###### ****一、单选题****

1. **下列哪种函数可用于为避免无效搜索所采取的策略？ （ ABCD ）。**

**A、递归函数 B、随机函数 C、剪枝函数 D、搜索函数**

1. **为了避免空间树中的无效搜索，依据什么来剪枝？（ ABCD  ）**

**A、约束条件和定界函数 B、回溯法和分支限界法**

**C、深度优先和广度优先 D、随机函数和折半查找**

1. **折半查找算法和快速排序算法的时间复杂度分别是（ ABCD ）。**

**A、O(n)、O(2^n ) B、O(n)、O(n^2)**

**C、O(logn )、O(nlogn) D、O(logn )、O(n^2 )**

1. **下列能用贪心法得到最优解的问题是（ ABCD ）。**

**A、背包问题 B、任务调度问题 C、零钱兑换问题 D、0-1背包问题**

1. **假设P!=NP，下面关于NP问题说法正确的是（ ABCD ）。**

**A、NP问题都是不可能解决的问题 B、P类问题包含在NP类问题中**

**C、NP完全问题是P类问题的子集 D、NP类问题包含在P类问题中**

1. **动态规划法和贪心法的共同点是都要求问题具有（ ABCD  ）。**

**A、贪心选择性质 B、最优子结构性质 C、无后效性 D、重叠子问题性质**

1. **目前可以采用（ ABCD ）在多项式时间内求出旅行商问题的一个近似最优解。**

**A、回溯法 B、近似算法 C、贪心法 D、动态规划法**

1. **设问题输入规模为n，分析算法的最坏情况时间复杂度，下列说法错误的是（ABCD）。**

**A、子集树空间搜索的时间复杂度是O(2^n）**

**B、排列树空间搜索的时间复杂度是O(n!）**

**C、贪心算法是多项式时间**

**D、动态规划法是多项式时间**

1. ***T*(*n*)表示输入规模为*n*的算法效率，且*T*(1) = 1，以下算法中效率最优的是（ABCD）。**

**A．*T*(*n*)＝*T*(*n* – 1) + *n* B．*T*(*n*)＝*T*(*n*/2) + *n***

**C．*T*(*n*)＝*T*(*n* – 1) + 1 D．*T*(*n*)＝2*T*(*n*/2) + 1**

1. **常见的两种分枝限界法是（ ABCD ）。**

**A．广度优先分枝限界法和深度优先分枝限界法**

**B．队列式（FIFO）分枝限界法和堆栈式分枝限界法**

**C．排列树分枝限界法和子集树分枝限界法**

**D．队列式（FIFO）分枝限界法和优先队列式分枝限界法**

1. **用回溯法求解下列问题，其搜索空间为子集树的是（ABCD）。**

**A．0-1背包问题 B．批处理作业调度问题**

**C．旅行售货员问题 D．n皇后问题**

1. **设问题的输入规模为*n*，c为正数，分析算法的最坏情况时间复杂度。下列说法错误的是（ ABCD）。**

**A．子集树搜索时间复杂度是*O*(2*n*) B．排列树搜索时间复杂度是*O*(*n*!)**

**C．分治算法时间复杂度是*O*(log*n*) D．贪心算法时间复杂度是*O*(*nc*)**

1. **下列关于算法的说法错误的是（ ABCD ）。**

**A．算法有零个或多个输入。**

**B．算法至少有一个输出。**

**C．算法的每一步都无歧义且是可行的。**

**D．算法步骤是有限的，且运行时间是最短的。**

1. **设函数*f*(*n*)、*g*(*n*)满足关系*f*(*n*) = *O*(*g*(*n*))，则存在正常数*c*和正常数*n*0，使得对所有*n*≥*n*0，有（ ABCD ）。**

**A．*f*(*n*) ≤ *cg*(*n*) B．*f*(*n*) ≥ *cg*(*n*)**

**C．*g*(*n*) ≤ *cf*(*n*) D．*g*(*n*) ≥ *cf*(*n*)**

1. **下列问题中，（ ABCD ）是一个NP完全问题。**

**A．排序问题 B．旅行商问题 C．最短路径问题 D．最小生成树问题**

1. **下列关于算法的说法错误的是（ ABCD ）。**

**A．算法至少有一个输入.**

**B．算法至少有一个输出.**

**C．算法的每一步都是无歧义而且可行的.**

**D．算法步骤有限.**

1. **一般说能用递归解决的问题，除下面的（ ABCD  ）以外，其它都应该满足。**

**A．问题可以转化为一个或多个相同性质的子问题来求解.**

**B．每个子问题的数量规模是父问题的一半.**

**C．递归调用的次数是有限的.**

**D．有结束递归的条件来中终止递归.**

1. **设问题规模为*n*，用分治算法求解该问题所需要的时间记为*f*(*n*)，且满足递归方程：*f*(*n*0)=*d*，*f*(*n*)=*af*(*n*/*c*)+*g*(*n*) (*n*>*n*0,c≥2是整数). 则其中的*g*(*n*)表示（ABCD）。**

**A．分解所需要的时间 B．求解子问题的时间**

**C．合并子问题的解所需要的时间 D．分解与合并所需时间的总和**

1. **分支限界法在问题的解空间树中，按（ ABCD ）策略从根结点出发搜索解空间树。**

**A．深度优先 B. 活结点优先 C. 广度优先 D. 扩展结点优先**

###### ****二、判断题****

1. **如果函数f在它的定义中调用它自己，则称为递归，可以用栈、队列和循环来消除递归。（ √× ）**
2. **分治法的设计思想是将一个难以直接解决的大问题分割成规模较小的子问题，分别解决子问题，最后将子问题的解组合起来形成原问题的解，原问题和子问题规模相同但性质不同。（ √× ）**
3. **用回溯法解0/1背包问题时，该问题的解空间是排列树结构；用回溯法解流水作业调度问题时，该问题的解空间是子集树结构。（ √× ）**
4. **有多项式时间算法的问题是P类问题，没有多项式时间算法的问题是NP类问题。（√×）**
5. **对于输入规模为n的问题，一般来说，在最坏情况下，回溯法和分支限界法的时间复杂度都是*O*(2*n*).（ √× ）**
6. **贪心法是在每一步确定一个当前最优的选择，动态规划法是求出一个动态规划数组，所以都是多项式时间算法。（√× ）**
7. **如果所有的NP问题都能在多项式时间内归约到A问题，则A是NP困难问题。如果NP困难问题A是NP问题，则A是NPC问题。（ √×）**
8. **回溯法解题的一个显著特征是在搜索过程中动态产生问题的解空间，在任何时刻，算法只保存从根结点到当前扩展结点的路径。（√×）**
9. **在一个图中，如果两个顶点有路径相连，则它们在深度优先遍历树中一定是祖先后继关系。（√×）**
10. **一个有向图G存在拓扑排序，则图G一定没有环；而一个有向无环图却不一定存在拓扑排序。（√×）**

###### ****三 综合题****

**1、根据下面的DFS程序框架回答问题：**

**void DFS(Graph g, int v) //g是图的邻接矩阵或邻接表，v是g的一个顶点。**

**{  v.color = GREY;       //进入访问**

**;**

**for each w in g.neighbor(v)  //找顶点v的所有相邻点**

**if (w.color == WHITE)**

**;**

**DFS(g,w); //找顶点v的未访问过的相邻点w**

**;**

**elseif (w.color == GREY)**

**;**

**else**

**;**

**;**

**v.color = BLACK;**

**}**

**(1)  上面的DFS算法能否遍历图g的所有顶点？如果能给出证明，否则说明理由并给出补救措施。**

**(2)** **如果要对树边、交叉边、后继边和返回边都要做处理，而且还希望给所有顶点建立逆拓扑排序（设全局计数器toponum和顶点拓扑编号数组topo[]都是全局变量，且初值都是0），请在算法中的空白处分别标明。**

**2、根据下面的BFS程序框架回答问题。**

**void BFS(Graph g, int v ) //图结构为邻接矩阵或邻接表的BFS算法，v是起始顶点。**

**{ queue<int> qu; //定义一个队列qu**

**visited[v] = 1; //置已访问标记**

①**;**

**qu.push(v); //v进队**

**while (!qu.empty()) //队列不空时循环**

**{ w = qu.pop(); //出队顶点w**

**②                                ;**

**for each u in g.neibor(w) //找与顶点w相邻的顶点**

**if (visited[u]==0)**

**{ ③                                ;**

**visited[u]=1; //置该顶点已被访问的标志**

**qu.push(u); //该顶点进队**

**}**

**}**

**}**

**（1）上述BFS程序是否能遍历图g的所有顶点？如果能给出证明，如果不能说明理由，并给出补救措施。**

**（2）给定连通图g及其两顶点v和t，用BFS方法求顶点v到t的最短路径，请在程序空白处分别标明要添加的操作代码。**

**3、根据下面的程序块，回答问题。其中n是输入规模, x是解向量。**

**void search (int t，int[] x)**

**{   if (t>n) {**

**output(x);**

**return;**

**}**

**for (int i=t;i<=n;i++) {**

**swap(x[t], x[i]);**

**if (legal(t, x)) search (t+1, x);**

**swap(x[t], x[i]);**

**}**

**}**

**（1）上面的程序采用的是哪一种策略？空间搜索树是什么类型？解向量x的初值是什么？**

**（2）用这段程序是否可以求解n皇后问题？如果能，请写出合法性判定函数 legal(t, x). 如果不能，请说明理由。**

**4. 根据下面的程序块，回答问题。其中n是输入规模，x是解向量。**

**void search (int t,int[] x)**

**{   if (t>n){**

**output(x);**

**return;**

**}**

**for (int i=0;i<=1;i++) {**

**x[t]=i;**

**if (legal(t, i, x)) search (t+1,x);**

**}**

**}**

**（1）上面的程序采用的是哪一种策略？空间搜索树是什么类型？解向量x的初值是什么？**

**（2）用这段程序是否可以求解0-1背包问题？如果能，请写出合法性判定函数 legal(t, i， x)。如果不能，请说明理由。**

**5、用动态规划法求解兑换零钱问题：n个可换的钱币值按从小到大分别是1= c1<c2<…<cn, 每种币值的钱币个数无限，需要换的整钱是B，要求换的零钱个数最少。**

**（1）定义子问题p(i, b). 设最优函数为dp，建立求dp的递推关系。**

**（2）写出动态规划算法求dp和解向量x，x[i]表示零钱ci的个数。**

**（3）分析算法的时间复杂度和空间复杂度。**

**6、根据下面的程序块回答问题**

**long f(int n) //n是正整数**

**{ if (n<5) return n;**

**long prod=0;**

**for (i=1; i<n; i++)**

**prod = max{prod, f(i)\*(n-i)};**

**return prod;**

**}**

**（1）写出求f(n)的递归模型，并分析该算法的时间复杂度。**

**（2）写一个改进算法，降低时间复杂度。**

**7、用分支限界法求解0-1背包问题：n个物品的重量和价值分别是数组w[0…n-1]和v[0…n-1]，背包容量B，装入物品到背包中，使得背包中物品总价值最大。**

1. **设计队列结点结构,写出定界函数bound( )，参数自行定义；**
2. **处理当前出队的结点e时，需要添加新结点到队列qu. 写一个函数**

**addNode2Qu(qu, e)**

**实现这个过程。**

**8. 用分支限界法求解装载问题：有一批共*n*个集装箱要装上一艘载重量为*C*的轮船，其中集装箱的重量为数组*w*[1…n]. 求一个合理的装载方案使得装上轮船的集装箱总重量尽可能的多。**

**（1）给出队列qu的结点类型定义， 并确定限界（剪枝）函数的计算方法。**

**（2）设e是当前出队的结点，如果e不是树叶结点，则应该如何添加新结点，写出根据e添加新结点的函数：addNode2Qu(qu, e), 其它需要的数据作为全局变量。**

**9. 对于任意正整数c≥2，假设一组硬币的面值为*Dn* = {1, *c*, *c*2, …, *cn*-1}，现在要换金额为*S*的钱，最少需要多少硬币？**

1. **给出一个贪心选择策略，根据该策略是否能得到问题的最优解，说明理由。**
2. **具体描述该问题的贪心算法，并分析算法的时间复杂度。**

**10．设*S*={1，…，*n*}是使用某资源的*n*个活动所组成的集合，该资源任何时刻只能被一个活动所占用，活动*i*的开始时间*si*和结束时间*fi*。设计贪心算法求一种最优活动安排方案，使得所有安排的活动个数最多。**

**（1）确定贪心选择策略，并证明该策略能得到问题的最优解。**

**（2）写出贪心算法，并分析时间复杂度。**

**11、给定两个数组A[0…n-1]和B[0…n-1]，都已按从小到大排序，求两个数组合并后的中位数。**

**（1）算法1：将两个数组按顺序合并得到的数组C[0…2n-1]仍然保持从小到大排序，输出C[n – 1]. 该算法的时间复杂度是多少？**

**（2）写一个分治算法改进上面的算法1，写出算法描述中的递归函数findMedian（），参数根据需要自行定义。并分析时间复杂度，是否优于算法1？**