软件设计师

案例分析专题 试题4

C语言算法

讲师：诸葛老师

1 考点分析

- 根据考试大纲，C语言算法已经成为案例分析的固定试题四，占15分，主要考查常见数据结构和算法的C语言实现。
- **1. C语言**

  熟练掌握C语言基础语法即可。
- **2. 数据结构相关概念**

  熟练掌握数据结构相关概念，如线性表、栈、队列、树、图、查找、排序等。
- **3. 常见的算法及复杂度**

熟练掌握常见的算法并能以C语言的形式来应用这些算法。常见的有迭代、穷举、递推、递归、回溯、贪心、动态规划和分治等，以及这些算法的复杂度。

2 C语言算法

- **1. C语言**

  掌握C语言基本语法。

- **2. 数据结构**

  见第3章《数据结构》，灵活掌握数据结构中的相关算法。

- **3. 算法设计与分析**

  见第8章《算法设计与分析》，尤其要掌握分治法、动态规划法、贪心法和回溯法的基本思想及典型实例。

- **4. 算法复杂度**

  见第8章《算法设计与分析》。

3 解题技巧

- C语言算法主要难在C代码填空上，建议是先不解决代码填空题（因为最难），先解决其他外围问题（如时间复杂度、算法技巧、取特殊值计算结果），最后解决代码填空，有助于理解整个题目，技巧如下：
- **1. 代码填空**

第一问，最后解决，并不影响解决其他题目，要理解题目算法原理，才能得出答案。结合算法描述中的公式，以及算法代码中类似的分支，能够发现填空的答案在代码中其它地方已经给出。

要注意的是，当遇到有最小值或最大值参与比较时，若比较出来比最小值更小，接下来肯定要更新这个最小值以及其下标元素值。当遇到一些条件判断的填空时，要注意对应上下文查看哪些变量是作为控制的。

3 解题技巧

- **2. 算法设计策略和时间复杂度**

第二问，先做，考查采用哪一种算法设计策略很好分辨，涉及到分组就是分治法，局部最优就是贪心法，整体规划最优就是动态规划法，迷宫类的问题是回溯法，记住关键字很好区分；时间复杂度就是看C代码中的for循环层数和每一层的循环次数的量级（关于n的量级），涉及到二分必然有O(logn)。

- **3. 特殊值计算**

第三问，一般应该先做，不需要根据C代码，直接根据题目给出的算法原理，一步步推导即可得出答案，耐心推导并不难。但要注意，如果遇到算法原理十分复杂的，建议放弃，掌握问题1和问题2的技巧即可。

本节练习-C语言算法

- 阅读下列说明和代码，回答问题1至问题3，将解答写在答题纸的对应栏内。

【说明】

排序是将一组无序的数据元素调整为非递减顺序的数据序列的过程，堆排序是一种常用的排序算法。用顺序存储结构存储堆中元素。非递减堆排序的步骤是：

（1）将含n个元素的待排序数列构造成一个初始大顶堆，存储在数组R(R[1]，R[2],…,R[n])中。此时堆的规模为n，堆顶元素R[1]就是序列中最大的元素，R[n]是堆中最后一个元素；

（2）将堆顶元素和堆中最后一个元素交换，最后一个元素脱离堆结构，堆的规模减1，将堆中剩余的元素调整成大顶堆；

（3）重复步骤(2)，直到只剩下最后一个元素在堆结构中，此时数组R是一个非递减的数据序列。

本节练习-C语言算法

- 【C代码】

  下面是该算法的语言实现。

  （1）主要变量说明

  n：待排序的数组长度

  R[]：待排序数组，n个数放在R[1]，$R[2], \cdots, R[n]$中

  （2）代码

```
#include<stdio.h>
#define MAXITEM 100
/*
调整堆
R：待排序数组；
V：结点编号，以v为根的二叉树，R[v] ≥ R[2v], R[v] ≥ R[2v + 1]，且其左子树和右子树都是大顶堆；
n：堆结构的规模，即堆中的元素数
*/
```

本节练习-C语言算法

```
void Heapify(int R[MAXITEM],int v,int n){
    int i,j;
    i=v;
    j=2*i;
    R[0]=R[i];
    while(j<=n){
        if(j<n&&R[j]<R[j+1]){
            j++;
        }
        if(____(1)____){
            R[i]=R[j];
            i=j;
            j=2*i;
        }
        else{
            j=n+1;
        }
    }
    R[i]=R[0];
}
```

```
/*堆排序，R为待排序数组；n为数组大小*/
void HeapSort(int R[MAXITEM],int n){
        int i;
        for(i=n/2;i>=1;i--){
            (____(2)____);
        }

        for(i=n;____(3)____;i--){
            R[0]=R[i];
            R[i]=R[1];
            (____(4)____);
            Heapify(R,1,i-1);
        }
}
```

本节练习-C语言算法

- 【问题1】（8分）

根据以上说明和代码，填充代码中的空（1）~（4）。

- 【问题2】（2分）

根据以上说明和代码，算法的时间复杂度为<u>（5）</u>（用符号O表示）。

- 【问题3】（5分）

考虑数据序列R=(7，10，13，15，4，20，19，8)，n=8，则构建的初始大顶堆为<u>（6）</u>，第一个元素脱离堆结构，对剩余元素再调整成大顶堆后的数组R为<u>（7）</u>。

本节练习-C语言算法

- 【参考答案】
- 【问题1】

（1）R[0]<R[i]

（2）Heapify(R,i,n)

（3）i>1

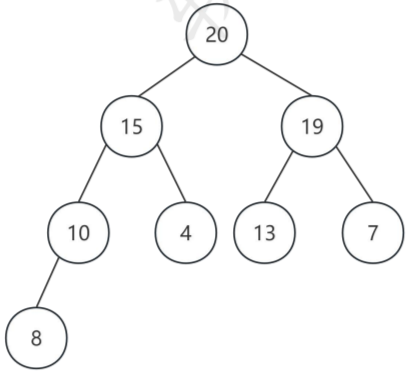
（4）R[1]=R[0]

- 【问题2】

（5）$O(n\log_2 n)$

- 【问题3】

（6）R=(20,15,19,10,4,13,7,8)



（7）R=(19,15,13,10,4,8,7,20)

本节练习-C语言算法

- 阅读下列说明和代码，回答问题1至问题3，将解答填入答题纸的对应栏内。

【说明】

某工程计算中经常要完成多个矩阵相乘（链乘）的计算任务，对矩阵相乘进行以下说明。

（1）两个矩阵相乘要求第一个矩阵的列数等于第二个矩阵的行数，计算量主要由进行乘法运算的次数决定，假设采用标准的矩阵相乘算法，计算$A_{m\times n} * B_{n\times p}$需要$m \times n \times p$次乘法运算，即时间复杂度为$O(m*n*p)$。

（2）矩阵相乘满足结合律，多个矩阵相乘时不同的计算顺序会产生不同的计算量。以矩阵$A1_{5\times 100}$，$A2_{100\times 8}$，$A3_{8\times 50}$三个矩阵相乘为例，若按$(A1*A2)*A3$计算，则需要进行$5*100*8+5*8*50=6000$次乘法运算，若按$A1*(A2*A3)$计算，则需要进行$100*8*50+5*100*50=65000$次乘法运算。

矩阵链乘问题可描述为：给定n个矩阵，对较大的n，可能计算的顺序数量非常庞大，用蛮力法确定计算顺序是不实际的。经过对问题进行分析，发现矩阵链乘问题具有最优子结构，即若$A_1*A_2*\cdots*A_n$的一个最优计算顺序从第k个矩阵处断开，即分为$A_1*A_2*\cdots*A_k$和$A_{k+1}*A_{k+2}*\cdots*A_n$两个子问题，则该最优解应该包含$A_1*A_2*\cdots*A_k$的一个最优计算顺序和$A_{k+1}*A_{k+2}*\cdots*A_n$的一个最优计算顺序。据此构造递归式，

$$cost[i][j]=\begin{cases}0 & if\ i=j\\ min_{i\le k<j}cost[i][k]+cost[k+1][j]+p_i*p_{k+1}*p_{j+1} & if\ i<j\end{cases}$$

其中，$cost[i][j]$表示$A_{i+1}*A_{i+2}*\cdots*A_{j+1}$的最优计算的代价。最终需要求解$cost[0][n-1]$。

本节练习-C语言算法

- 【代码】

　　算法实现采用自底向上的计算过程。首先计算两个矩阵相乘的计算量，然后依次计算3个矩阵、4个矩阵、...、n个矩阵相乘的最小计算量及最优计算顺序。下面是该算法的C语言实现。

　　（1）主要变量说明

　　n：矩阵数

　　seq[]：矩阵维数序列

　　cost[i][j]：二维数组，长度为n*n，其中元素cost[i][j]表示$A_{i+1} * A_{i+2} * \cdots * A_{j+1}$的最优计算的计算代价。

　　trace[][]：二维数组，长度为n*n，其中元素trace[i][j]表示$A_{i+1} * A_{i+2} * \cdots * A_{j+1}$的最优计算对应的划分位置，即k。

本节练习-C语言算法

（2）函数cmm

```
#define N 100
int cost[N][N];
int trace[N][N];
int cmm(int n,int seq[]){
    int tempCost;
    int tempTrace;
    int i,j,k,p;
    int temp;
    for(i = 0; i < n ; i++ ){
        cost[i][i] = 0;
    }
    for(p = 1; p < n ; p++){
        for(i = 0; i < n-p ; i++){
            ___(1)___;
            tempCost = -1;
            for(k = i ;___(2)___; k++){
                temp = ___(3)___;
                if(tempCost == -1 || tempCost > temp){
                    tempCost = temp;
                    tempTrace = k;
                }
            }
            cost[i][j] = tempCost;
            ___(4)___;
        }
    }
    return cost[0][n-1];
}
```

本节练习-C语言算法

- 【问题1】（8分）

  根据以上说明和代码，填充代码中的空（1）～（4）。

- 【问题2】（4分）

  根据以上说明和代码，该问题采用了（5）算法设计策略，时间复杂度为（6）（用符号O表示）。

- 【问题3】（3分）

  考虑实例n=4，各个矩阵的维数：A1为15*5，A2为5*10，A3为10*20，A4为20*25，即维数序列为15，5，10，20和25。则根据上述代码得到的一个最优计算顺序为（7）（用加括号方式表示计算顺序），所需要的乘法运算次数为（8）。

本节练习-C语言算法

- 【参考答案】
- 【问题1】

1. j=i+p
2. k<j
3. cost[i][k]+cost[k+1][j]+sep[i]*sep[k+1]*sep[j+1]
4. trace[i][j] = tempTrace

- 【问题2】

5. 动态规划
6. $O(n^3)$

- 【问题3】

7. $A1 * ((A2 * A3) * A4)$
8. 5375

感谢

THANKS

软件设计师QQ群：759713504　　诸葛老师微信：zhugeruankao