《算法设计与分析》复习提纲

第1章 算法概述

一、基本概念

1．算法：算法是指解决问题的一种方法或一个过程。

2．算法的性质

输入，输出，确定性，有限性，有效性。

3．算法复杂性：算法所需要的计算机资源。

4．算法的时间复杂性T(n)：算法需要时间资源的量。

5．算法的空间复杂性S(n)：算法需要空间资源的量。

1. 算法复杂性在渐近意义下的阶

(1)渐近上界记号O：*O*(*g*(*n*)) = { *f*(*n*) | 存在正常数*c*和*n*0使得对所有*n*≥ *n*0有：0 ≤ *f*(*n*) ≤ *cg*(*n*) }

(2)渐近下界记号Ω ：Ω (*g*(*n*)) = { *f*(*n*) | 存在正常数*c*和*n*0使得对所有*n*≥ *n*0有：0≤ *cg*(*n*) ≤ *f*(*n*) }

(3)非紧上界记号o：*o*(*g*(*n*)) = { *f*(*n*) | 对于任何正常数*c*>0*，*存在正数和*n*0 >0使得对所有*n*≥ *n*0有：0 ≤ *f*(*n*)<*cg*(*n*) }

(4)非紧下界记号*ω*：*ω* (*g*(*n*)) = { *f*(*n*) | 对于任何正常数*c*>0*，*存在正数和*n*0 >0使得对所有*n*≥ *n*0　　　有：0 ≤ *cg*(*n*) < *f*(*n*) }

(5)紧渐近界记号Θ：Θ (*g*(*n*)) = { *f*(*n*) | 存在正常数*c*1*,c*2和*n*0使得对所有*n*≥ *n*0有：*c*1*g*(*n*) ≤ *f*(*n*) ≤ *c*2*g*(*n*)

7．定理 设f和g是定义域为自然数集合的函数。

(1)如果=c，c >0为常数，则f(n)=Θ(g(n))。

(2)如果=0，则f(n)=o(g(n))。

(3)如果=+∞，则f(n)=ω(g(n))。

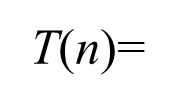
二、典型问题

1．递归算法的复杂性分析



*T*(*n*)=*a*n-1 *T*(1)+

2．递归算法的复杂性分析



三、例题

1．二分搜索技术时间复杂度T(n)=T(n/2)+O(1)

　解：

　　T(n) = T(n/2) + 1

　　　　= T(n/2^2) + 2

　　　　= T(n/2^3) + 3

　　　　= ...

　　　　= T(n/2^(log2(n))) + log2(n)

　故复杂度是O(Log2n)

2．单循环时间复杂度

*T*(*n*)= =*n*-1=*O*(*n*)

3．双循环时间复杂度

*T*(*n*)= ===(*n*-1)*n*/2=*O*(*n*2)

第2章 递归与分治策略

一、基本概念

1．递归：直接或间接地调用自身的算法称为递归算法。用函数自身给出定义的函数称为递归函数。

2．分治法的设计思想：将一个难以直接解决的大问题，分解为规模较小的相同子问题，直至这些  
 子问题容易直接求解，并且可以利用这些子问题的解求出原问题的解。各个击破，分而治之。

3．最优子结构性质：问题的最优解包含子问题的最优解。反过来说就是，我们可以通过子问题的  
 最优解，推导出问题的最优解。

二、典型适用问题

1．二分搜索技术，时间复杂度T(n)=T(n/2)+O(1) T(n)= O(logn)

2．合并排序，时间复杂度 T(n)=2T(n/2)+O(n) T(n)=O(nlogn)

3．快速排序，平均时间复杂度T(n)=2T(n/2)+O(n) T(n)=O(nlogn)， 最坏时间复杂度 O(n2)

4．棋盘覆盖，T(k)=4T(k-1)+O(1) T(n)=O(4k)

三、例题

二分搜索算法：

template<class Type>

int BinarySearch(Type a[], const Type& x, int l, int r)

{ while (r >= l)

{ int m = (l+r)/2;

if (x == a[m]) return m;

if (x < a[m]) r = m-1; else l = m+1;

}

return -1;

}

第3章 动态规划

一、基本概念

1．最优子结构性质：问题的最优解包含子问题的最优解。反过来说就是，我们可以通过子  
 问题的 最优解，推导出问题的最优解。

2．重叠子问题性质：求解问题时，每次产生的子问题并不总是新问题，有些子问题被反复  
 计算多次。

3．动态规划算法设计思想：其思想把求解的问题分成许多阶段或多个子问题，然后按顺序  
 求解各子问题。最后一个阶段或子问题的解就是初始问题的解。

4．动态规划基本步骤：

(1)找出最优解的性质，并刻划其结构特征。

(2)递归地定义最优值。

(3)以自底向上的方式计算出最优值。

(4)根据计算最优值时得到的信息，构造最优解。

二、典型适用问题

1．0-1背包问题，时间复杂度为 O(min{nc,2n})，空间复杂度为O(n2)

2．矩阵连乘问题，时间复杂度为O(n3)，空间复杂度为O(n2)

3．最长公共子序列，时间复杂度为 O(mn)

4．最大子段和，时间复杂度为 O(n)

三、例题

1．0-1背包问题

问题描述：给定n种物品和一背包。物品i的重量是wi，其价值为vi，背包的容量为c。

问题：应如何选择装入背包中的物品，使得装入背包中物品的总价值最大。

说明：在选择装入背包的物品时，对每种物品i只有两个选择，装入背包或不  
 装入背包，也不能将物品装入背包多次。

给定c>0，wi>0，vi>0，1≤i ≤n，要求找出一个n元0-1向量(x1,x2,…xn)，  
 xi ∈{0,1}，使得

 且  达到最大。

0-1背包问题的子问题结构为





的最优值为m(i，j)，即m(i，j)是背包容量为j，可选择物品为i，i+1，…，n时0-1背包问题的最优值。由0-1背包问题的最优子结构性质，可以建立计算m(i，j)的递归式如下。





第4章 贪心算法

一、基本概念

1．最优子结构性质：问题的最优解包含子问题的最优解。反过来说就是，我们可以通过子

问题的 最优解，推导出问题的最优解。

2．贪心选择性质：指所求问题的整体最优解可以通过一系列局部最优的选择，即贪心选择  
 来达到。这是贪心算法可行的基本要素，也是贪心算法与动态规划算法的主要区别。

3．贪心算法设计思想：贪心算法总是作出在当前看来最好的选择。也就是说贪心算法并不  
 从整体最优考虑，它所作出的选择只是在某种意义上的局部最优选择。

二、典型适用问题

1. 活动安排问题，时间复杂度O(nlogn)，

2. 哈夫曼编码，时间复杂度O(nlogn)

3. 单源最短路径，时间复杂度O(n2)

4. 最小生成树Prim，时间复杂度O(n2) n为结点数

5．最小生成树Kruskal，时间复杂度O(eloge) e为边数

三、例题

已知n个学生的身高分别为h1, h2, … , hn，若要求计算这n个学生的最小身高差 ，请为该问题设计一个有效算法，并对所设计的算法进行时间复杂性分析。

要求：（1）给出算法基本思想描述。

（2）给出算法的伪代码描述。

（3）对所设计算法进行时间复杂性分析。需要给出步骤和结果，包括：确定问题规模参数，算法基本操作，影响基本操作执行次数因素等。

说明：不要求代码实现。

（1）算法基本思想描述：首先对n个学生按身高进行非递减排序，初始化当前身高差最小值为身高最大值或极大值；然后从2～n扫描学生，分别计算其与前一位学生身高差，若身高差小于当前最小值，替换当前身高差最小值。

（2）算法伪代码描述：

float getMinDiff(a)

输入：学生身高数组a[1-n]

返回：最小身高差minDiff

sort(a)//按学生身高非递减排序

minDiff ← MAXDIFF

for i←1 to n-1

min ← a[i] – a[i-1]

if min < minDiff

minDiff ← min

return minDiff

（3）在不考虑排序情况下：

问题规模参数为输入数组中元素个数；

算法基本操作为min ← a[i] – a[i-1]；

影响基本操作执行次数的因素只取决于问题规模n；

基本操作执行次数为n-1次；时间复杂度为O(n)。

但前提需排序，排序可以在O(nlogn)时间复杂度下完成，因此算法时间复杂度为取决于排序时间复杂度，为O(nlogn)。

第5章 回溯法

一、基本概念

1．问题解的表示：回溯法将一个问题的解表示成一个n元式(x1,x2,…,xn)的形式。

2．显示约束：对分量xi的取值限定。

3．隐示约束：为满足问题的解而对不同分量之间施加的约束。

4．解空间：对于问题的一个实例，解向量满足显式约束条件的所有多元组，构成了该实例  
 的一个解空间。

5．子集树：当所给的问题是从n个元素的集合S中找出满足某种性质的子集时，相应的解  
 空间树称为子集树。子集树通常有2n个叶结点。

6．排列树：当所给的问题是确定n个元素满足某种性质的排列时，相应的解空间树称为排  
 列树。排列树通常有n!个叶结点。

7．回溯法思想：具有限界函数的深度优先生成法。

二、典型适用问题

1．n后问题，时间复杂度 O(nn+1)

2．装载问题,，时间复杂度 O(2n)

3．批处理作业调度，时间复杂度 O(n!)

三、例题

1．设某一机器由n个部件组成，每一种部件都可以从m个不同的供应商处购得。设wij是从供应商j处购得零件i的重量，cij是相应的价格。现在希望得到一个总价格不超过d的最小重量机器设计方案，若采用回溯发求解问题，请完成以下任务：

（1）给出问题解的形式。

（2）给出回溯法中节点下探的条件。

（3）填空完成回溯法求解问题的主要方法backtrack方法。

解：

（1）问题解形式为n个零件的供应商向量{x1, x2, …, xn}，1≤xi≤m

（2）节点下探条件：当前选择零件重量总和小于当前找到的机器最小重量；当前选择零件价格总和小于d。

（3）代码框架

template<class Typew, class Typep>

bool Machine<Typew, Typep>::Backtrack(int i){

if(i>n){

bestw = cw;

for(int j=1; j<=n; j++) bestx[j]=x[j];

return true;

}

bool found=false;

if(cp<=d) found=true;

for(int j=1; j<=m; j++){//2’

x[i]=j;

cw+=w[i][j];

cp+=c[i][j];

if(cp<=d && cw<bestw) if(Backtrack(i+1)) found=true;3’,3’

cw-=w[i][j];

cp-=c[i][j];

}

return found;

}

2．羽毛球队有男女运动员各n人，给定两个n阶矩阵P和Q，P[i][j]表示男运动员i和女运动员j配对组成混合双打男运动员的竞赛优势，Q[i][j]表示女运动员i和男运动员j配对组成混合双打女运动员的竞赛优势。由于技术配合与心理等各种因素影响，P[i][j]不一定等于Q[j][i]。男运动员i和女运动员j配对双打的整体竞赛优势为P[i][j]\*Q[j[i]。请设计算法，计算男女运动员的最佳搭配，使得各组运动员整体竞赛优势总和最大。

要求：

（1）分析该问题适合用何种算法求解。

（2）给出问题解的形式。

（3）给出求解问题的代码框架。

解：

（1）求解排列最优值问题，适合用回溯法求解。

（2）问题解形式为n个女队员的一个最优排列{x1, x2, …, xn}，1≤xi≤n

（3）代码框架

final class Pairing{

//属性与构造函数

…

public void backtrack(int d){

if(d == n){//到达叶节点

if(cv > maxValue){//找到一组更优配对

for(int i=0; i<n; i++){//更新配对向量

optimalPair[i] = curPair[i];

}

maxValue = cv;

}

}

else{//下探排列树

for(int j=d; j<n; j++){

swap(curPair, d, j);

cv += p[d][curPair[j]]\*q[curPair[j]][d];

backtrack(d+1);

cv -= p[d][curPair[j]]\*q[curPair[j]][d];

swap(curPair, d, j);

}

}

}

}

第6章 分支限界法

一、基本概念

1．队列式(FIFO)分支限界法：按照队列先进先出（FIFO）原则选取下一个节点为扩展节点。

2．优先队列式分支限界法：按照优先队列中规定的优先级选取优先级最高的节点成为当前  
 扩展节点。

3．分支限界法设计思想：以广度优先或以最小耗费（最大效益）优先的方式搜索问题的解  
 空间树。

二、典型适用问题

1．单源最短路径问题，时间复杂度 O(n!)

2．布线问题，时间复杂度 O(mn)

第7章 随机化算法

一、基本概念

1．随机算法通常分成4类：数值随机算法、蒙特卡罗算法、拉斯维加斯算法、舍伍德算法。如果取得结果的途径是随机的，则称为随机算法，如拉斯维加斯算法； 如果取得的解是否正确存在随机性，称为概率算法，如蒙特卡罗算法。

2．数值随机算法（Numericalrandomized algorithm）：数值化随机算法常用于数值问题求解。这类算法得到的往往是近似解，且近似解的精度随着计算时间的增加而不断提高。在许多情况下，要计算出问题的精确解是不可能的或没有必要的，用数值化税基算法可得到相当满意的解。

3． 蒙特卡罗算法（MonteCarlo）:不保证所求得的解是正确的，也就是说，蒙特卡罗算法求 得的解有时是错误的。不过，由于可以设法控制这类算法得到错误解的概率，并因它的简单高效，是很有价值的一类随机算法。一般情况下，蒙特卡罗算法求得正确解的概率随计算时间的增加而增大。但无论如何不能确保解的正确性，而且通常无法有效地判断所求得的解究竟是否正确，这是蒙特卡罗算法的缺陷。

4．拉斯维加斯算法（LasVegas）：求得的解总是正确的，但有时拉斯维加斯算法可能始终找不到解。使用拉斯维加斯算法求解同一问题的同一实例，能够得到相同的结果，但算法的执行时间会不一样。一般情况下，求得正确解的概率随计算时间的增加而增大。因此，为了减少求解失败的概率，可以使用一个拉斯维加斯算法对同一实例，重复多次执行该算法。

5．舍伍德算法（Sherwood）:总能求得问题的正确解。当一个确定性算法在最坏情况下的计算复杂度与其在平均情况下的计算复杂度两者相差较大时，可以在这个确定算法中引入随机性将它改造成一个舍伍德算法，用来消除或减少问题的不同实例之间这种在计算时间上的差别。舍伍德算法的精髓不是避免算法的最坏情况行为，而是设法消除这种最坏行为与特定实例之间的关联性。

**一、选择题**

1、二分搜索算法是利用（ A ）实现的算法。

A、分治策略   B、动态规划法   C、贪心法    D、回溯法

2、下列不是动态规划算法基本步骤的是（ A ）。

A、找出最优解的性质   B、构造最优解   C、算出最优解   D、定义最优解

3、最大效益优先是（ A ）的一搜索方式。

A、分支界限法      B、动态规划法    C、贪心法    D、回溯法

4、在下列算法中有时找不到问题解的是（ B ）。

A、蒙特卡罗算法    B、拉斯维加斯算法   C、舍伍德算法   D、数值概率算法

5. 回溯法解旅行售货员问题时的解空间树是（ A ）。

A、子集树 B、排列树 C、深度优先生成树 D、广度优先生成树

6．下列算法中通常以自底向上的方式求解最优解的是（ B ）。

A、备忘录法 B、动态规划法 C、贪心法 D、回溯法

7、衡量一个算法好坏的标准是（ C ）。  
A 运行速度快 B 占用空间少 C 时间复杂度低 D 代码短  
8、以下不可以使用分治法求解的是（ D ）。  
A 棋盘覆盖问题 B 选择问题 C 归并排序 D 0/1背包问题  
9. 实现循环赛日程表利用的算法是（ A ）。

A、分治策略 B、动态规划法 C、贪心法 D、回溯法

10、下列随机算法中运行时有时候成功有时候失败的是（ C ）  
A 数值概率算法 B 舍伍德算法 C 拉斯维加斯算法 D 蒙特卡罗算法

11．下面不是分支界限法搜索方式的是（ D ）。

A、广度优先 B、最小耗费优先 C、最大效益优先 D、深度优先

12．下列算法中通常以深度优先方式系统搜索问题解的是（ D ）。

A、备忘录法 B、动态规划法 C、贪心法 D、回溯法

13.备忘录方法是那种算法的变形。（ B ）

A、分治法 B、动态规划法 C、贪心法 D、回溯法

14．哈弗曼编码的贪心算法所需的计算时间为（ B ）。

A、O（n2n） B、O（nlogn） C、O（2n） D、O（n）

15．分支限界法解最大团问题时，活结点表的组织形式是（ B ）。

A、最小堆 B、最大堆 C、栈 D、数组

16．最长公共子序列算法利用的算法是（ B ）。

A、分支界限法 B、动态规划法 C、贪心法 D、回溯法

17．实现棋盘覆盖算法利用的算法是（ A ）。

A、分治法 B、动态规划法 C、贪心法 D、回溯法

18.下面是贪心算法的基本要素的是（ C ）。

A、重叠子问题 B、构造最优解 C、贪心选择性质 D、定义最优解

19.回溯法的效率不依赖于下列哪些因素（ D ）

A.满足显约束的值的个数 B. 计算约束函数的时间

C. 计算限界函数的时间 D. 确定解空间的时间

20.下面哪种函数是回溯法中为避免无效搜索采取的策略（ B ）

A．递归函数 B.剪枝函数 C。随机数函数 D.搜索函数

21、下面关于NP问题说法正确的是（ B ）  
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

22、蒙特卡罗算法是（ B ）的一种。

A、分支界限算法      B、概率算法    C、贪心算法    D、回溯算法

23.下列哪一种算法不是随机化算法（ C ）

A. 蒙特卡罗算法B. 拉斯维加斯算法C.动态规划算法D.舍伍德算法

24. （ D ）是贪心算法与动态规划算法的共同点。

A、重叠子问题 B、构造最优解 C、贪心选择性质 D、最优子结构性质

25. 矩阵连乘问题的算法可由（ B ）设计实现。

A、分支界限算法      B、动态规划算法    C、贪心算法    D、回溯算法

26. 分支限界法解旅行售货员问题时，活结点表的组织形式是（ A ）。

A、最小堆 B、最大堆 C、栈 D、数组

27、Strassen矩阵乘法是利用（ A ）实现的算法。

A、分治策略   B、动态规划法   C、贪心法    D、回溯法

29、使用分治法求解不需要满足的条件是（ A ）。  
A 子问题必须是一样的  
B 子问题不能够重复  
C 子问题的解可以合并  
D 原问题和子问题使用相同的方法解  
30、下面问题（ B ）不能使用贪心法解决。  
A 单源最短路径问题 B N皇后问题   
C 最小花费生成树问题 D 背包问题  
31、下列算法中不能解决0/1背包问题的是（ A ）  
A 贪心法 B 动态规划 C 回溯法 D 分支限界法  
32、回溯法搜索状态空间树是按照（ C ）的顺序。  
A 中序遍历 B 广度优先遍历 C 深度优先遍历 D 层次优先遍历

33、下列随机算法中运行时有时候成功有时候失败的是（ C ）  
A 数值概率算法 B 舍伍德算法 C 拉斯维加斯算法 D 蒙特卡罗算法  
34．实现合并排序利用的算法是（ A ）。

A、分治策略 B、动态规划法 C、贪心法 D、回溯法

35．下列是动态规划算法基本要素的是（ D ）。

A、定义最优解 B、构造最优解 C、算出最优解 D、子问题重叠性质

36．下列算法中通常以自底向下的方式求解最优解的是（ B ）。

A、分治法 B、动态规划法 C、贪心法 D、回溯法

37．采用广度优先策略搜索的算法是（ A ）。

A、分支界限法 B、动态规划法 C、贪心法 D、回溯法

38、合并排序算法是利用（ A ）实现的算法。

A、分治策略   B、动态规划法   C、贪心法    D、回溯法

39、在下列算法中得到的解未必正确的是（ B ）。

A、蒙特卡罗算法    B、拉斯维加斯算法   C、舍伍德算法   D、数值概率算法

40、背包问题的贪心算法所需的计算时间为（ B ）

A、O（n2n）     B、O（nlogn）    C、O（2n）      D、O（n）

41．实现大整数的乘法是利用的算法（ C ）。

A、贪心法 B、动态规划法 C、分治策略 D、回溯法

42．0-1背包问题的回溯算法所需的计算时间为（ A ）

A、O（n2n） B、O（nlogn） C、O（2n） D、O（n）

43．采用最大效益优先搜索方式的算法是（ A ）。

A、分支界限法 B、动态规划法 C、贪心法 D、回溯法

44．贪心算法与动态规划算法的主要区别是（ B ）。

A、最优子结构 B、贪心选择性质 C、构造最优解 D、定义最优解

45. 实现最大子段和利用的算法是（ B ）。

A、分治策略 B、动态规划法 C、贪心法 D、回溯法

46.优先队列式分支限界法选取扩展结点的原则是（ C ）。

A、先进先出 B、后进先出 C、结点的优先级 D、随机

47.背包问题的贪心算法所需的计算时间为（ B ）。

A、O（n2n） B、O（nlogn） C、O（2n） D、O（n）

48、广度优先是（ A ）的一搜索方式。

A、分支界限法      B、动态规划法    C、贪心法    D、回溯法

49、舍伍德算法是（ B ）的一种。

A、分支界限算法      B、概率算法    C、贪心算法    D、回溯算法

50、在下列算法中有时找不到问题解的是（ B ）。

A、蒙特卡罗算法    B、拉斯维加斯算法   C、舍伍德算法   D、数值概率算法

51下列哪一种算法是随机化算法（ D ）

A. 贪心算法B. 回溯法C.动态规划算法D.舍伍德算法

52. 一个问题可用动态规划算法或贪心算法求解的关键特征是问题的（ B ）。

A、重叠子问题 B、最优子结构性质 C、贪心选择性质 D、定义最优解

53．采用贪心算法的最优装载问题的主要计算量在于将集装箱依其重量从小到大排序，故算法的时间复杂度为 ( B ) 。

A、O（n2n） B、O（nlogn） C、O（2n） D、O（n）

54. 以深度优先方式系统搜索问题解的算法称为 ( D ) 。

A、分支界限算法      B、概率算法    C、贪心算法    D、回溯算法

55. 实现最长公共子序列利用的算法是（ B ）。

A、分治策略 B、动态规划法 C、贪心法 D、回溯法

**二、 填空题**

1.算法的复杂性有 时间 复杂性和 空间 复杂性之分。

2、程序是 算法 用某种程序设计语言的具体实现。

3、算法的“确定性”指的是组成算法的每条 指令 是清晰的，无歧义的。

4.矩阵连乘问题的算法可由 动态规划 设计实现。

5、拉斯维加斯算法找到的解一定是 正确解 。

6、算法是指解决问题的 一种方法 或 一个过程 。

7、从分治法的一般设计模式可以看出，用它设计出的程序一般是 递归算法 。

8、问题的 最优子结构性质 是该问题可用动态规划算法或贪心算法求解的关键特征。

9、以深度优先方式系统搜索问题解的算法称为 回溯法 。

10、数值概率算法常用于 数值问题 的求解。

11、计算一个算法时间复杂度通常可以计算 循环次数 、基本操作的频率 或计算步。

12、利用概率的性质计算近似值的随机算法是 数值概率算法 ，运行时以一定的概率得到正确解的随机算法是 蒙特卡罗算法 。

14、解决0/1背包问题可以使用动态规划、回溯法和分支限界法，其中不需要排序的是 动态规划 ，需要排序的是 回溯法 ， 分支限界法 。  
15、使用回溯法进行状态空间树裁剪分支时一般有两个标准：约束条件和目标函数的界，N皇后问题和0/1背包问题正好是两种不同的类型，其中同时使用约束条件和目标函数的界进行裁剪的是 0/1背包问题 ，只使用约束条件进行裁剪的是 N皇后问题 。

16、 贪心选择性质 是贪心算法可行的第一个基本要素，也是贪心算法与动态规划算法的主要区别。

17、矩阵连乘问题的算法可由 动态规划 设计实现。

18、拉斯维加斯算法找到的解一定是 正确解 。

19.贪心算法的基本要素是 贪心选择 质和 最优子结构 性质 。

21. 动态规划算法的基本思想是将待求解问题分解成若干 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。

22.算法是由若干条指令组成的有穷序列，且要满足输入、 输出 、确定性和 有限性 四条性质。

23、大整数乘积算法是用 分治法 来设计的。

24、以广度优先或以最小耗费方式搜索问题解的算法称为 分支限界法 。

25、舍伍德算法总能求得问题的 一个解 。

26、 贪心选择性质 是贪心算法可行的第一个基本要素，也是贪心算法与动态规划算法的主要区别。

27.快速排序算法是基于 分治策略 的一种排序算法。

28.动态规划算法的两个基本要素是 最优子结构 性质和 重叠子问题 性质 。

30.回溯法是一种既带有 系统性 又带有 跳跃性 的搜索算法。

31.分支限界法主要有 队列式（FIFO） 分支限界法和 优先队列式 分支限界法。

32．分支限界法是一种既带有 系统性 又带有 跳跃性 的搜索算法。

33．回溯法搜索解空间树时，常用的两种剪枝函数为 约束函数 和 限界函数 。

34.任何可用计算机求解的问题所需的时间都与其 规模 有关。

35.快速排序算法的性能取决于 划分的对称性 。

**三、算法填空**

1.背包问题的贪心算法

void Knapsack(int n,float M,float v[],float w[],float x[])

{

Sort(n,v,w);

int i;

for (i=1;i<=n;i++) x[i]=0;

float c=M;

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

if (w[i]>c) break;

x[i]=1;

c - =w[i];

}

if (i<=n) x[i]=c/w[i];

}

2.最大子段和: 动态规划算法

int MaxSum(int n, int a[])

{

int sum=0, b=0； //sum存储当前最大的b[j], b存储b[j]

for(int j=1； j<=n； j++) {

if (b>0) b+= a[j] ；

else b=a[i]; ； //一旦某个区段和为负，则从下一个位置累和

if(b>sum) sum=b;

}

return sum；

}

3.贪心算法求装载问题

template<class Type>

void **Loading**(int x[], Type w[], Type c, int n)

{

int \*t = new int [n+1];

;

for (int i = 1; i <= n; i++) x[i] = 0;

for (int i = 1; i <= n && w[t[i]] <= c; i++)

{x[t[i]] = 1;

;}

}

4.贪心算法求活动安排问题

template<class Type>

void **GreedySelector**(int n, Type s[], Type f[], bool A[])

{

A[1]=true;

int j=1;

for (int i=2;i<=n;i++) {

if (s[i]>=f[j])

{ A[i]=true;

j=i;

}

else A[i]=false;

}

}

5.快速排序

template<class Type>

void **QuickSort** (Type a[], int p, int r)

{

if (p<r) {

int q=Partition(a,p,r);

QuickSort (a,p,q-1); //对左半段排序

QuickSort (a,q+1,r); //对右半段排序

}

}

6.排列问题

Template <class Type>

void perm(Type list[], int k, int m )

{ //产生[list[k:m]的所有排列

if(k==m)

{ //只剩下一个元素

for (int i=0;i<=m;i++) cout<<list[i];

cout<<endl;

}

else //还有多个元素待排列，递归产生排列

for (int i=k; i<=m; i++)

{

swap(list[k]，list[i]);

perm(list,k+1;m);

swap(list[k],list[i]);

}

}

**四、问答题**

1.分治法的基本思想时将一个规模为n的问题分解为k个规模较小的子问题，这些子问题互相独立且与原问题相同。递归地解这些子问题，然后将各个子问题的解合并得到原问题的解。

2设计动态规划算法的主要步骤为：

（1）找出最优解的性质，并刻划其结构特征。（2）递归地定义最优值。（3）以自底向上的方式计算出最优值。（4）根据计算最优值时得到的信息，构造最优解。

1. 分治法与动态规划法的相同点与不同点分别是什么？

两者的相同点：将待求解的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。

两者的不同点是：适合于用动态规划法求解的问题，经分解得到的子问题往往不是互相独立的。而用分治法求解的问题，经分解得到的子问题往往是互相独立的。

1. 分支限界法与回溯法的相同点与不同点分别是什么？

相同点：都是一种在问题的解空间树T中搜索问题解的算法。

不同点：（1）求解目标不同；（2）搜索方式不同；（3）对扩展结点的扩展方式不同； （4）存储空间的要求不同。

5用回溯法搜索子集树的算法为：

void **backtrack** (int t)

{

if (t>n) output(x);

else

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

x[t]=i;

if (constraint(t)&&bound(t)) backtrack(t+1);

}

}

6. 分治法所能解决的问题一般具有的几个特征是：

（1）该问题的规模缩小到一定的程度就可以容易地解决；

（2）该问题可以分解为若干个规模较小的相同问题，即该问题具有最优子结构性质;

（3）利用该问题分解出的子问题的解可以合并为该问题的解；

（4）原问题所分解出的各个子问题是相互独立的，即子问题之间不包含公共的子问题。

7. 用分支限界法设计算法的步骤是：

(1)针对所给问题，定义问题的解空间（对解进行编码）；分

(2)确定易于搜索的解空间结构（按树或图组织解） ；

(3)以广度优先或以最小耗费（最大收益）优先的方式搜索解空间，并在搜索过程中用剪枝函数避免无效搜索。

8. 常见的两种分支限界法的算法框架

（1）队列式(FIFO)分支限界法：按照队列先进先出（FIFO）原则选取下一个节点为扩展节点。 （2）优先队列式分支限界法：按照优先队列中规定的优先级选取优先级最高的节点成为当前扩展节点。

9. 回溯法中常见的两类典型的解空间树是子集树和排列树。

当所给的问题是从n个元素的集合S中找出满足某种性质的子集时，相应的解空间树称为子集树。这类子集树通常有2n个叶结点，遍历子集树需O(2n)计算时间 。

当所给的问题是确定n个元素满足某种性质的排列时，相应的解空间树称为排列树。这类排列树通常有n!个叶结点。遍历排列树需要O(n!)计算时间。

10. 分支限界法的搜索策略是：

在扩展结点处，先生成其所有的儿子结点（分支），然后再从当前的活结点表中选择下一个扩展结点。为了有效地选择下一扩展结点，加速搜索的进程，在每一个活结点处，计算一个函数值（限界），并根据函数值，从当前活结点表中选择一个最有利的结点作为扩展结点，使搜索朝着解空间上有最优解的分支推进，以便尽快地找出一个最优解。

**五、算法题**

1. 给定已按升序排好序的n个元素a[0:n-1]，现要在这n个元素中找出一特定元素x，返回其在数组中的位置，如果未找到返回-1。

写出二分搜索的算法，并分析其时间复杂度。

1. template<class Type>

int BinarySearch(Type a[], const Type& x, int n)

{//在a[0:n]中搜索x，找到x时返回其在数组中的位置，否则返回-1

Int left=0; int right=n-1;

While (left<=right){

int middle=(left+right)/2;

if (x==a[middle]) return middle;

if (x>a[middle]) left=middle+1;

else right=middle-1;

}

Return -1;

}

时间复杂性为O(logn)

2. 利用分治算法写出合并排序的算法，并分析其时间复杂度

1. void MergeSort(Type a[], int left, int right)

{

if (left<right) {//至少有2个元素

int i=(left+right)/2; //取中点

mergeSort(a, left, i);

mergeSort(a, i+1, right);

merge(a, b, left, i, right); //合并到数组b

copy(a, b, left, right); //复制回数组a

}

}

算法在最坏情况下的时间复杂度为O(nlogn)。

3.N皇后回溯法

bool Queen::Place(int k)

{ //检查x[k]位置是否合法

for (int j=1;j<k;j++)

if ((abs(k-j)==abs(x[j]-x[k]))||(x[j]==x[k])) return false;

return true;

}

void Queen::Backtrack(int t)

{

if (t>n) sum++;

else for (int i=1;i<=n;i++) {

x[t]=i;

if ( 约束函数 ) Backtrack(t+1);

}

}

4.最大团问题

void Clique::Backtrack(int i) // 计算最大团

{ if (i > n) { // 到达叶结点

for (int j = 1; j <= n; j++) bestx[j] = x[j];

bestn = cn; return;}

// 检查顶点 i 与当前团的连接

int OK = 1;

for (int j = 1; j < i; j++)

if (x[j] && a[i][j] == 0) { // i与j不相连

OK = 0; break;}

if (OK ) { // 进入左子树

x[i] = 1; cn++;

Backtrack(i+1);

x[i] = 0; cn--; }

if ( cn + n - i > bestn ) { // 进入右子树

x[i] = 0;

Backtrack(i+1); }

}

5.最长公共子序列问题

6．哈弗曼编码算法

一、选择题（每题2分，10题，共20分）

1、1、关于记号*O*正确的定义是（ A ）。

A、*O*(g(n)) = {f(n) |存在正数c和n0使得对所有n≥n0有0≤f(n)≤cg(n)}

B、*O*(g(n)) = {f(n) |存在正数c和n0使得对所有n≥n0有0≤cg(n) ≤f(n)}

C、*O*(g(n)) = {f(n) |对任意正数c存在n0使得对所有n≥n0有0≤f(n)<cg(n)}

D、*O*(g(n)) = {f(n) |对任意正数c存在n0使得对所有n≥n0有0≤cg(n)< f(n)}

2、在无序元素序列中查找固定值x，若采用分治法查找，则求解该问题的算法时间复杂度为（ C ）

A、*O*(log*n*) B、*O*(*n*log*n*) C、*O*(*n*) D、*O*(*n*2)

3、最长公共子序列问题适合采用下面哪种算法求解（ B ）。

A、分治   B、动态规划   C、贪心     D、回溯

4、对于物品数量为n，背包容量为c的情况下的0-1背包问题，若用回溯法求解，算法的时间复杂度为（ D ）。

A、nc B、2nc C、c2n D、n2n

**二、填空题（每空2分，15空，共30分）**

1、算法是由若干条指令组成的有穷序列，且要满足输入，输出，有效，有限，确定性性质。算法的效率一般可以从时间，空间两个方面衡量。

2、假设算法A理论上的时间复杂度T(n)=2n，现在有两台同类型计算机M1和M2，M2的计算速度是M1的64倍。若在M1和M2上分别测试算法A，则在相同时间内，M1和M2能够求解的问题规模n1和n2的关系为 6+n1=n2 。若算法B理论上的时间复杂度为T(n)=n2，若在M1和M2上分别测试算法B，则M1和M2求解相同规模问题所耗费的时间t1和t2的关系为 t1 =64t2 。

3、用分治法求解问题算法时间复杂度典型的递推方程为，比如求n个元素全排列。



4、采用分治法解决问题时，算法时间复杂度分析典型的递推方程为

，对于*a*，*b*和*f*(*n*)的不同取值情况，算法时间复杂度分析。例如：

对于某一问题，若*f*(*n*) = *O*(1)且*a* = 1，则求解该问题的时间复杂度为 O() ；若*f*(*n*) = *O*(*n*)且*a* = *b*>1，则求解该问题的时间复杂度为 O(n) ；若*f*(*n*) = *O*(*n*)且*a* = *b*2>1，则求解该问题的时间复杂度为 O(n2) ；若已知*f*(*n*) = *O****(****n*)，且*a* ,*b*均大于1，若希望将算法时间复杂度降低为不超过线性情况，则a和b应满足的条件为： a<=b 。

**三、****算法设计与分析题（3题，每题10分，共30分）**

1、伪代码阅读，算法复杂的度分析，例如：

ALG(A[1..*n*])

IF *n*=1 THEN RETURN A[*n*]

ELSE *temp*=ALG(A[1.*.n*-1])

IF *temp*<A[*n*]

THEN RETURN *temp*

ELSE RETURN A[*n*]

首先，n=1是递归的结束条件，整个过程中只执行一次，temp的赋值语句从n开始每次减一，直到n=1结束，共执行n次递归，每次递归执行一次比较语句，所以也执行n次，最后执行n次返回语句，所以最终时间复杂度为n。

2、动态规划算法求解问题，给出递推方程，求解具体问题解，例如：数塔问题

7

2

0

1

8

8

3

7

4

4

自底向上，找出相邻两个数的最大值，与双亲结点相加 ，得到的数就为当前情况的最大值，得到n-1个数，以此类推，n=1，即为最终的最大值。

首先，设给出的数塔为a[n][n];

设置存储最大值数据b[n][n]

递推方程：i>=0;j<=i;

3、设计算法解决具体问题，要求：给出算法基本思想描述，伪代码描述，对所设计算法进行复杂度分析。例如：找无序数值序 列的最小差值问题。

算法设计基本思想：首先可以对所有数进行排序，设置初始最小差值为最大值，然后分别比较相邻两个数的差值，若比当前最小差值小，则将最小差值替换为当前差值。当比较完后，最小差值即整个无序数列的最小差值

伪代码：

Int min\_dif(int arr[])

{

Qsort(arr);

Int min = maxvalue;

For(int i=0;i<arr.len-1;i++)

{

Int dif = Arr[i+1]-arr[i];

If(dif<min)

Min = dif;

}

Return min;

}

复杂度分析：快排复杂度,循环时间复杂度：n，取大者，所以时间复杂度为。

# 其他班：

先对这些无序数值进行递减排序

然后两两相邻的数值进行比较，求出差值

再在差值中找到最小值，得到问题的解

Float get\_min()

{

输入；

输出；‘’

Sort(a);

Init = max\_value;

For(i=1;i<n-1;i++)

{ min=a[i]-a[i-1] ; if(min<init) init=min;}

Return init;

}

复杂度分析：排序时间复杂度，处理的复杂度

**四、算法实现题（1题，共20分）**

设计算法并实现算法代码框架解决具体问题，例如：配对问题。

羽毛球队有男女运动员各*n*人，给定两个*n*阶矩阵P和Q，P[*i*][*j*]表示男运动员*i*和女运动员*j*配对组成混合双打男运动员的竞赛优势，Q[*i*][*j*]表示女运动员*i*和男运动员*j*配对组成混合双打女运动员的竞赛优势。由于技术配合与心理等各种因素影响，P[*i*][*j*]不一定等于Q[*j*][*i*]。男运动员*i*和女运动员*j*配对双打的整体竞赛优势为P[*i*][*j*]\*Q[*j*[*i*]。请设计算法，计算男女运动员的最佳搭配，使得各组运动员整体竞赛优势总和最大。

分析该问题适合用何种算法求解。

答：首先，回溯法适用于搜索问题和优化问题。而本题就是给定两个n阶矩阵，求p[i][j]\*q[j][i]的最大值，属于寻找最优值的问题，适合用回溯法求解。

给出问题解的形式。

最大配对值：p p为float类型值

配对模式：男 m[n] 有n个元素的一维数组

女f[n] 有n个元素的一维数组

给出求解问题的代码框架。

看老师给的代码。

# 这个答案应该是老师代码中的回溯部分

public void backtrack(int d){

if(d == n){//到达叶节点

if(cv > maxValue){//找到一组更优配对

for(int i=0; i<n; i++){//更新配对向量

optimalPair[i] = curPair[i];

}

maxValue = cv;

}

}

else{//下探排列树

for(int j=d; j<n; j++){

swap(curPair, d, j);

cv += p[d][curPair[j]]\*q[curPair[j]][d];

backtrack(d+1);

cv -= p[d][curPair[j]]\*q[curPair[j]][d];

swap(curPair, d, j);

}

}

}