《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))。

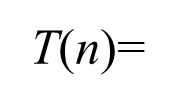
二、典型问题

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。

8．多米诺性质：如果当前结点不满足约束条件,能够推导出它的子结点也不满足约束条件。

如果子结点满足约束条件能够推导出其父结点满足约束条件。

9．剪枝函数：用约束函数在扩展结点处剪去不满足约束的子树；和用限界函数剪去得不到  
 最优解的子树。这两类函数统称为剪枝函数。采用剪枝函数，可避免无效搜索，提高回  
 溯法的搜索效率。

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++){’

x[i]=j;

cw+=w[i][j];

cp+=c[i][j];

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

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。

4．多米诺性质：如果当前结点不满足约束条件,能够推导出它的子结点也不满足约束条件。

如果子结点满足约束条件能够推导出其父结点满足约束条件。

5．剪枝函数：用约束函数在扩展结点处剪去不满足约束的子树；和用限界函数剪去得不到  
 最优解的dao子树。这两类函数统称为剪枝函数。

在分支限界法中使用剪枝函数，可以加速搜索。该函数给出每一个可行结点相应的子树  
 可能获得的最大价值的上界UB（最小价值的下界LB）。另一方面，也可以将上界函数确  
 定的每个结点的上界值作为优先级，以该优先级的非增序抽取当前扩展结点。这种策略  
 有时可以更迅速地找到最优解。

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。