2017 年下半年软件设计师

下午试题四

比特网：http://www.bitpx.com/

试题四(共 15 分)

阅读下列说明和 C 代码，回答问题 1 至问题 2，将解答写在答题纸的对应栏内。

【说明】

一个无向连通图 G 上的哈密尔顿(Hamilton)回路是指从图 G 上的某个顶点出发，经过图上所有其他顶点一次且仅一次，最后回到该顶点的路径。一种求解无向图上的哈密尔顿回路算法的基本思想如下：

假设图 G 存在一个从顶点v_0出发的哈密尔顿回路v_0-v_1-v_2-v_3-…-v_{n-1}-v_0，算法从顶点v_0出发，访问该顶点的一个未被访问的邻接顶点v_1，接着从顶点v_1出发，访问v_1的一个未被访问的邻接顶点v_2，…。对顶点v_i，重复进行以下操作：访问v_i的一个未被访问的邻接顶点v_{i+1}；若v_i的所有邻接顶点均已被访问，则返回到顶点v_{i-1}，考虑v_{i-1}的下一个未被访问的邻接顶点，仍记为v_i；直到找到一条哈密尔顿回路或者找不到哈密尔顿回路，算法结束。

【C 代码】

下面是算法的 C 语言实现。

(1)常量和变量说明

n：图 G 中的顶点数

c[][]：图 G 的邻接矩阵

k：统计变量，当前已经访问的顶点数为 k+1

x[k]：第 k 个访问的顶点编号，从 0 开始

visited[x[k]]：第 k 个顶点的访问标志，0 表示未访问，1 表示已访问

(2)C 程序

```
#include<stdio.h>
#include<stdlib.h>
#define MAX 4

void Hamilton(int n,int x[MAX],int c[MAX][MAX]) {
    int i;
    int visited[MAX];
    int k;
    /*初始化 x 数组和 visited 数组*/
    for (i =0; i < n; i++) {
        x[i] = 0;
        visited[i] = 0;
    }
    /*访问起始顶点*/
    k = 0;
    ____(1)____;//(1)
    x[0] = 0;
    k = k + 1;
    /*访问其他顶点*/
    while (k >= 0) {
        x[k] = x[k] + 1;//题目有错误?
        while (x[k] < n) {
            if (____(2)____ && c[x[k - 1]][x[k]] == 1) { /*邻接顶点 x[k]未被访问过*///  (2)
                break;
            }
            else {
                x[k] = x[k] + 1;
```

```
                }
            }
            if (x[k] < n && k == n - 1 &&  (3)  ) {/*找到一条哈密尔顿回路*/// (3)
                for (k = 0; k < n; k++) {
                    printf("%d--", x[k]);/*输出哈密尔顿回路*/
                }
                printf("%d\n", x[0]);
                return;
            }
            else if (x[k] < n&& k < n - 1) {/*设置当前顶点的访问标志，继续下一个顶点*/
                 (4)  ;//(4)
                k = k + 1;
            }
            else { /*没有未被访问过的邻接顶点，回退到上一个顶点*/
                x[k] = 0;
                visited[x[k]] = 0;
                 (5)  ;//(5)
            }
        }
    }
```

【问题 1】(10 分)

根据题干说明，填充 C 代码中的空(1)～(5)。

【问题 2】(5 分)

根据题干说明和 C 代码，算法采用的设计策略为 (6) ，该方法在遍历图的顶点时，采用的是 (7) 方法(深度优先或广度优先)。

●试题四 参考答案及解析

【参考答案】

(1)visited[0] = 1
(2)visited[x[k]] == 0
(3)c[x[k]][0] == 1
(4)visited[x[k]] = 1
(5)k = k - 1 或 k-- 或 --k
(6)回溯法
(7)深度优先

【试题解析】

问题(1)处及上下几行代码(while 循环之前)是默认从 0 号顶点开始，“x[0] = 0”表示 0 号顶点被访问过了，“k = k + 1”也表示已经找到了一个满足条件的顶点，故空(1)处肯定是设置 0 号顶点已经被访问过了，应填写“visited[0] = 1”。

空(2)处根据注释知邻接顶点 x[k]未被访问过则执行 break，则 x[k]号顶点未被访问成立的判断条件是“visited[x[k]] == 0”，即(2)的答案。“c[x[k - 1]][x[k]] == 1”是判断之前已经被访问过的顶点(x[k - 1])与 x[k]是否为相邻顶点。

空(3)处的 if 判断表达式“找到一条哈密尔顿回路”，成立条件为 x[k] < n、且 k == n - 1，同时还要满足第 x[k]顶点未被访问过(空(2)处已经判断)、最后还要保证 x[k]号顶点与 0 号顶点之间有边(判断条件 c[x[k]][0] == 1)才行，故空(3)处应该填写“c[x[k]][0] == 1”。

空(4)处为“设置当前顶点的访问标志，继续下一个顶点”，则 k 应该加 1，且应设置 x[k]号顶点被访问过，即空(4)应该填写“visited[x[k]] = 1”。

空(5)处所属的 else 代码块表示“没有未被访问过的邻接顶点，回退到上一个顶点”，则应该进行回溯，回退到上一个顶点，回溯的过程即是取消前一步因为“试探”而作的操作，即取消之前“试探”过程中设置的顶点编号(x[k] = 0)，取消之前“试探”过程中访问过的顶点(visited[x[k]] = 0),取消之前因为“试探”而增加的顶点数量(k=k-1)，故空(5)应该填写“k=k-1”(或 k-- 或 --k)。

算法中，如下的代码块即是去查找与 x[k - 1]号顶点相邻的顶点(从 x[k]号开始“试探”)，找到一个马上执行关键字 break(即结束循环)，然后执行该 while 循环后的代码块，之后的过程将不再查找 x[k - 1]号顶点的其他相邻顶点，如果 x[k]号顶点不满足条件，则执行循环中 else 部分代码，即继续“试探”x[k] + 1 号顶点。如果在找到了一个相邻顶点的情况下，还要继续去搜索其他的相邻顶点，则为广度优先方式，本题明显不是广度优先，而是深度优先。

```
while (x[k] < n) {
if (__(2)__&& c[x[k - 1]][x[k]] == 1) { /*邻接顶点 x[k]未被访问过*///
    break;
}
else {
    x[k] = x[k] + 1;
    }
}
```

根据以上分析，再结合以下的代码块，此块代码的功能为回退到上一个顶点继续搜索上一个顶点的其它相邻顶点，同时在回溯的过程中要取消之前因为“试探”而进行的操作。

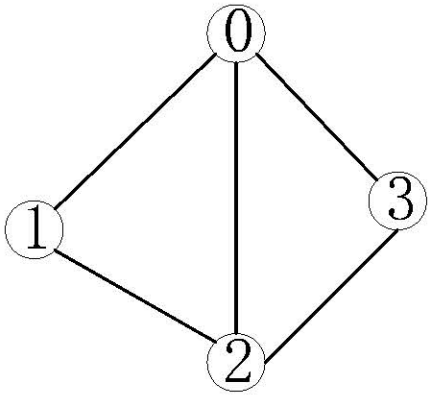
```
else { /*没有未被访问过的邻接顶点，回退到上一个顶点*/
x[k] = 0;
visited[x[k]] = 0;
_____(5)_____;//
}
```

通过以上分析，本题使用的是回溯法。回溯法有“通用的解题法”之称，也称为试探法，用它可以系统地搜索一个问题的所有解或任一解。回溯法是一个既有系统性又带有跳跃性的搜索算法。它在包含问题所有解的解空间树中，按照深度优先的策略，从根结点出发搜索解空间树，算法搜索至解空间树的任一节点时，总是先判断该点是否肯定不包含问题的解。如果肯定不包含，则跳过以该节点为根的子树的系统搜索，逐层向其祖先节点回溯，否则，进入该子树，继续按深度优先的策略进行搜索。回溯法在求问题的所有解时，要回溯到根，且根节点的所有子树都已被搜索遍才结束。而在求解任一解时，只要搜索到任一解就可以结束。这种以深度优先的方式系统地搜索问题的解的算法称为回溯法，它适用于解一些组合数较大的问题。

本题对算法的描述并结合以上对代码的分析，空(6)填写“回溯法”，空(7)填写“深度优先”。

为了让广大软考考生能更加清晰地理解本题程序的运行过程，比特培训给出如下完整的程序，考生可在环境中运行本程序，通过单步跟踪程序运行的方式来加深理解！

程序使用的无向连通图及对应的邻接矩阵为：



无向连同图

$$C=\begin{Bmatrix}0&1&1&1\\1&0&1&0\\1&1&0&1\\1&0&1&0\end{Bmatrix}$$

邻接矩阵

【完整程序代码】

```
#include<stdio.h>
#include<stdlib.h>
#define MAX 4

void Hamilton(int n,int x[MAX],int c[MAX][MAX]) {
int i;
int visited[MAX];
int k;
/*初始化 x 数组和 visited 数组*/
for (i = 0; i < n; i++) {
    x[i] = 0;
    visited[i] = 0;
}
/*访问起始顶点*/
k = 0;
visited[0] = 1;//(1)
x[0] = 0;
k = k + 1;
/*访问其他顶点*/
while (k >= 0) {
    x[k] = x[k] + 1;
    while (x[k] < n) {
        if (visited[x[k]] == 0 && c[x[k - 1]][x[k]] == 1) { /*邻接顶点 x[k]未被访问过*/ (2)
            break;
        }
        else {
            x[k] = x[k] + 1;
        }
    }
    if (x[k] < n && k == n - 1 && c[x[k]][0] == 1) {/*找到一条哈密尔顿回路*/// (3)
        for (k = 0; k < n; k++) {
            printf("%d--", x[k]);/*输出哈密尔顿回路*/
```

```
            }
            printf("%d\n", x[0]);
            return;
        }
        else if (x[k] < n&& k < n - 1) {/*设置当前顶点的访问标志，继续下一个顶点*/
            visited[x[k]] = 1;//(4)
            k = k + 1;
        }
        else { /*没有未被访问过的邻接顶点，回退到上一个顶点*/
            x[k] = 0;
            visited[x[k]] = 0;
            k = k - 1;//(5)
        }
    }
    }

    void main()
    {
        int c[MAX][MAX] = { { 0,1,1,1 },{ 1,0,1,0 },{ 1,1,0,1 },{ 1,0,1,0 } };
        int x[MAX];
        Hamilton(MAX,x,c);
    }
```

输出结果为：

```
0--1--2--3--0
```

更详细的解析请参见比特培训(http://www.bitpx.com)下午试题四分析与解答视频。