4.数据结构与算法应用解析

2021 上

【问题1】（8分）
(1)i<=N或其等价表示形式
(2)j=i+r-1或其等价表示形式
(3)temp<m[i][j] 或其等价表示形式
(4)s[i][j]+1,j或其等价表示形式
【问题2】（7分）
(5)动态规划法
(6)O(n^3)
(7)O(n^2)

本题考查的是凸多边形最优三角剖分动态规划设计过程。
本题算法难度较大，在没有理解算法过程的前提下，首先可以根据相关信息进行部分填空。
首先根据题干描述出现的将问题规模从k开始截断，此时其实就是"最优子结构"的说法，并且本题出现了递归式的应用，是典型的动态规划法的应用。
又根据题目中的代码，出现了三层嵌套for循环，此时代码的时间复杂度为O(n^3)。
本题用到的辅助空间记录中间解有2个数组m[i][j]和S[i][j]，都是二维数组，空间复杂度的量级为O(n^2)。
最后分析代码填空部分。
第（1）空，r表示的是子问题规模，规模划分已知从r=2开始，子问题最大应该能够取到N，因此本空填写r<=N或其等价表示形式。
第（2）空缺失的是j的初始化赋值，本空较难。代码计算前边界为i，链长为r的链的后边界取值，结果为i+r-1，即本题填写j=i+r-1或其等价表示形式。
第（3）空缺失判断条件，此时注释明确说明此处判断最小值，判断后，m[i][j]值进行修改并修改为temp，也就是意味着m[i][j]此时记录的不是最优解（最小值），需要进行修正改为最小，即填写temp<m[i][j] 或其等价表示形式（某一个数值比最小值还小，则修改最小值）。
第（4）空缺失的是打印参数，结合代码上下文进行分析，上文打印print_triangle(i,S[i][j]);即截断的前一部分编号，下面print_ triangle((4));打印的应该是截断的后一部分，即填写s[i][j]+1,j。

2021 下

问题1：
（1） d[0][j]=j
(2)str1[i-1]==str2[j-1]
(3)d[i-1][j-1] +1
(4) d[len1][len2]
问题2：
（5）动态规划法　（6）O(mn)
问题3：
（7）4

问题1：
（1） d[0][j]=j
(2)str1[i-1]==str2[j-1]
(3)d[i-1][j-1] +1
(4) d[len1][len2]
问题2：
（5）动态规划法　（6）O(mn)
问题3：
（7）4

2020

【问题1】（8分）
（1）k=k/2
（2）k>1
（3）data[k]<data[k-dk]
（4）data[j+dk]=t
【问题2】（4分）
（5）小于
（6）否
【问题3】（3分）
（7）(4，9，-1，8，20，7，15)

问题1:
希尔排序是一种经典的高效插入类排序算法。不稳定的排序算法，将每个步长划分为多个不连续的子序列，对每个子序列再次采用直接插入排序算法。
如对某数组A=(a1,a2,a3...a10),在某趟排序时，若delta=3,则将A分成三个子序列，A1=(a1,a4,a7,a10),A2=(a2,a5,a8),A3=(a3,a6,a9),然后分别在原位置上对A1、A2和A3进行直接插入排序处理。最后一趟排序中，delta=1,这样可以确保输出序列是有序的。delta 序列是希尔排序算法在具体实现的过程中定义的，本题在题干中已经给出，delta1 = n / 2 , 后面的每个delta是前面的1/2，最后一个deltak=1。根据题干，很容易得到空(1)为k=k/2,空(2)填k>1。
接下来的代码段是根据delta 值进行每一趟的排序，每趟排序是对不连续的每个子序列进行插入排序，因此，空(3)填data[k]<data[k-dk],即判断是否需要进行插入，空(4)填data[j+dk]=t,即确定了待插入的元素的位置。

问题2:
希尔排序算法是一种不稳定的排序算法，时间复杂度约在O（n^1.3)。

问题3:
对于数组（15、9、7、8、20、-1、4）用希尔排序方法进行排序，n=7，根据题干说明delta=n/2=3，A1（15，8，4），A2（9，20），A3（7，-1），每个子序列排序后得到A1（4，8，15），A2（9，20），A3（-1，7），还原得到(4，9，-1，8，20，7，15)

2017 上

答案

【问题1】

（1）first+(last-first)/2+1 或(first+last)/2+1

（2）firstSum<lastSum

（3）first+(last-first)/2 或(first+last)/2

【问题2】

（4）分治法

（5）O(lgn)

【问题3】

（6）2　（7）4

4.数据结构与算法应用解析

试题分析

【问题1】

对于本题代码填空，可以根据算法过程推导。

第一空，缺少循环的停止条件，根据题干描述，在左侧比较应该是到(last+first) / 2为止，由于这里是小于符号，所以第一空填写(last+first) / 2+1，或first + (last-first) / 2 + 1，或其他等价形式。

第二空，缺少判断条件，进入较小部分继续比较，因此本空应该填写firstSum<lastSum。

第三空，缺少返回值，此时既不在左侧，也不在右侧，则当前位置即为目标位置，返回当前位置first+(last-first)/2。

【问题2】

本题采用的是分治法策略。整个算法过程类似于树形结构，所以时间复杂度为O(lgn)。

【问题3】

若输入30个硬币，找假硬币的比较过程为：

第1次：15 比 15，此时能发现假币在15个的范围内。

第2次：7比7，此时，如果天平两端重量相同，则中间的硬币为假币，此时可找到假币，这是最理想的状态。

第3次：3比3，此时若平衡，则能找出假币，不平衡，则能确定假币为3个中的1个。

第4次：1比1，到这一步无论是否平衡都能找出假币，此时为最多比较次数。

因此最少比较2次，最多比较4次。

2014 上

答案

【问题1】（8分）

（1）k<=r

（2）arr[k]=right[j]

（3）begin<end

（4）mergeSort(arr,mid+1,end)

【问题2】（5分）

（5）分治

（6）T(n)=2T(n/2)+n

（7）O(nlgn)

（8）O(n)

【问题3】（2分）

（9）n1+n2

4.数据结构与算法应用解析

试题分析

根据题目中的参数说明，void merge(int arr[],int p,int q,int r)是将数组arr[p...q]和数组arr[q+1...r]进行合并成一个排序的数组，因此合并之后数组的长度为r-q+1>0，k=q，也就是k<=r或k<r+1；数组arr存入子数组arr[p...q]、arr[q+1...r]当前进行比较的最小值，因此当left[i]> right[j]时，数组arr中存入right[i]，即arr[k]=right[j]；

void mergeSort(int arr[],int begin,int end)是指将数组arr递归进行划分，直到分成多个由一个元素组成的子数组时，停止划分，此时也就是begin==end，因此（3）处为begin<end，也就是只要begin!=end则继续划分。划分的时候每次分成两半，两半再分别递归，因此mergeSort(arr,begin,mid);mergeSort(arr,mid+1,end);

很明显归并排序使用的分治算法，每次将数组分割成两个小的子数组。

假设对n个元素的数组进行归并排序时间复杂度为T(n)，则分成两个小的子数组后分别进行排序所需的时间为T(n/2)，两个子数组则时间复杂度为2T(n/2)，再加上归并的时间n，即可得出答案归并排序的时间复杂度为O(nlgn)，因为在归并过程中，需要借助left和right两个数组辅助，因此空间复杂度为O(n)。

2019 上

答案

【问题1】

（1）queen[i]==queen[j]或其等价形式

（2）1

（3）Place(j)或其等价形式

（4）Nqueen(j+1)

【问题2】

（5）回溯法

【问题3】

（6）2个

（7）（2413）或（2,4,1,3）

（3142）或（3,1,4,2）

4.数据结构与算法应用解析

试题分析

【问题1】

（1）第一空根据代码上下文：

```
for(i=1;i<j;i++){   /* 检查与已摆放的皇后是否在同一列或者同一斜线上 */
        if(    (1)    ) || abs(queen[i]-queen[j]) == (j-i)) {
            return 0;
        }
    }
```

abs(queen[i]-queen[j]) == (j-i) 判断是否在同一斜线上，此处还缺少对同一列的判断，即 queen[i]==queen[j]或其等价形式。

（2）第二空根据 Place(int j)函数首行注释：

```
int Place(int j){     /* 检查当前列能否放置皇后，不能放返回0，能放返回1 */
```

此处是成功后的返回，返回值应该是1。

（3）第三空根据代码上下文

```
if(       (3)      ){
            if(j == n) {    /* 如果所有皇后都摆放好，则输出当前摆放方案 */
                Show();
            } else {        /* 否则继续摆放下一个皇后 */
                    (4)     ;
            }
        }
```

（3）与j==n结合可以判断所有皇后都摆好，（3）与j!=n结合可以判断继续摆放下一个皇后，即前面的皇后已摆放好。

所以（3）的判断条件应该是摆放函数Place()返回值为1，即（3）Place(j)或其等价形式。

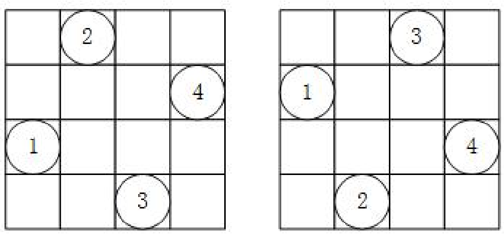
（4）第四空填写摆放下一个皇后，即（4）Nqueen(j+1)。

【问题2】

根据题干描述"如果该行没有合适的位置，回溯到上一个皇后，考虑在原来位置的下一个位置上继续尝试摆放皇后"，本题采用的是回溯法的设计策略。

【问题3】

当n=4时，可以有2种摆放方式，如下所示：



即（2413）（3142）。

2017 下

答案

【问题1】

1、visited[0] = 1

2、visited[x[k]] == 0

3、c[x[k]] [0]== 1

4、visited[x[k]] = 1

5、k = k - 1

【问题2】

6、回溯法

7、深度优先

试题分析

哈密顿图是一个无向图，由天文学家哈密顿提出，由指定的起点前往指定的终点，途中经过所有其他节点且只经过一次。在图论中是指含有哈密顿回路的图，闭合的哈密顿路径称作哈密顿回路，含有图中所有顶点的路径称作哈密顿路径。

回溯法是一种选优搜索法，又称为试探法，按选优条件向前搜索，以达到目标。但当探索到某一步时，发现原先选择并不优或达不到目标，就退回一步重新选择，这种走不通就退回再走的技术为回溯法，而满足回溯条件的某个状态的点称为"回溯点"。在包含问题的所有解的解空间树中，按照深度优先搜索的策略，从根结点出发深度探索解空间树。当探索到某一结点时，要先判断该结点是否包含问题的解，如果包含，就从该结点出发继续探索下去，如果该结点不包含问题的解，则逐层向其祖先结点回溯（其实回溯法就是对隐式图的深度优先搜索算法）。 若用回溯法求问题的所有解时，要回溯到根，且根结点的所有可行的子树都要已被搜索遍才结束。 而若使用回溯法求任一个解时，只要搜索到问题的一个解就可以结束。

算法题历来都被认为是比较难的题，一个程序开发人员都不喜欢看别人的代码。但是要得分也不是太难。

问题2比较容易得分，而且第二空就是个二选一的填空。只要了解到回溯法的相关原理，基本可以得满分。对于问题1就需要要花一些心思，去读懂题干和代码，但是这里的第1空和第5空也是比较容易发挖出来的空。第一空是初始化第一个结点，第五空是此路不通，得回走，所以得退回。

2015 上

答案

【问题1】

（1）pos[i] ==pos[k]

（2）j=1

（3）isplace(pos,j)==0

（4）j<N

（5）j=j-1

【问题2】

（6）回溯法

【问题3】

（7）

方案1：2 4 1 3

方案2：3 1 4 2

4.数据结构与算法应用解析

试题分析

本题考查算法设计和 C 程序设计语言的相关知识。

此类题目要求考生认真阅读题目，理解算法思想，并思考将算法思想转化为具体的程序设计语言的代码。

【问题1】

根据题干描述。空（1）所在的代码行判断皇后合法放置的约束条件，即不在同一行，这通过把第 i 个皇后放在第 i 行实现，条件 "fabs(i-k) = fabs(pos[i]-pos[k])" 判断的是当前摆放的皇后是否与之前摆放的皇后在同一对角线上。因此，空（1）判断的是当前摆放的皇后是否和之前摆放的皇后在同一列上，即应填入"pos[i]==pos[k]" 。

根据算法思想和主函数上下文，空（2）处应该考虑第 1 个皇后，即初始化 j为1，空（2）填写"j=1"。空（3）所在的行是判断放置第j 个皇后的位置是否合适，"pos[j]<= N" 表示在该行的合法列上，但还需要进一步判断是否与前面的皇后有冲突，根据满足条件后的语句，尝试放入下一列，因此空（3）处填入"!isplace(pos，j) 。根据前面的注释，空（4）所在的行是考虑下一个皇后，其条件是，当前皇后找到了合适的位置，而且还存在下一个皇后，因此空（4）处应填入"j<N" 。根据下面的注释，若当前皇后没有找到合适的位置，则应回溯，即再次考虑上一个皇后的位置，因此空（5）处填入 "j=j-1"。

【问题2】

从上述题干的叙述和 C 代码很容易看出，从第一个皇后开始，对每个皇后总是从第一个位置开始尝试，找到可以放置的合法位置；若某个皇后在对应的行上没有合法位置，则回溯到上一个皇后，尝试将上一个皇后放置另外的位置。这是典型的深度优先的系统搜索方式，即回溯法的思想。

【问题3】

四皇后问题的答案为：

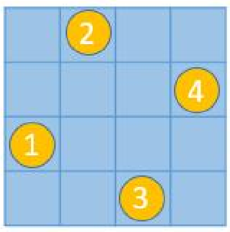
方案 1： 2 4 1 3

方案 2： 3 1 4 2

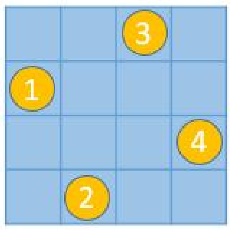
如表 4-1 所示：

表4-1

方案1



方案2



4.数据结构与算法应用解析

2019 下

答案

【问题1】

（1）c[i][j]

（2）i>0&&j>=w[i]

（3）Calculate_Max_Value(v, w, i-1, j-w[i]) + v[i]

（4）c[i][j]=temp

【问题2】

（5）动态规划

（6）自顶向下

【问题3】

（7）40

试题分析

本题考查的是动态规划法，将中间结果存在二维数组c[][]当中。

【问题1】

结合题干描述中的递归式，函数最终返回值应该是c[i][T]，又根据代码上下文，T在函数中传参为j，所以第（1）空返回的结果为c[i][j]。

根据递归式和代码上下文可知，c[i][j]=0已处理，if后面处理的是max递归部分，又根据上下文可以看到temp会与c[i][j]进行比较，递归时i=i-1，所以这里判断的是temp与c[i-1][T]的值，第（3）空的处理是将c[i-1][j-w[i]]+v[i]传递给temp，即temp= Calculate_Max_Value(v, w,i-1,j-w[i])+v[i]，注意这里不是直接穿c[][]数组值，而是递归调用Calculate_Max_Value()函数。那么第（2）空缺失的判断条件根据第3个表达式条件为i>0且T>=w[i]，对应代码中的参数即i>0&&j>=w[i]。

第（4）空是c[i][j]<temp比较后的返回值，根据题干，我们需要返回max最大值，最终返回的结果是c[i][j]，因此要保证c[i][j]最大，当c[i][j]<temp时，将temp赋值给c[i][j]，即c[i][j]=temp。

【问题2】

本题采用递归形式，并且求取的是全局最优解，中间结果存在二维数组c[][]当中，所以采用的是动态规划法。动态规划法采用递归形式，i会从N-1递减至0，所以是自顶向下的方式。

【问题3】

本题考查最优解，可以忽略题干描述，直接凑数，可得第3和第4个物品，重量为11，价值为40，此时为最优解，即为最大价值。

4.数据结构与算法应用解析

2018 下

答案

【问题1】

（1）max=C[i][j-1]

（2）t=i

（3）isMatch(B[t],B[j])，或isMatch(B[t],B[j])==1，或与其等价的形式

（4）C[1][n]

【问题2】

采用的算法策略：动态规划法

时间复杂度：$O(n^3)$

【问题3】

最大字符对数：2

4.数据结构与算法应用解析

试题分析

本题考查的是用动态规划法，以非递归方式实现。

根据题干，配对要求：

（1）满足四种组合之一；

（2）配对的2个字符间距至少有4个字符；

（3）若字符已配对，则其他配对不再考虑，也就是说1个字符不能配对2次，比如ACCCCUCCCCA，只有1组配对AU，U不能再与后面的A形成第2组配对；

（4）不交叉，2组配对字符位置能交叉，比如ACCCCCUUUUG，只有1组配对AU，CG与AU有交叉不能形成配对。

【问题1】

对于问题1代码填空，主要根据题干描述和代码上下文进行推导。

根据代码上下文可知，在整段代码中，缺少对变量max和t赋初值，这两个初值的赋值，应该填在空（1）和空（2）中，一般t作为循环变量，在for中进行赋值。

代码中有三层嵌套for循环。

其中第一层for循环，变量为k，取值范围从5到n-1，从题干描述，我们可以看到对于整个比较过程，要求字符对的位置相差大于4，因此此处的k值是字符对下标的差值；

第二层for循环，变量为i，取值范围从1到n-k，从题干描述，我们可以得出i是字符对较小的下标；

第三层for循环，变量为t，取值范围需要赋初值，并且t<=j-4（此处有异议，与题干描述中的>4有不符，但不影响本题解题过程），从题干描述和递归式可以看到，t是中间字符下标，用来划分子问题的，并且从递归式我们可以得出，t的最小值应该从i开始，因此空（2）为t=i；

在第二层for循环内部，有j=i+k，根据代码和题干描述，可以得出j是字符对较大的下标，根据i和k的取值，可以看到j的取值范围为从6到n-1，对于空（1）作为max的初始赋值，又根据递归式，可以看到max应该在C[i][j-1]和C[i][t-1]+1+C[t+1][j-1]之间取最大值，在代码中可以看到if会判断max与C[i][t-1]+1+C[t+1][j-1]之间的大小，因此，max之前的赋值应该为C[i][j-1]，才能对二者进行比较，也就是说空（1）应该为max=C[i][j-1]。

空（3）在if判断中作为判断条件，根据递归式的条件和代码上下文，此处缺少字符匹配的判断，题干描述字符下标从1开始，因此，在比较过程中，实际比较的应该为B[t]和B[j]位置的字符，空（3）应该填写isMatch(B[t],B[j])，或isMatch(B[t],B[j])==1，或与其等价的形式。

空（4）作为整个函数的返回值，因此空（4）应该为C[1][n]为最终结果。

【问题2】

本题采取的是动态规划的策略，代码为三层嵌套循环时间复杂度为k*i*t，由于k的取值范围是6~n-1，i的取值范围是1~n-5，t的取值范围是1~n-5，都是与n的取值相关，因此本题的时间复杂度为O(n3)。

【问题3】

对于本题最大字符匹配对数，根据题干描述或代码推导，可以看到，字符序列ACCGGUAGU的最大匹配情况为，(b1,b6)，(b1,b9)或(b2,b8)，(b3,b8)，这两种情况的最大匹配对数都为2，因此本题答案（7）空为2。

4.数据结构与算法应用解析

2018 上

答案

【问题1】

（1）：i<=n

（2）：i<=j

（3）：(temp>=p[i]+r[j-i])?temp:(p[i] + r[j - i])

（4）：r[j] = temp

或

（3）：(temp>=r[i]+r[j-i])?temp:(r[i] + r[j - i])

（4）：r[j] = (temp>p[j])?temp:p[j];

【问题2】

（5） 动态规划法

（6）$O(2^n)$

（7）$O(n^2)$

试题分析

【问题1】

在自顶向下实现过程中，n-i表示规模从大到小即n-1~0，即对应i的初始值为1，结束值为n，第一空填写i<=n，递归式也有范围提示可以参照。

在自底向上实现过程中，采用双重嵌套循环，内层循环从1~j，第二空填写i<=j。

第三空和第四空比较复杂，是具体的实现过程，是本题的难点。

根据题干内容，本题考查的是钢条切割问题最优化问题，求解的思路即先考虑最左侧的切割考虑，再依次向右扩展，中间的最优解结果记录在数组r[]中，并用temp中间变量传递最大值。

根据递归式$r_n = \max_{1\le i\le n}(p_i + r_{n-i})$，即r[]最终结果是该过程的最大值，（3）空给temp赋值，那么（4）空应该是将这个中间值传给最终的rn，也就是代码中的r[j]，即第四空填写r[j]=temp，那么此时第三空对应最大值的求取，也就是本算法的核心，这里的最大值是在1~j的规模范围循环比较，用temp放置本轮结果，再与下一轮结果进行比较，第三空temp=(temp>=p[i]+r[j-i])?(temp:(p[i] + r[j - i])。

【问题2】

题干中提到说考虑所有可能的i，得到最大收益的方式，而自底向上算法实现时，使用到数组把其中最优的解记录，并用r[]记录中间解，因此本题算法策略是动态规划法。

动态规划法自顶向下时需要对规模n进行求取，此时需要递归至规模1并最终返回结果规模n的解并记录，规模n-1同样如此，时间复杂度较大，可以达到$O(2^n)$;

动态规划法自底向上时先求取规模1的解并记录，然后查询规模1的解从而求解规模2的解，以此类推，直至求取至规模n，有查询和循环求解2层嵌套循环，时间复杂度为$O(n^2)$。

4.数据结构与算法应用解析

2016 上

答案

【问题1】

（1）size[1][j]=1

（2）size[i][j]=size[i-1][j]

（3）net[m++]=i;

【问题2】

（4）动态规划算法

（5）$O(n^2)$

（6）O(n)

【问题3】

（7）4

（8）（3，π（3））,（5，π（5））,（7，π（7））,（9，π（9））

或：（3，4）,（5，5）,（7，9）,（9，10）

4.数据结构与算法应用解析

试题分析

本题是算法设计题，涉及的算法策略是动态规划法。

【问题1】

本题要求补充代码，主要参照代码注释、题干的算法思路和递归式即可得到。

对于第（1）空，有注释“当j>=π(1)时”，此时属于i=1的其他情况，找到递归式的条件，所以（1）空填写size[1][j]=1；

对于第（2）空，有注释“当j<π(i)时”，此时属于i>1时，j<π(i)的条件，找到递归式对应条件，所以（2）空填写size[i][j]=size[i-1][j]；

对于第（3）空，有注释“将i记录到数组net中，连接线数自增1”，将i记录到net数组，即net[]=i，其中net位置应该时连接线数m，此时为m++，因此（3）空填写net[m++]=i。本空也可以根据后面的代码推导。

【问题2】

1、根据题干描述“经分析，该问题具有最优子结构性质。对规模为n的电路布线问题，可以构造如下递归式”，根据最优子结构可判断本题使用的是动态规划法的算法策略。

2、根据代码，可以看到maxNum函数有两层嵌套循环，因此时间复杂度为O(n2)。

3、根据代码，可以看到constructSet函数只有一层循环结构，因此事件复杂度为O(n)。

【问题3】

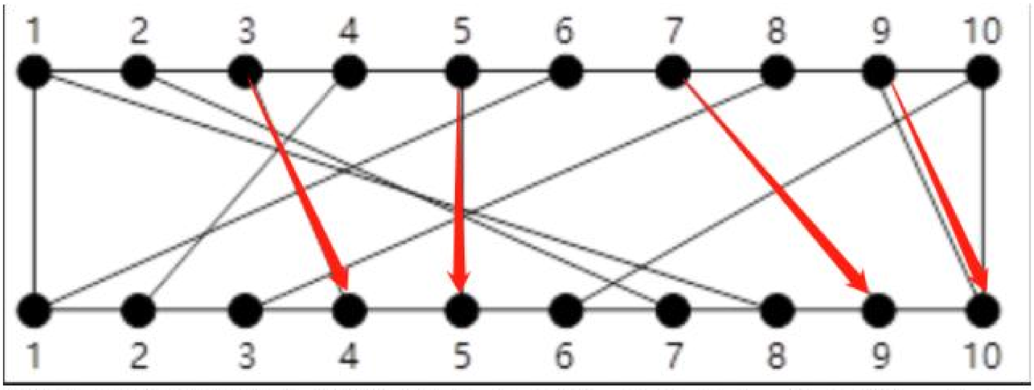
这个是动态规划问题，不相交的平行线。

设a[i][j]为上端接线柱i与下端接线柱j前的最大不相交子集，则：

若i与j不相连，则i与j前的最大不想交子集等于i与j - 1前或i - 1与j前的最大不相交子集的最大值，即a[i][j] = max(a[i][j - 1], a[i - 1][j])

若i与j相连，则i与j前的最大不想交子集等于i - 1与j - 1前的最大不想交子集加1，即a[i][j] = a[i - 1][j - 1] + 1

题目的意思就是要求出，没有交叉的这种连线的数量达到最大的情况。此时，有4条这样的线不会交叉，所以是大不相交子集连接数为4。如果你能找到5条这样不交叉的线，则是5。就这个意思。



由此可得，最大不相交连接数为4，包含的连接线为：（3，π（3），（5，π（5））,（7，π（7））,（9，π（9））

2015 下

答案

【问题1】

（1）x[i-1] == y[j-1]

（2）max=c[i][j]

（3）c[i][j]=0

（4）i=maxi-max

【问题2】

（5）动态规划法

（6）O(m*n)

【问题3】

（7）AB

试题分析

首先对于C语言算法题，一般的解题思路是先解决除程序填空以外的问题，这些问题弄清楚，有利于程序填空部分的分析。

第一步，分析程序所采用的算法，常见的算法包括：分治法、动态规划法、回溯法、贪心法。本题中要求的是两个串的最长公共子串，在程序中采用了数组来记录子问题的中间结果，这一特征与动态规划法的做法非常吻合，所以应选动态规划法。

第二步，解决"输入字符串x= "ABCADAB'，'y="BDCABA"，则输出为 （7） "的问题，该问题相对容易解决，因为题目已告知程序的作用是求最长公共子串，而且从程序的输出函数可以看出，要输出的，只有子串，没有其他信息，所以我们只需要手动求两个串的公共子串，并写出答案即可。两个串第一个公共子串明显是"AB"，所以输出的结果为"AB"。

第三步，求时间复杂度，由于程序中最多双重循环，其中外层循环的规模为m，而内层循环的规模为n，所以时间复杂度为O(m*n)。

第四步，也是最难的一步，是解决程序填空的问题。动态规划的问题，一般都会给出递归定义式，这个式子，往往是多个空的关键点。以本题为例：

第1空是判断的条件，输出的结果是c[i][j] = c[i -1][j -1] + 1，此时根据递归式可以看到，所需的条件是x[i-1] = y[j-1]，因此第1空填写判断条件x[i-1] == y[j-1]。

第2空是在 if(max<c[i][j])跟随的大括号中，根据前后代码和此处的判断条件，当max小于当前时，应该改变max的值，并记录此时的maxi和maxj，此处缺少改变max的值，因此，第2空应该将当前最大值赋值给max，即max=c[i][j]。

第3空是第一个if语句的else选择，根据递归式，if（x[i-1] == y[j-1]）不满足时，属于其他情况，应该赋值为0，因此第3空填写c[i][j]=0。

而第4空是用于打印结果，由于maxi记录了子串末尾+1的位置信息，子串长度为max，所以用maxi-max定位至子串开始位置，以便打印子串，因此第4空填写i=maxi-max。

2014 下

答案

【问题1】

（1）b[0]=1

（2）j<i

（3）a[j]<=a[i]

（4）b[i]=len+1

【问题2】

（5）动态规划法

（6）$O(n^2)$

【问题3】

b={1,2,2,3,3,4}

试题分析

本题考查算法设计与分析技术以及算法的C语言实现，是比较传统的题目，要求考生细心分析题目中所描述的内容。

（1）根据题中说明，b数组记录最长递增子序列的长，故应初始化b[0]=1，这是第一问的答案。两重for循环中，第一重是从a数组的第二个元素开始，考虑每个子数组a[0...i]的最长递增子序列的长度，第二重是具体的计算过程。考虑子数组a[0..i]，其最长递增子序列的长度应该等于子数组a[0..i-1]中的比元素a[i]小的元素的最长递增子序列的长度加1，当然，可能存在多个元素比元素a[i]小，那么存在多个最长递增子序列的长度，此时，取最大者。因此，空

（2）填写"j<i"，即考虑子数组a[0..i-1]第三问为a[j]<=a[i]，第四问为b[i]=len+1。

（2）算法将待求解问题分解成若干个子问题，先求解子问题，然后从这些子问题的解得到原问题的解。使用的是动态规划的思想。时间复杂度计算最坏情况下的运算次数，最坏情况时i和j都从1跑到n,故运算n的平方次。算法的时间复杂度为$O(n^2)$。

（3）初始b[0]=1，a[0]=3，a[1]=10进入时b[1]=2，a[2]=5进入时有3、5的序列故b[2]=2，a[3]=15进入时有3、10、15，故子序列为3，a[4]=6时有子序列3、5、6，故为3，当最后一个元素8进入时有3、5、6、8，故b[5]=4。所以b=[1,2,2,3,3,4]。

2016 下

答案

【问题1】

（1）：j<ls;

（2）：t[i]==s[j];

（3）：j=next[j];

（4）：i-ls+1 或其等价形式;

【问题2】

O(lt+ls)

【问题3】

（6）：[-1,-1,1,-1,-1,2,0,0]，（7）6。

试题分析

【问题1】

本题问题1根据KMP算法的伪代码描述进行推导。

根据伪代码中第2步可以推导（1）是判断字符串s是否还有字符，即j<ls。i表示字符串t的下标，j表示字符串s的下标。

根据伪代码第2.1步可以推导（2）是判断字符串t和字符串s当前位置的字符是否相同，即t[i]==s[j]。

根据伪代码第2.2步可以推导（3）是当第2.1步判断条件不满足时，改变j所指向的字符位置。即j=next[j]。

根据伪代码第3步可以推导（4）是返回匹配的起始位置。由于当前i所指向字符串中匹配子串的最后一个字符的位置，且已知子串的长度为ls。（4）的代码为i-ls+1或其等价形式。

【问题2】

本题问题2是计算KMP算法的复杂度。算法的复杂度一般考虑最坏情况，那么在子串读到ls及主串读到lt的时候是最坏情况。所以复杂度是O(lt+ls)

4.数据结构与算法应用解析

【问题3】

本题问题3中已知字符串"BBABBCAC"，则根据get_next()函数可以求得next数组的元素值为[-1,-1,1,-1,-1,2,0,0]。并计算得到起始位置为6。

代入字符串"BBABBCAC"到get_next函数。

```
void get_next( int *next, char *s, int ls) {
   int i=0, j=-1;
   next[0]=-1;/*初始化next[0]*/
   while(i < ls){/*还有字符*/
   if(j==-1I ls[i]==s[j]){/*匹配*/
      j++;
      i++;
   if( s[i]==s[j])
      next[i] = next[j];
   else
      Next[i] = j;
  }
else
  j = next[j];
 }
}
```

> 这里涉及的只是代码的代入分析过程，注意循环的处理即可。
>
> 下面将循环过程依次代入数值并且写作顺序处理过程如下：
>
> 传参：s[]={B,B,A,B,B,C,A,C}，ls=8，next[]数组只声明未取值。
>
> 初始化：i=0,j=-1,next[0]=-1。
>
> while(i<ls)执行后面的循环体，即当i<8时执行循环。
>
> （1）当i=0,j=-1时：
>
> 判断if(j==-1||s[0]==s[-1])，满足条件1执行下一步：i++=1,j++=0。
>
> 判断if(s[1]==s[0])，满足条件执行下一步next[1]=next[0]=-1。
>
> 【此时i=1,j=0】
>
> （2）当i=1,j=0时：
>
> 判断if(j==-1||s[1]==s[0])，满足条件2执行下一步：i++=2,j++=1。
>
> 判断if(s[2]==s[1])，不满足条件执行else下一步next[2]=j=1。
>
> 【此时i=2,j=1】
>
> （3）当i=2,j=1时：
>
> 判断if(j==-1||s[2]==s[1])，不满足条件1和2执行else下一步：j=next[1]=-1。
>
> 【此时i=2,j=-1】
>
> （4）当i=2,j=-1时：
>
> 判断if(j==-1||s[2]==s[-1])，满足条件1执行下一步：i++=3,j++=0。
>
> 判断if(s[3]==s[0])，满足条件执行下一步next[3]=next[0]=-1。

【此时i=3,j=0】

（5）当i=3,j=0时：

判断if(j==-1||s[3]==s[0])，满足条件2执行下一步：i++=4,j++=1。

判断if(s[4]==s[1])，满足条件执行下一步next[4]=next[1]=-1。

【此时i=4,j=1】

（6）当i=4,j=1时：

判断if(j==-1||s[4]==s[1])，满足条件2执行下一步：i++=5,j++=2。

判断if(s[5]==s[2])，不满足条件执行else下一步next[5]=j=2。

【此时i=5,j=2】

（7）当i=5,j=2时：

判断if(j==-1||s[5]==s[2])，不满足条件1和2执行else下一步：j=next[2]=1。

【此时i=5,j=1】

（8）当i=5,j=1时：

判断if(j==-1||s[5]==s[1])，不满足条件1和2执行else下一步：j=next[1]=-1。

【此时i=5,j=-1】

（9）当i=5,j=-1时：

判断if(j==-1||s[5]==s[-1])，满足条件1执行下一步：i++=6,j++=0。

判断if(s[6]==s[0])，不满足条件执行else下一步next[6]=j=0。

【此时i=6,j=0】

（10）当i=6,j=0时：

判断if(j==-1||s[6]==s[0])，不满足条件1和2执行else下一步：j=next[0]=-1。

【此时i=6,j=-1】

（11）当i=6,j=-1时：

判断if(j==-1||s[6]==s[-1])，满足条件1执行下一步：i++=7,j++=0。

判断if(s[7]==s[0])，不满足条件执行else下一步next[7]=j=0。

【此时i=7,j=0】

（12）当i=7,j=0时：

判断if(j==-1||s[7]==s[0])，不满足条件1和2执行else下一步：j=next[0]=-1。

【此时i=7,j=-1】

（13）当i=7,j=-1时：

判断if(j==-1||s[7]==s[0])，满足条件1执行下一步：i++=8，i=ls，退出while循环。

next[]数组下标从0到7，结果分别为：[-1，-1，1，-1，-1，2，0，0]