16337341_朱志儒_数据结构作业（四）

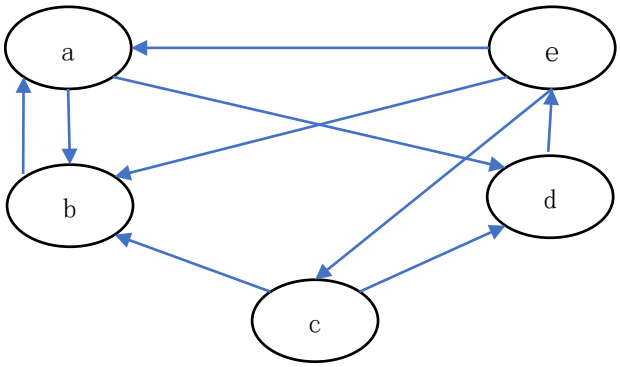
1、 （1）设顶点的度数之和为 D，边数之和为 E，则 D = 2E

（2）有向图中顶点的入读之和等于出度之和

（3）具有 n 个顶点的无向图，至少应有 n-1 条边才能确保是一个连通图，若采用邻接矩阵表示，则该矩阵为 n*n 矩阵

（4）具有 n 个顶点的有向图，至少有 n 条弧才能确保是强连通图

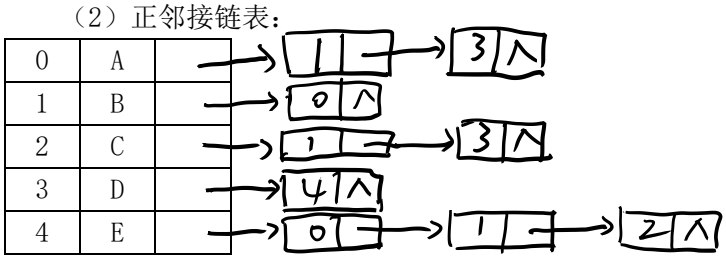
2、 （1）有向图：



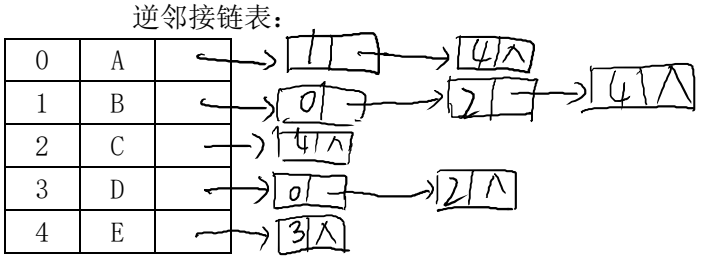
入度：a:2 b:3 c:1 d:2 e:1

出度：a:2 b:1 c:2 d:1 e:3

（2）正邻接链表：



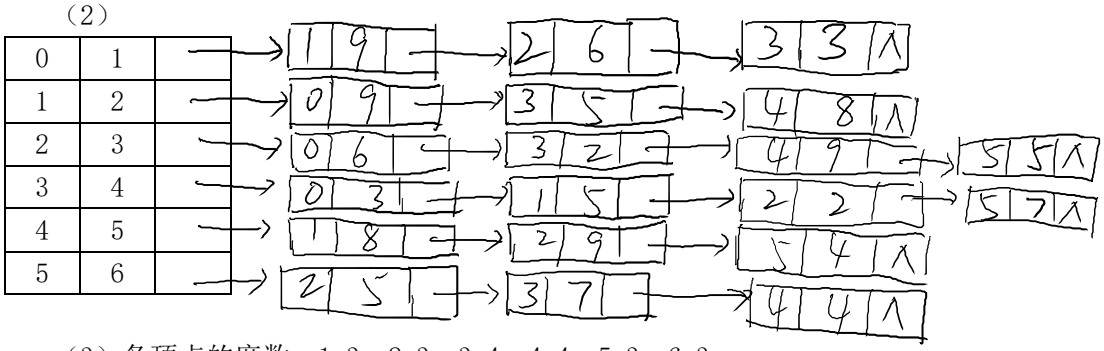
逆邻接链表：



3、（1）邻接矩阵：

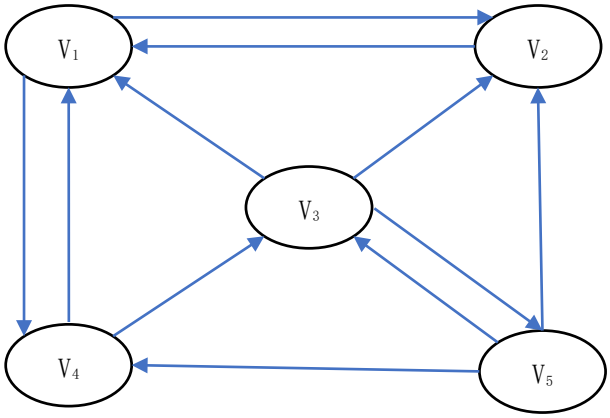
$$\begin{bmatrix} 0 & 9 & 6 & 3 & \infty & \infty \\ 9 & 0 & \infty & 5 & 8 & \infty \\ 6 & \infty & 0 & 2 & 9 & 5 \\ 3 & 5 & 2 & 0 & \infty & 7 \\ \infty & 8 & 9 & \infty & 0 & 4 \\ \infty & \infty & 5 & 7 & 4 & 0 \end{bmatrix}$$

（2）



（3）各顶点的度数：1:3　2:3　3:4　4:4　5:3　6:3

4、（1）有向图：



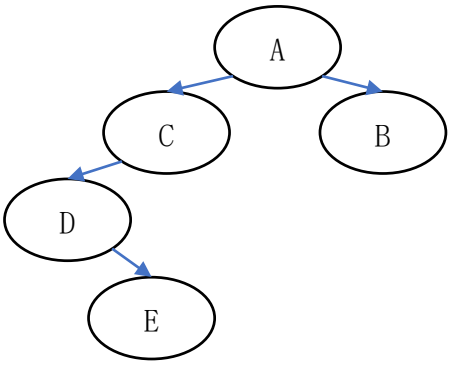
（2）邻接矩阵：

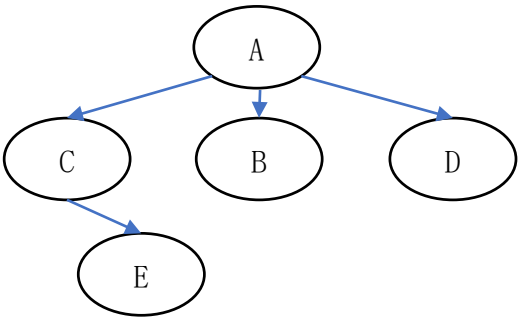
$$\begin{bmatrix} 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 \end{bmatrix}$$

（3）深度优先遍历序列：a，c，d，e，b

广度优先遍历序列：a，c，b，d，e

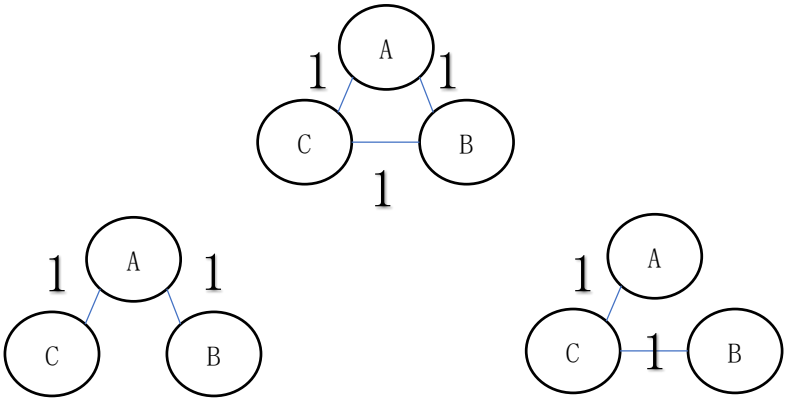
（4）深度优先生成树：



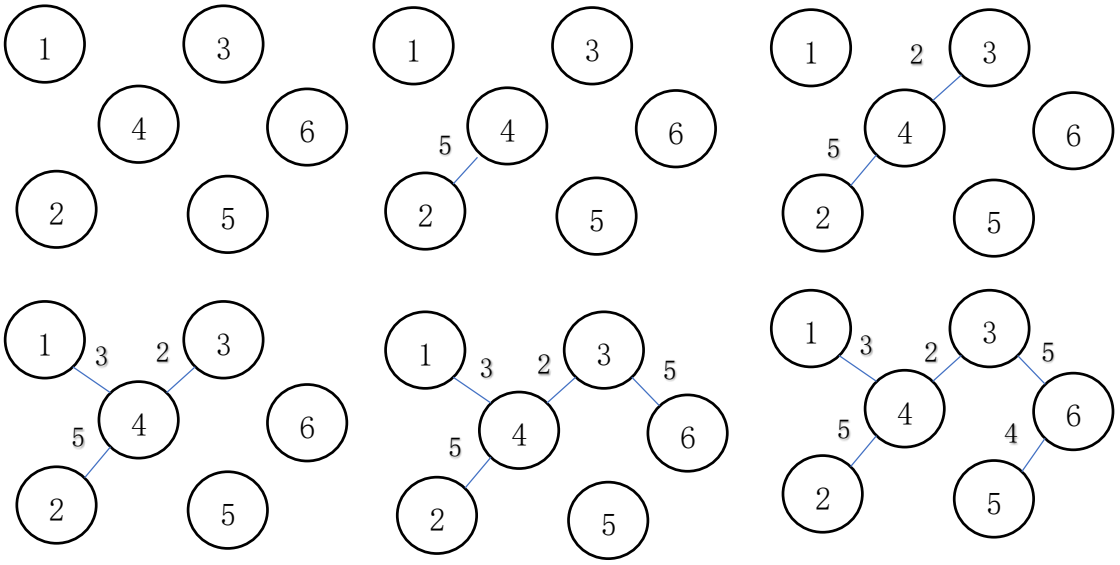


5、 一个带权连通图的最小生成树不是唯一的

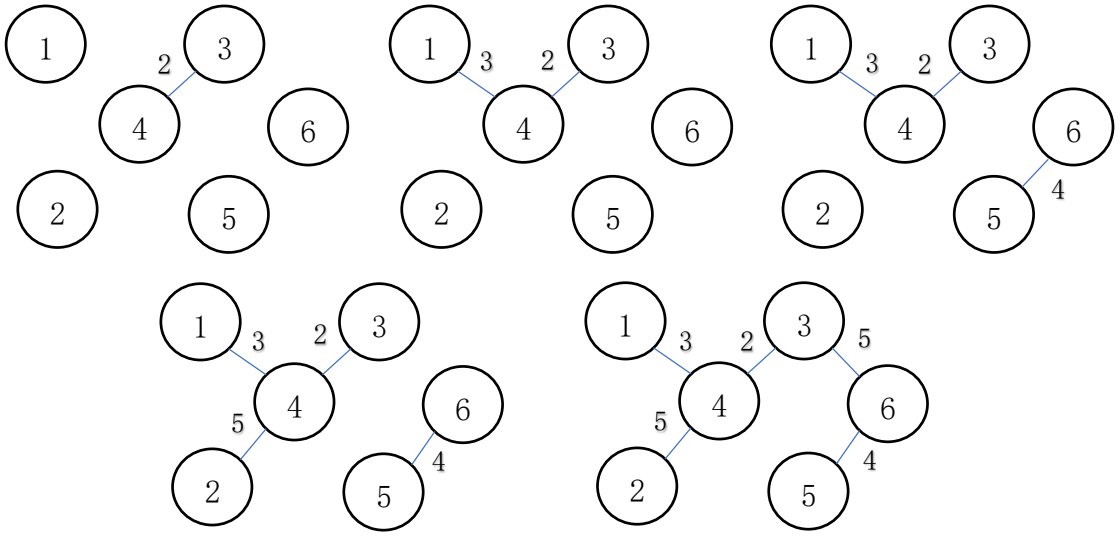
例子：



6、 （1）Prime 算法：



（2）Kruskal 算法：



7、 Dijkstra 算法：

源点	终点	最短路径	路径长度
V4	V1	(V4,V2,V1)	∞,∞,30
	V2	(V4,V2)	20
	V3	(V4,V2,V1,V3)	∞,∞,∞,45
	V5	(V4,V2,V5)	∞,∞,50
	V6	(V4,V6)	15

8、Floyd 算法：

$$A_{-1}=\begin{bmatrix}0&5&3&\infty&\infty&\infty\\ \infty&0&\infty&\infty&9&4\\ \infty&\infty&0&2&5&\infty\\ \infty&\infty&\infty&0&4&\infty\\ 6&\infty&\infty&\infty&0&\infty\\ \infty&\infty&\infty&\infty&3&0\end{bmatrix}\qquad Path_{-1}=\begin{bmatrix}-1&-1&-1&-1&-1&-1\\ -1&-1&-1&-1&-1&-1\\ -1&-1&-1&-1&-1&-1\\ -1&-1&-1&-1&-1&-1\\ -1&-1&-1&-1&-1&-1\\ -1&-1&-1&-1&-1&-1\end{bmatrix}$$

$$A_a=\begin{bmatrix}0&5&3&\infty&\infty&\infty\\ \infty&0&\infty&\infty&9&4\\ \infty&\infty&0&2&5&\infty\\ \infty&\infty&\infty&0&4&\infty\\ 6&11&9&\infty&0&\infty\\ \infty&\infty&\infty&\infty&3&0\end{bmatrix}\qquad Path_a=\begin{bmatrix}-1&-1&-1&-1&-1&-1\\ -1&-1&-1&-1&-1&-1\\ -1&-1&-1&-1&-1&-1\\ -1&-1&-1&-1&-1&-1\\ -1&a&a&-1&-1&-1\\ -1&-1&-1&-1&-1&-1\end{bmatrix}$$

$$A_b=\begin{bmatrix}0&5&3&\infty&14&9\\ \infty&0&\infty&\infty&9&4\\ \infty&\infty&0&2&5&\infty\\ \infty&\infty&\infty&0&4&\infty\\ 6&11&9&\infty&0&15\\ \infty&\infty&\infty&\infty&3&0\end{bmatrix}\qquad Path_b=\begin{bmatrix}-1&-1&-1&-1&b&b\\ -1&-1&-1&-1&-1&-1\\ -1&-1&-1&-1&-1&-1\\ -1&-1&-1&-1&-1&-1\\ -1&a&a&-1&-1&b\\ -1&-1&-1&-1&-1&-1\end{bmatrix}$$

$$A_c=\begin{bmatrix}0&5&3&5&8&9\\ \infty&0&\infty&\infty&9&4\\ \infty&\infty&0&2&5&\infty\\ \infty&\infty&\infty&0&4&\infty\\ 6&11&9&11&0&15\\ \infty&\infty&\infty&\infty&3&0\end{bmatrix}\qquad Path_c=\begin{bmatrix}-1&-1&-1&c&c&b\\ -1&-1&-1&-1&-1&-1\\ -1&-1&-1&-1&-1&-1\\ -1&a&a&c&-1&b\\ -1&-1&-1&-1&-1&-1\end{bmatrix}$$

$$A_d=\begin{bmatrix}0&5&3&5&8&9\\ \infty&0&\infty&\infty&9&4\\ \infty&\infty&0&2&5&\infty\\ \infty&\infty&\infty&0&4&\infty\\ 6&11&9&11&0&15\\ \infty&\infty&\infty&\infty&3&0\end{bmatrix}\qquad Path_d=\begin{bmatrix}-1&-1&-1&c&c&b\\ -1&-1&-1&-1&-1&-1\\ -1&-1&-1&-1&-1&-1\\ -1&-1&-1&-1&-1&-1\\ -1&a&a&c&-1&b\\ -1&-1&-1&-1&-1&-1\end{bmatrix}$$

$$A_e=\begin{bmatrix}0&5&3&5&8&9\\ 15&0&18&20&9&4\\ 11&16&0&2&5&20\\ 10&15&13&0&4&19\\ 6&11&9&11&0&15\\ 9&14&12&14&3&0\end{bmatrix}\qquad Path_e=\begin{bmatrix}-1&-1&-1&c&c&b\\ e&-1&e&e&-1&-1\\ e&e&-1&-1&-1&e\\ e&e&e&-1&-1&e\\ -1&a&a&c&-1&b\\ e&e&e&e&-1&-1\end{bmatrix}$$

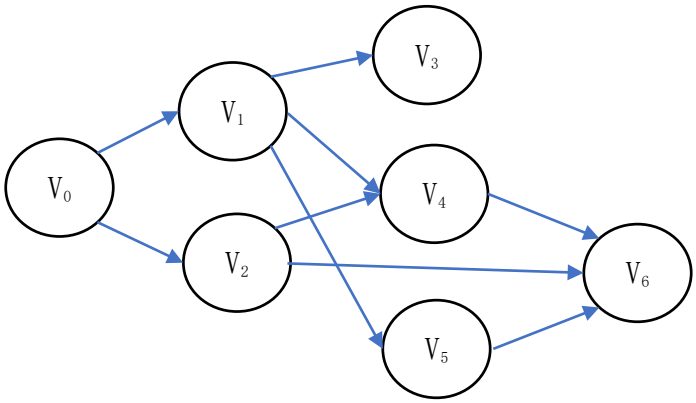
$$A_f=\begin{bmatrix}0&5&3&5&8&9\\ 13&0&16&18&7&4\\ 11&16&0&2&5&20\\ 10&15&13&0&4&19\\ 6&11&9&11&0&15\\ 9&14&12&14&3&0\end{bmatrix}\qquad Path_f=\begin{bmatrix}-1&-1&-1&c&c&b\\ f&-1&f&f&f&-1\\ e&e&-1&-1&-1&e\\ e&e&e&-1&-1&e\\ -1&a&a&c&-1&b\\ e&e&e&e&-1&-1\end{bmatrix}$$

每对顶点之间的最短路径及路径长度：

(a, b|5), (a, c|3), (a, c, d|5), (a, c, e|8), (a, b, f|9), (b, f, e, a|13), (b, f, e, a, c|16),

(b, f, e, a, c, d|18), (b, f, e|7), (b, f|4), (c, e, a|11), (c, e, a, b|16), (c, d|2), (c, e|5), (c, e, a, b, f|20), (d, e, a|10), (d, e, a, b|15), (d, e, a, c|13), (d, e|4), (d, e, a, b, f|19), (e, a|6), (e, a, b|11), (e, a, c|9), (e, a, c, d|11), (e, a, b, f|15), (f, e, a|9), (f, e, a, b|14), (f, e, a, c|12), (f, e, a, c, d|14), (f, e|3)

9、 由 AOV 网的邻接矩阵可得 AOV 网的图：



由图可得 AOV 网的一个拓扑排序：V_0, V_1, V_2, V_3, V_4, V_5, V_6

10、V9, V8, V1, V2, V4, V3, V7, V6, V5　　V9, V8, V1, V2, V4, V7, V6, V3, V5
V9, V8, V1, V2, V4, V7, V3, V6, V5　　V9, V1, V8, V2, V4, V3, V7, V6, V5
V9, V1, V8, V2, V4, V7, V6, V3, V5　　V9, V1, V8, V2, V4, V7, V3, V6, V5
V9, V1, V2, V8, V4, V3, V7, V6, V5　　V9, V1, V2, V8, V4, V7, V6, V3, V5
V9, V1, V2, V8, V4, V7, V3, V6, V5　　V9, V1, V2, V4, V8, V3, V7, V6, V5
V9, V1, V2, V4, V8, V7, V6, V3, V5　　V9, V1, V2, V4, V8, V7, V3, V6, V5
V9, V1, V2, V4, V3, V7, V8, V6, V5　　V9, V1, V2, V4, V3, V8, V7, V6, V5
V9, V1, V2, V4, V7, V3, V8, V6, V5　　V9, V1, V2, V4, V7, V8, V3, V6, V5
V9, V1, V2, V4, V7, V8, V6, V3, V5　　V1, V9, V8, V2, V4, V3, V7, V6, V5
V1, V9, V8, V2, V4, V7, V6, V3, V5　　V1, V9, V8, V2, V4, V7, V3, V6, V5
V1, V9, V2, V8, V4, V3, V7, V6, V5　　V1, V9, V2, V8, V4, V7, V6, V3, V5
V1, V9, V2, V8, V4, V7, V3, V6, V5　　V1, V9, V2, V4, V8, V3, V7, V6, V5
V1, V9, V2, V4, V8, V7, V6, V3, V5　　V1, V9, V2, V4, V8, V7, V3, V6, V5
V1, V9, V2, V4, V3, V7, V8, V6, V5　　V1, V9, V2, V4, V3, V8, V7, V6, V5
V1, V9, V2, V4, V7, V3, V8, V6, V5　　V1, V9, V2, V4, V7, V8, V3, V6, V5
V1, V9, V2, V4, V7, V8, V6, V3, V5　　V1, V2, V9, V8, V4, V3, V7, V6, V5
V1, V2, V9, V8, V4, V7, V6, V3, V5　　V1, V2, V9, V8, V4, V7, V3, V6, V5
V1, V2, V9, V4, V8, V3, V7, V6, V5　　V1, V2, V9, V4, V8, V7, V6, V3, V5
V1, V2, V9, V4, V8, V7, V3, V6, V5　　V1, V2, V9, V4, V3, V7, V8, V6, V5
V1, V2, V9, V4, V3, V8, V7, V6, V5　　V1, V2, V9, V4, V7, V3, V8, V6, V5
V1, V2, V9, V4, V7, V8, V3, V6, V5　　V1, V2, V9, V4, V7, V8, V6, V3, V5
V1, V2, V4, V9, V8, V3, V7, V6, V5　　V1, V2, V4, V9, V8, V7, V6, V3, V5
V1, V2, V4, V9, V8, V7, V3, V6, V5　　V1, V2, V4, V9, V3, V7, V8, V6, V5
V1, V2, V4, V9, V3, V8, V7, V6, V5　　V1, V2, V4, V9, V7, V3, V8, V6, V5
V1, V2, V4, V9, V7, V8, V3, V6, V5　　V1, V2, V4, V9, V7, V8, V6, V3, V5
V1, V2, V4, V3, V9, V8, V7, V6, V5　　V1, V2, V4, V3, V9, V7, V8, V6, V5

11、基于 DFS 的拓扑排序：

```
#include <stack>
#include <iostream>
using namespace std;
void DFS(Graph G, int v, bool visited[], stack<int> &reversepost) {
    visited[v] = true;
    int w = GetFistNeighbor(G, v);
    while (w != -1) {
        if (!visited[w]) DFS(G, w, visited, reversepost);
        w = GetNextNeighbor(G, v, w);
    }
    reversepost.push(v);
    //在即将退出 dfs 方法的时候，将当前顶点添加到结果集中
}
void DFS_Order(Graph G) {
    stack<int> reversepost;//使用栈来保存最后的结果
    bool *visited = new bool[G.vertices_num];
    for (int i = 0; i < G.n; ++i) visited[i] = false;
    for (int i = 0; i < G.n; ++i) {
        if (!visited[i]) {
            DFS(G, i, visited, reversepost);
        }
    }
    if (!reversepost.empty()) {
        cout << reversepost.top();
        reversepost.pop();
        if (!reversepost.empty()) cout << "->";
        else cout << endl;
    }
};
```

12、

```
void path(AdjList adj, int vi, int vj, int len)
{
    int v, w, top = 0, dist = 0, head;
    int stack[max], visited[max]; //定义栈和标志数组
    struct vnode *p;
    v = vi;
    visited[v] = 1;
    head = 1 //head 在邻接表头第一次取邻接顶点时为 1，否则为 0；
    do //w 为 v 在图中的邻接点，若邻接点已经查遍，则 w=0
    {
        if (head != 0) p = adj[v]->firstarc;
            else
        {
```

```
            head = 0;
        p = p->next;

    }
         if (p = NULL) w = 0;
         else      w = p->adjvex;
         if (w != 0)
 if (visited[w] == 0) //顶点 v 未被访问过
 if (w = vj && dist == len - 1)

    {
            dist++;
        top++;
        stack[top] = v;

    }
         if (w != vj && dist < len - 1)

    {
            dist++;
        top++;
        stack[top] = v;
        visited[w] = 1;
        v = w;
        head = 1

    }
         else      if (top > 0)

    {
            visited[v] = 0;
        v = stack[top];
        head = 1;
        top--;
        dist--;

    }
}
while ((top != 0 || w == 0) && (dist != len))
    ;
    if (top > 0)
for (i = 1; i <= top; i++)
 printf("%d", stack[i]);
     else      printf("没有这样的路径! \n");
```

}

13、（1）

事件	V0	V1	V2	V3	V4	V5	V6	V7	V8	V9
Ve	0	5	6	18	21	21	23	25	28	30
V1	0	15	6	18	22	26	23	26	28	30

（2）该工程完工至少需要 30 时间

（3）

	0-1	0-2	1-3	2-3	2-4	3-4	3-5	3-6	4-6	4-7	5-9	6-8	7-8	8-9
E	0	0	5	6	6	18	18	18	21	21	21	23	25	28
L	10	0	15	6	19	19	23	18	22	22	26	23	26	28
L-E	10	0	10	0	13	1	5	0	1	1	5	0	1	0

关键路径：V0 -> V2 -> V3 -> V6 -> V8 -> V9

关键活动：V0->V2, V2->V3, V3->V6, V6->V8, V8->V9