算法设计与分析考试重点

一、题目类型：解答题、应用题、算法设计

二、解答题

1．名词解释（详细描述见手写）

·**决策树**：是在已知各种情况发生概率的基础上，通过构成决策树来求取净现值的期望值大于等于零的概率，评价项目风险，判断其可行性的决策分析方法，是直观运用概率分析的一种图解法。由于这种决策分支画成图形很像一棵树的枝干，故称决策树。

·**归纳法**：这种方法基于数学归纳法证明技术。给出一个带有参数 n 的问题，用归纳法设计一个算法，如果我们知道如何求解带有参数小于 n 的问题，那么我们的任务就化为如何把解法扩展到带有参数 n 的实例。

·**动态规划**：与分治算法的情况不一样，直接实现递推结果，导致了不止一次的递归调用，因此这种技术采取自底向上的方式递推求值，并把中间结果存储起来以便以后用来计算所需要求的解。广泛用于求解组合最优化问题。

·**最优子结构**：当一个问题的最优解包含其子问题的最优解时，称此问题具有最优子结构性质。

·**随机算法**：分治、动态规划、贪心法等每一个计算步骤是确定的，而随机算法每一步都是随机的。对于相同的输入每次随机算法产生的结果可能不同。

·**分支限界法**：采用广度优先遍历产生状态空间树的结点，并使用剪枝函数。

·**贪心算法**：对问题求解时，不从整体最优上加以考虑，而是做出一个看上去最优的决策（即局部最优解），并希望通过每次所做的局部最优解产生全局最优解。需要具有最优子结构和贪心选择性质。

·**分治法**：一个分治算法把问题实例划分为成若干子问题（通常是两个），并分别递归地解决每个子实例，然后把这些子实例的解组合起来，得到原问题实例的解。

·**回溯法**：一种选优搜索法，按选优条件向前搜索，以达到目标。但当搜索到某一步时，发现原先选择并不优或达不到目标，就退回一步重新选择，这种走不通就退回去再走的技术就是回溯法。大多使用深度遍历。

2．时间、空间复杂度衡量

时间复杂度：时间复杂度实际上是一个函数，代表基本操作重复执行的次数，进而分析函数虽变量的变化来确定数量级，数量级用 O 表示，所以算法的时间复杂度为： T（n）=O（f（n））；在一个算法存在最好、平均、最坏三种情况，我们一般关注的是最坏情况。

空间复杂度： 是对一个算法在运行过程中临时占用存储空间的度量

3．分治法思想：二分搜索

给定排好序的 n 个元素，在这 n 个元素找出一特定元素 x。把有序序列分成大致相同的两部分，然后取中间元素与特定元素 x 进行比较，如果 x 等于中间元素，算法停止，如果 x 小于中间元素，则在序列的左半部分继续查找，然后在序列的左半部分重复分解和治理操作；否则，在序列的右半部分继续查找，然后在序列的右半部分重复分解和治理操作

4．分治法求合并排序的思想与算法

合并排序法是将两个（或两个以上）有序表合并成一个新的有序表，即把待排序序列分为若干个子序列，每个子序列是有序的。然后再把有序子序列合并为整体有序序列。将已有序的子序列合并，得到完全有序的序列；即先使每个子序列有序，再使子序列段间有序。

5．子串与子序列比较

子串一定要连续，子序列可以不连续, 子序列就好比把子串一个个拆开，按顺序插入任意串里。

6．动态规划算法思想，可能与分治算法思想比较

动态规划算法与分治法类似，其基本思想也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。与分治法不同的是，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相互独立的。若用分治法来解这类问题，则分解得到的子问题数目太多，以至于最后解决原问题需要耗费指数时间。

动态规划也是一种分治思想（比如其状态转移方程就是一种分治），但与分治算法不同的是，分治算法是把原问题分解为若干个子问题，自顶向下求解子问题，合并子问题的解，从而得到原问题的解。动态规划也是把原始问题分解为若干个子问题，然后自底向上，先求解最小的子问题，把结果存在表格中，在求解大的子问题时，直接从表格中查询小的子问题的解，避免重复计算，从而提高算法效率。

三、应用题

1. 递推公式

找最大最小值比较次数：

$$C(n)=\begin{cases}1, & n=2\\ 2C(n/2)+2, & n>2\end{cases}$$

$$\begin{aligned}C(n) &= 2C\left(\tfrac{n}{2}\right)+2\\ &= 2\left(2C\left(\tfrac{n}{4}\right)+2\right)+2\\ &= 4C\left(\tfrac{n}{4}\right)+4+2\\ &= 4\left(2C\left(\tfrac{n}{8}\right)+2\right)+4+2\\ &= 8C\left(\tfrac{n}{8}\right)+8+4+2\\ &\cdots\cdots\\ &= 2^{k-1}C\left(\tfrac{n}{2^{k-1}}\right)+2^{k-1}+2^{k-2}+\cdots+2\\ &= 2^{k-1}C\left(\tfrac{n}{2^{k-1}}\right)+2^{k}-2\\ &= \tfrac{n}{2}\overset{1}{C(2)}+n-2\\ &= \tfrac{3n}{2}-2 \text{ 次比较.}\end{aligned}$$

等比求和： $\dfrac{2(1-2^{k-1})}{1-2} = 2^k-2$

n是2的整数幂 设 $n=2^k$

二分搜索比较次数：

$$C(n)\le\begin{cases}1, & n=1\\ 1+C(\lfloor n/2\rfloor), & n\ge 2\end{cases}$$

设对 $k\ge 2$ 有 $2^{k-1}\le n<2^k$

$\Rightarrow k-1\le \log n<k$

$\Rightarrow k\le \log n+1\le k+1$

$\Rightarrow k=\lfloor\log n\rfloor+1$

$$\begin{aligned}\Rightarrow C(n) &\le 1+C(\lfloor n/2\rfloor)\\ &\le 2+C(\lfloor n/4\rfloor)\\ &\ \ \vdots\\ &\le (k-1)+C(\lfloor n/2^{k-1}\rfloor)\\ &= k-1+1 = k = \lfloor\log n\rfloor+1\end{aligned}$$

$\Rightarrow C(n)\le\lfloor\log n\rfloor+1$ 最多比较次数.

合并算法比较次数：

合并比较次数在 $\frac{n}{2} \sim n-1$ 之间，所以：

最小比较次数：$C(n)=\begin{cases}0 & ,n=1\\ 2C(\frac{n}{2})+\frac{n}{2} & ,n\geq 2\end{cases}$

$$
\begin{aligned}
C(n) &= 2C(\tfrac{n}{2})+\tfrac{n}{2}\\
&= 2(2C(\tfrac{n}{4})+\tfrac{n}{4})+\tfrac{n}{2}\\
&= 4C(\tfrac{n}{4})+\tfrac{n}{2}+\tfrac{n}{2}\\
&= 4(2C(\tfrac{n}{8})+\tfrac{n}{8})+\tfrac{n}{2}+\tfrac{n}{2}\\
&= 8C(\tfrac{n}{8})+\tfrac{n}{2}+\tfrac{n}{2}+\tfrac{n}{2}\\
&\vdots\\
&= 2^k C(\tfrac{n}{2^k})+\tfrac{n}{2}k\\
&= n\cdot \overset{=0}{C(1)}+\tfrac{n\log n}{2}\\
&= \tfrac{n\log n}{2}
\end{aligned}
$$

设 $2^k=n$

$\Rightarrow k=\log n$

最大比较次数：$C(n)=\begin{cases}0 & ,n=1\\ 2C(\frac{n}{2})+n-1 & ,n\geq 2\end{cases}$

$$
\begin{aligned}
C(n) &= 2C(\tfrac{n}{2})+n-1\\
&= 2(2C(\tfrac{n}{4})+\tfrac{n}{2}-1)+n-1\\
&= 4C(\tfrac{n}{4})+2n-2-1\\
&= 4(2C(\tfrac{n}{8})+\tfrac{n}{4}-1)+2n-2-1\\
&= 8C(\tfrac{n}{8})+3n-4-2-1\\
&\vdots\\
&= 2^k C(\tfrac{n}{2^k})+kn-2^{k-1}-2^{k-2}-\cdots-1\\
&= 2^k C(1)+kn-(2^k-1)\\
&= kn-2^k+1 = n\log n-n+1
\end{aligned}
$$

→ 等比求和

2. 矩阵链相乘

3. 0-1 背包问题

4. 最短路径问题（迪杰斯特拉）

5. 最小耗费生成树（克鲁斯卡尔，普里姆）

四、算法设计

6.50. 设计一个分治算法，判定两棵给定的二叉树 T_1 和 T_2 是否相同

```
int EqualBtr(bitreptr t1, bitreptr t2) {
        if(t1 == NULL && t2 == NULL)
                return    1;
        if((t1==NULL&&t2!=NULL)||t1!=NULL&&t2==NULL)||
             (t1->data!=t2->data))
                return    0;
        hl = EqualBtr (t1->Lchild, t2->Lchild);
        hr = EqualBtr (t1->Rchild, t2->Rchild);
        if(hl==1&&hr==1)
                return 1;
        else
                return 0;
}
```

6.51：设计一个分治算法，求给定二叉树的高度

```
void Btdepth(bitreptr t, int k, int &h){
      //指针 t 指向二叉树的根结点，k, h 的初值均设为 0
      if (t! = NULL) {
              k++;       //表示结点的层次
              if (k>h)
                   h = k;
              Btdepth(t->Lchild,k,h);
              Btdepth(t->Rchild,k,h);
      }
}
```

14.13. 判断 AB=C

```
Public static boolean product(double [][]a,double [][]b, double [][]c,int n){
        cnd=new Random();
        double []x=new double [n+1];
        double []y=new double [n+1];
        double []z=new double [n+1];

        for(int i=1;i<=n;i++){
                x[i]=cnd.random(2);
                if(x[i]==0)    x[i]=-1;
        }
        mult(b,x,y,n);
        mult(a,y,z,n);
        mult(c,x,y,n);
        for(int i=1;i<=n;i++){
```

```
            if(y[i]!=z[i])      return false;
            else
                  return ture;
      }
}
```

14. 14. A=B^(-1)即 AB=E

```
Public static boolean product(double [][]a,double [][]b,int n){
      cnd=new Random();
      double []x=new double [n+1];
      double []y=new double [n+1];
      double []z=new double [n+1];
      Double []e=new double [n+1];

      for(int i=1;i<=n;i++){
              x[i]=cnd.random(2);
              if(x[i]==0)    x[i]=-1;
      }

      for(int i=1;i<=n:i++){
            for(int j=1;j<=n;j++)
                  if(i==j)    e[i][j]=1
                  else      e[i][j]=0;
      }
      mult(b,x,y,n);
      mult(a,y,z,n);
      mult(e,x,y,n);
      for(int i=1;i<=n;i++){
            if(y[i]!=z[i])      return false;
            else
                  return ture;
      }
}
```

二叉树叶子节点个数

```
int leaf(BitTree T){
  if(T==null)
     return 0;
  else if(T->lchild==null && T->rchild==null)
     return 1;
  else return leaf(T->lchild)+leaf(T->rchild);
}
```

3 着色问题

```
n 个顶点
1.for k=1 to n
2.    c[k]←—0
3.end for
4.k←—1
5.while k>=1
6.     While c[k]<= 2
7          c[k]←— c[k]+1
8          if c 为合法着色  then output c  退出所有循环
9          else  if c 为部分解  then k←—k+1
10     end while
11.    c[k]←—0
12.    k←—k-1  {回溯}
13.end while
```

8 皇后问题

```
1.for k=1 to 8
2.    c[k]←—0
3.end for
4.k←—1
5.while k>=1
6.     While c[k]<= 7
7          c[k]←— c[k]+1
8          if c 为合法布局  then output c  退出所有循环
9          else  if c 为部分解  then k←—k+1
10     end while
11.    c[k]←—0
12.    k←—k-1  {回溯}
13.end while
```

14.20 最小割问题

1.min=MAXINT，固定一个顶点 P
2.从点 P 用类似 prim 的 s 算法扩展出“最大生成树”，记录最后扩展的顶点和最后扩展的边
3.计算最后扩展到的顶点的切割值（即与此顶点相连的所有边权和），若比 min 小更新 min
4.合并最后扩展的那条边的两个端点为一个顶点
5.转到 2，合并 N-1 次后结束
6.min 即为所求，输出 min