作业11解答：

1. 给出一个O(e)时间的算法，判定一个图是否2-连通图。

解：算法如下：

```
PROCEDURE  SEARCHB (v);
BEGIN
    mark v "old";
    DFNUMBER[v] := COUNT;
    COUNT := COUNT+1;
    LOW[v] := DFNUMBER[v];
    FOR each vertex w on L[v] DO
    BEGIN
      IF (v, w) is not in STACK THEN
        Push((v, w), STACK);
      IF w is marked "new" THEN
      BEGIN
        add (v, w) to T;
        FATHER[w] := v;
        SEARCHB(w);
        IF LOW[w] ≥ DFNUMBER[v] THEN
        BEGIN  /* 找到一个新的2-连通分支 */
          number := number+1;  /* 2连通分支数加1 */
          REPEAT  /* 将该2-连通分支弹出栈 */
            (x, y) := Pop (STACK);
          UNTIL (x, y) = (v, w);
        END;
```

找二连通分支的个数

```
        LOW[v] := MIN(LOW[v], LOW[w])
    END
    ELSE IF w is not FATHER[v] THEN
        LOW[v] := MIN(LOW[v], DFNUMBER[w]);
  END
END;
```

主程序

```
BEGIN
  T := φ;
  COUNT := 1;
  number := 0;
  set STACK empty;
  FOR each vertex v in V DO mark v "new";
  WHILE there is a vertex v0 marked "new" DO
      SEARCHB(v0);
  IF number = 1 THEN RETURN("true")
  ELSE RETURN("false");
END;
```

2. 利用BFS的思想，设计一个算法，找出一个图G中经过某一指定点v的最短圈。

孩子继承父亲的标号，若发现一个顶点和一个访问过的顶点且标号不同，则找到最短圈

解：

```
PROCEDURE BFS (v: integer; adjlist: 图的邻接表;
                VAR visit: 访问标志数组);
VAR
```

装 订 线

中山大学本科生考试答题纸

学院(系)　　专业　　级

考试科目　　成绩评定

考生姓名　　教师签名

学　号　　年　月　日

警示 《中山大学授予学士学位工作细则》第八条："考试作弊者不授予学士学位。"

装 订 线

```
queue : ARRAY[1..max] OF integer;
Number of ancestor : ARRAY[1..max] OF integer;
Ancestor : ARRAY[1..max] OF integer;
j, hp, tp : integer;
p :顶点指针;
BEGIN
  FOR j:=1 TO max DO
  BEGIN  /*置初值*/
    visit[j] := false; /*顶点j设为未访问*/
    Number of ancestor[j] :=0; /*顶点j的祖先,即v
                                 的儿子的个数*/
    Ancestor[j] := 0; /*顶点j的父亲顶点*/
  END;
  hp := 1; tp := 1;
  visit[v] := true; /*设v为已访问过*/
  found_cycle := false; /*找到圈标志设为"假"*/
  j := 0;
  FOR each vertex w in adjlist[v] DO
```

队列目前还是空

```
BEGIN /*为访问v的所有相邻点w*/
  visit[w]:=true; /*标志已访问w*/
  j:=j+1;          记录祖先
  Number_of_ancestor[w]:=j;/*v的儿子w的号码*/
  Ancestor[w]:=v; /*w的父亲为v*/
  Queue[tp]:=w; /*w进队列*/
  tp:=tp+1;
END;
REPEAT
  p:=adjlist[Queue[hp]]; /*队列头顶点的邻接表
                          中第一个元素p*/
  WHILE (p≠NIL) AND NOT found_cycle DO
  BEGIN
    j:=p↑.vertex; /*p顶点的号码j*/
    IF NOT visit[j] THEN
    BEGIN
      visit[j]:=true; /*标志已访问顶点j*/
      tp:=tp+1;
      Queue[tp]:=j; /*j进队列*/
      Number_of_ancestor[j]:=
                  Number_of_ancestor[Queue[hp]];
      /*j的祖先设为j的父亲的祖先*/
      Ancestor[j]:=Queue[hp]; /*j的父亲设为队列头
                               顶点*/
END
```

```
        ELSE IF Number_of_ancestor[j]≠
                Number_of_ancestor[Queue[hp]] THEN
                BEGIN  /* 若j已访问过,且j和队列头顶点是v
                  found_cycle:=true;      的不同儿子的名代,
                  A:=j;                   则找到所求回圈 */
                  B:=Queue[hp];
                END;
            p:=p↑.link;
        END;
        hp:=hp+1;
    UNTIL (hp>tp) OR found_cycle; /* 队列空或找到圈,则
                                     循环结束 */

    IF found_cycle THEN
    BEGIN  /* 输出圈的所有边 */
      WRITE("(",A,",",B,")"); /* 打印边(A,B) */
      WHILE A≠V DO
      BEGIN
        WRITE("(",A,",",Ancestor[A],")");
                          /* 打印边(A,Ancestor[A]) */
        A:=Ancestor[A];
      END;
      WHILE B≠V DO
      BEGIN
        WRITE("(",B,",",Ancestor[B],")");
                          /* 打印边(B,Ancestor[B]) */
```

```
        B := Ancestor[B];
    END;
END
ELSE WRITE ("Cycle not found");
                /*如果没找到圈，输出"没找到圈"*/
END;
```

装 订 线