v_5 的最迟开始时间－9}＝{24－11，22－9}＝13。该 AOE 网的两条关键路径如图 5-9 所示。

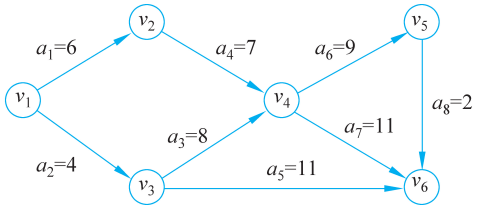


图 5-8　一个 AOE 网

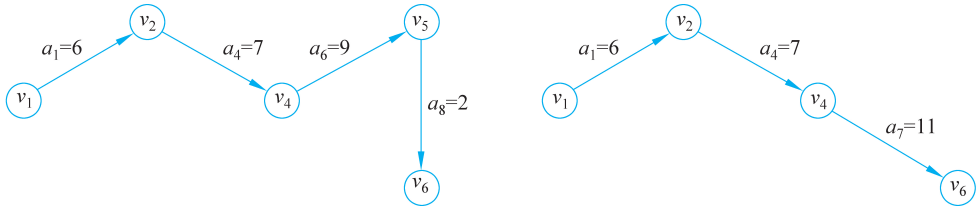


图 5-9　两条关键路径

23. 在有向图 G 的拓扑序列中，若顶点 v_i 出现在顶点 v_j 之前，则下列情形中不可能出现的是(　　)。

A. 图 G 中有弧$<v_i,v_j>$　　B. 图 G 中有一条从 v_i 到 v_j 的路径

C. 图 G 中没有弧$<v_i, v_j>$　　D. 图 G 中有一条从 v_j 到 v_i 的路径

【解答】 D

【分析】 在有向图 G 的拓扑序列中，若顶点 v_i 出现在顶点 v_j 之前，则从顶点 v_i 到顶点 v_j 一定存在路径。路径长度可以是 1，此时图 G 中有弧$<v_i,v_j>$；也可以大于 1，此时图 G 中可以没有弧$<v_i, v_j>$。

24. 在 AOE 网中，关键路径是(　　)。

A. 从源点到终点的最长路径　　B. 从源点到终点的最短路径

C. 从源点到终点边数最多的路径　　D. 从源点到终点边数最少的路径

【解答】 A

【分析】 在 AOE 网中，所有活动都完成才能到达终点，因此完成整个工程所必须花费的时间(即最短工期)应该为源点到终点的最大路径长度。具有最大路径长度的路径称为关键路径。

25. 关于工程计划的 AOE 网，下列说法中不正确的是(　　)。

A. 关键活动不按期完成就会影响整个工程的完成时间

B. 任何一个关键活动提前完成，整个工程将会提前完成

C. 所有的关键活动都提前完成，整个工程将会提前完成

D. 某些关键活动若提前完成，整个工程将会提前完成

【解答】 B

【分析】 AOE 网的关键路径可能不止一条，如果某一个关键活动提前完成，只能减少该关键活动所在路径的长度，其他关键路径的长度不变。

二、解答下列问题

1. 无向网图含有 n 个顶点和 e 条边，每个顶点的信息(假设只存储编号)占用 2 字节，每条边的权值信息占用 4 字节，每个指针占用 4 字节，计算采用邻接矩阵和邻接表分别占用多少存储空间。

【解答】 在邻接矩阵存储结构中，存储顶点信息的一维数组需要 $2n$ 字节，存储权值信息的二维数组需要 $4n^2$ 字节，因此，邻接矩阵共需要 $4n^2+2n$ 字节。

在邻接表存储结构中，顶点表存储顶点信息和边表的头指针，需要$(2+4)n=6n$ 字节；边表中共 $2e$ 个顶点，每个顶点需要存储顶点编号、权值和指针信息，需要$(2+4+4)\times 2e=20e$ 字节，因此，邻接表共需要 $6n+20e$ 字节。

2. 无向图含有 n 个顶点和 e 条边，判断任意两个顶点 i 和 j 是否有边相连，如果采用邻接矩阵存储，该操作的时间复杂度是多少？如果采用邻接表存储，该操作的时间复杂度是多少？请给出简要分析过程。

【解答】 如果采用邻接矩阵存储，只需判断元素 edge[i][j](或 edge[j][i])是否等于 1，时间复杂度是 $O(1)$。如果采用邻接表存储，需要判断第 i 个边表中是否含有顶点 j，这需要遍历第 i 个边表，时间复杂度是 $O(e/n)$。