**杭州师范大学杭州国际服务工程学院（信息科学与工程学院）**

班级： 学号： 姓名：

装 订 线

**2014-2015学年第2学期期末考试**

**《算法分析与设计》试卷（A）**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 题号 | 一 | 二 | 三 | 四 | 总分 |
| 得分 |  |  |  |  |  |
| 评卷签名 |  |  |  |  |  |

|  |  |
| --- | --- |
| 得分 |  |

# 填空（共20分，每空格2分）

1．算法是一个由 有限的指令集 组成的过程。

2．判断算法优劣的两个尺度是： 时间复杂度和空间复杂度 。

3．二分搜索算法只能应用于在 有序（升序或降序） 序列中搜索指定的关键值。

4．蛮力法的特点是： 穷举问题所有的状态，判断是否是问题的解 。

5．举两个用变治法解决的典型问题： 高斯消去法，AVL平衡树，堆 。

6. 背包问题可以用 蛮力法、动态规划法 方法解决。

7. NP问题是一类可以用不确定多项式算法求解的判定问题。

8．举出三个NP完全问题：哈密尔顿回路，旅行商问题，划分问题，装箱问题，图的着色问题。

9．分支限界法包含三个基本过程： 穷举所有状态、广度搜索，剪枝 。

10．随机算法分成两大类： Las Vegas 算法和Monte Carlo算法 。

|  |  |
| --- | --- |
| 得分 |  |

　　　　［评分标准：每个空２分，给出一个答案给１分］

# 选择（共10分，每小题2分。把最恰当的答案题号填入括号内）

1．下面的函数的时间复杂度为O(*n*3)是（ B ）。

A、3*n*+log log *n* B、(*n*2+*n*)\*(*n*+log *n*2)

C、*n*!/(*n*+2) D、78*n*3-2n+*n*\*log *n*

2．二分搜索算法的时间复杂度为（ C ）。

A、 O(*n*2 ) B、O(*n*3) C、O(log *n*) D、O(*n*\*log *n*)

3．堆排序的时间复杂度为（ D ）。

A、 O(*n*2 ) B、O(*n*3) C、O(log *n*) D、O(*n*\*log *n*)

4．下列解决装箱问题的四个启发式方法中，需要先排序的方法是：( B,D )

A、 FF B、BFD C、BF D、FFD

5. 单源点最短路径问题，常用的解决算法是（ B ）

A、Prim算法 B、Dijkstra算法

C、Kruskal 算法 D、Flody算法

［评分标准：每个选择２分，第４题选对一个给１分］

|  |  |
| --- | --- |
| 得分 |  |

# 图解题(四选三，每题10分，共30分)

1. 利用Prim算法求出下图的最小生成树。



解：［评分标准：每步给２分，相邻步骤之间正确给１分］



1. 利用Dijkstra算法，求出下图以A为单源点出发的所有最短距离，要求给出计算步骤。



解：［评分标准：每步给２分，相邻步骤之间正确给１分］

|  |  |
| --- | --- |
| 树中的顶点变化 | 图中余下未处理的顶点 |
| a(-,0) | b(a,3),c(-,∞),d(a, 7 ),e(-,∞) |
| a(-,0),b(a,3) | c(b, 3+4 ),d(b, 3+2 ),  e(-,∞ ) |
| a(-,0), b(a,3),d(b,5) | c(b, 7 ),e(d, 5+4 ) |
| a(-,0), b(a,3),  d(b,5),c(b,7) | e(d, 9 ) |

1. 给出下图的有向图，应用基于DFS的算法来解决拓扑排序问题。



解：［评分标准：搜索步骤４分，标记３分，边分类３分］



d->a->c->b->g->ef

1. 给定道路图，在所有起点到终点的路径中找一条长度最短的路径，通过图示给出分析过程。



解：

1. 第一步确定任何到终点的最短路径，令表示从到终点的最短路径

记作u,2

记作d,3

…



1. 第二步确定从任何到终点的最短路径

, u,5

…



1. 第三步，第四步



|  |  |
| --- | --- |
| 得分 |  |

# 问答题（八选四，每题10分，共40分）

1. 简述Horspool算法的思想，并填充如下表（10分）：

已知待查找的模式为：BARBER

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 字符c | A | B | C | D | E | F | … | R | … | Z | \_ |
| 移动距离t(c) | 4 | 2 | 6 | 6 | 1 | 6 | 6 | 3 | 6 | 6 | 6 |

解：［评分标准：每一步骤１分］

1. 对于给定的长度为m的模式和在模式及文本中用到的字母表，按照一定规则求出构造表。
2. 将模式与文本在开始处对齐。
3. 从模式的第一个字符开始，比较模式和文本中的相应字符，直到：要么所有m个字符都匹配，要么遇到了一对不匹配的字符。对于后一种情况，如果c 是当前文本中和模式的最后一个字符相对齐的字符，从移动表的第c列中取出单元格t(c)的值，然后将模式沿着文本向右移动t(c)个字符的距离。

|  |  |
| --- | --- |
| 得分 |  |

1. 利用Graham扫描法求凸包问题。（10分）

解：[评分标准：文字描述或图示１０分]

设p1=(x1,y1),p2=(x2,y2)是平面上的两个点，如果x1<=x2 且y1<=y2,则称p2支配p1,记为p1<p2。

设S是平面中的一个点集，点p ∈S是极大点或最大点，如果不存在点　q∈ S ,使得p≠q并且p<q。

算法：把S中的所有点按照它们x坐标的非升序排列，最右点（有最大x值的那点）无疑是个最大点。从右到左扫描这些点，同时确定它是否在y坐标上被先前扫描过的任何点所支配。

输入：平面上n个点的集合S.

输出：S中极大点集合M。

1. 设A为S中按x坐标非生序排列的点的集合。如果两个点有相同的x坐标，则有较大y坐标值的点出现在集合的前面。
2. M={A[1]}
3. Maxy=A［１］的y坐标值。
4. For j=2 to n
5. (x,y)=a[j]
6. if y>maxy then
7. M=M U {A[j]}
8. maxy=y
9. endif
10. endfor
11. 用动态规划算法求解矩阵连乘问题：计算5个矩阵连乘所需要的最少次数。（10分）

|  |  |
| --- | --- |
| 得分 |  |

这5个矩阵是：

M1:5 × 10 , M2:10 × 4, M3:4 × 6, M4:6 × 10, M5:10 × 2

解：［评分标准：计算步骤５分，表格３分，最优乘法顺序２分］

用c[s,t]表示Ms\*…\*Mt的最小的乘法次数。则：

c[1,1]=0,c[2,2]=0,c[3,3]=0,c[4,4]=0,c[5,5]=0

c[1,2]=5×10×4 =200,

c[2,3]= 10×4×6 =240,

c[3,4]= 4×6×10 =240,

c[4,5]= 6×10×2 =120

c[1,3]=min{c[1,2]+ 5×4×6,c[2,3]+ 5×10×6 }=320

c[2,4]=min{c[2,3]+ 10×6×10,c[3,4]+ 10×4×10 }=640

c[3,5]=min{c[3,4]+ 4×10×2,c[4,5]+ 4×6×2 }=168

c[1,4]=min{ c[1,2]+ c[3,4]+5×4×2,c[1,3]+ 5×6×2,c[2,4]+ 5×6×2}=620

c[2,5]=min{c[2,4]+ 5×6×2,c[2,3]+c[4,5]+ 5×6×2,c[3,5]+ 5×6×2}=248

c[1,5]=min{c[1,2]+c[3,5] 5×6×2,c[1,3]+c[4,5]+ 5×6×2,c[3,5]+ 5×6×2}=348

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| C[1,1]=0 | C[1,2]=200 | C[1,3]=320 | C[1,4]=620 | C[1,5]=348 |
|  | C[2,2]=0 | C[2,3]=240 | C[2,4]=640 | C[2,5]=248 |
|  |  | C[3,3]=0 | C[3,4]=240 | C[3,5]=168 |
|  |  |  | C[4,4]=0 | C[4,5]=120 |
|  |  |  |  | C[5,5]=0 |

最优的乘法顺序为：

M1× (M2× (M3× (M4×M5))

1. 描述利用回溯法求解3着色问题。（10分）

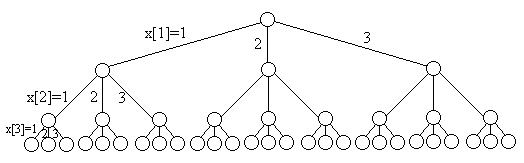
|  |  |
| --- | --- |
| 得分 |  |

解：［评分标准：说明部分５分，算法思想５分］

状态空间--- 穷举设计,求解过程--- 深度优先搜索+剪枝

状态空间--- 穷举设计

n个顶点的图有3n 个可能的着色方案，建立搜索树。

n=3时的搜索树

输入：无向图G=(V,E)

输出：G的顶点的3着色，其中每个c[j]为1，2，3

1. for k=1 to n

2. c[k]=0

3. end for

4. flag= flase

5. k=1

6. while k>=1

7. while c[k]<=2

8. c[k]=c[k]+1

9. if c 为合法着色 then set flag =true 且退出

10. else if c 是部分解 then k=k+1 {前进}

11. end while

12. c[k]=0

13. k=k-1 {回溯,就是剪枝}

14. end while

15. if flag then output c

16. else output “no solution”

1. 二分查找：在一个排好序的数组T[1..n]中查找x，如果x在T中，输出x在T的下标j；如果x不在T中，输出j=0.给出算法的伪代码，并给出算法的复杂度和证明。（10分）：

解：

|  |  |
| --- | --- |
| 得分 |  |

#include "stdio.h"

#include "math.h"

#define n 10

int BinarySearch(int T[],int x)

{

int l=1,r=n;

while(l<=r)

{

int m=ceil((l+r)/2.0);

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

else if (T[m]>x) r=m-1;

else l=m+1;

}

return 0;

}

int main(void)

{

int T[n]={1,3,5,7,8,10,11,15,18,19};

int x=5;

int r=BinarySearch(T,x);

printf("%d",r);

getchar();

return 0;

}

复杂度：

检索次数满足下述递推方程：

1. ，每次n为偶数

当时，

1. ，每次n为奇数

，递推式

，第一次展开

，第二次展开

当时，

因此，

，即 (\*)

e.g.,

则，由（1）和（2）得

，即

证明：数学归纳法

1. 当
2. 假设成立，则当n+1时

，由归纳假设得，

令，，由(\*)得

，即

再由(\*)得

因此，由得

，得证。

1. 装载问题



给出轻者先装的算法，并证明对于任何正整数k，该算法对k个集装箱的实例得到最优解。（10分）：

|  |  |
| --- | --- |
| 得分 |  |

解：

1. 对集装箱重量排序
2. I={1}
3. W=w1
4. for j=2 to n do
5. if W+wi<=C
6. then W=W+ wi
7. I=I+{j}
8. else return I,W

证明：（数学归纳法）

（1）k=1,只有1个集装箱，其重量小于C。任何装法都只有各种方式，因此都是最优解，因此轻者先装也是最优解。

（2）归纳假设：假设算法对于规模为k的输入都能得到最优解。考虑规模为k+1的输入，是集装箱重量，。

从N中拿掉最轻的集装箱，得到：



根据归纳假设，对于k个输入，N’、W’、C’的最优解为I’，令



那么I必然是N的最优解，这也是算法对于N,W,C的解。

证明：I必然是N的最优解，采用反证法。假设I不是N的最优解。则必然存在最优解I\*，如果I\*中没有1，用1替代I\*中的第一个集装箱标号得到的解也是最优解（个数不变，因此也是最优解），使得I\*为包含1的关于N的最优解，且|I\*|>|I|。那么I\*-{I}是关于N’、W’、C’的最优解（N’、W’、C’不包含1）且



与I’的最优性矛盾（最优解I\*-{I}大于最优解I’）。

1. 分治法：设有n=2k个运动员要进行循环赛，请设计一个满足以下三点要求的比赛日程表。提示：可将比赛日程表设计成一个 n 行n列的二维表，其中，第i行第j列表示第i个选手在第j-1天比赛的选手，并且i和j都属于[1,n]。

①每个选手必须与其他n-1名选手比赛各一次；

②每个选手一天至多只能赛一次；

③循环赛要在最短时间内完成。

（1）简述用分治策略解决此问题的思路；

（2）当n=23=8时，请通过自底向上的迭代过程手工画出循环赛日程表，并简要说明解题思路。（10分）：

|  |  |
| --- | --- |
| 得分 |  |

解： **【评分标准：每小题各5分，其中最后的赛程表占3分】**

（1）可将所有参赛的选手分为两部分，n＝2k个选手的比赛日程表就可以通过为n/2＝2k-1个选手设计的比赛日程表来决定。递归地执行这种分割，直到只剩下2个选手时，比赛日程表的制定就变得很简单：只要让这2个选手进行比赛就可以了。

（2）解题思路：把求解2k个选手比赛日程问题划分成依次求解21、22、…、2k个选手的比赛日程问题，换言之，2k个选手的比赛日程是在2k-1个选手的比赛日程的基础上通过迭代的方法求得的。在每次迭代中，将问题划分为4部分：

（a）左上角：左上角为2k-1个选手在前半程的比赛日程；

（b）左下角：左下角为另2k-1个选手在前半程的比赛日程，表中每个单元上的数字可由左上角相应位置上的数字加2k-1得到；

（c）右上角：右上角为前2k-1个选手在后半程的比赛日程，由左上角加2k-1得到；所以，等同于直接将左下角数字抄到右上角；

（d）右下角：右下角为后2k-1个选手在后半程的比赛日程，表中每个单元上的数字可由左下角相应位置上的数字减去2k-1得到；所以，等同于将左上角直接抄到右下角；

图表 1 完整赛程表

1. 动态规划：针对用如下邻接权重矩阵所表示的有向图，利用Floyd算法求解完全最短路径。（共10分）

# （1）简要描述算法思想；（5分）

# （2）给出由邻接矩阵变换到距离矩阵的过程。（5分）

|  |  |
| --- | --- |
| 得分 |  |

|  |  |  |  |
| --- | --- | --- | --- |
| 0 | 5 | 10 | 10 |
| ∞ | 0 | 4 | 7 |
| 10 | ∞ | 0 | 7 |
| 12 | ∞ | ∞ | 0 |

图表 2 图的邻接矩阵

解：【评分标准：第(1)小题根据表述准确性酌情给分；第（2）小题每一步正确得2分】

1. 算法思想：设图中顶点数为n，所有顶点的编号分别为1至n号。考虑任意两个顶点（设编号分别为i，j）间的最短距离路径，它**可能经过的中间顶点集合（记为S）**中最大的顶点编号记为k。

当S为空集，也即k=0时，表示i、j之间无任何中间顶点时，原始邻接矩阵（记为D(0)，其每个元素记为d0（i,j））记录了任意i、j点之间的当前最短距离；当k=1时，我们用矩阵D(1)中的每个元素d1（i,j）来记录当S中最大顶点编号为1时，任意一对顶点i、j之间的当前最短距离。依此类推，矩阵D(n)即为我们希望解得的最终距离矩阵。

根据最短距离的最优子结构性质，我们可以得到如下从D（k-1）到D(k)之间的递推关系：对于任意k，当i、j之间的最短距离为dk(i,j)=min{dk-1(i,k) + dk-1(k,j), dk-1(i,j)}。也即若最短路径经过k，则dk(i,j)必等于前者。否则取后者。

（2）从矩阵D(0)开始，经过4步，求得的最后一个矩阵即为该图的距离矩阵：

|  |  |  |  |
| --- | --- | --- | --- |
| 0 | 5 | 10 | 10 |
| ∞ | 0 | 4 | 7 |
| 10 | ∞ | 0 | 7 |
| 12 | ∞ | ∞ | 0 |

表 1 D（0）

|  |  |  |  |
| --- | --- | --- | --- |
| 0 | 5 | 10 | 10 |
| ∞ | 0 | 4 | 7 |
| 10 | **15** | 0 | 7 |
| 12 | 17 | 22 | 0 |

表 2 D（1）

|  |  |  |  |
| --- | --- | --- | --- |
| 0 | 5 | **9** | 10 |
| ∞ | 0 | 4 | 7 |
| 10 | 15 | 0 | 7 |
| 12 | 17 | **21** | 0 |

表 3 D（2）

|  |  |  |  |
| --- | --- | --- | --- |
| 0 | 5 | 9 | 10 |
| **14** | 0 | 4 | 7 |
| 10 | 15 | 0 | 7 |
| 12 | 17 | 21 | 0 |

表 4 D（3）

|  |  |  |  |
| --- | --- | --- | --- |
| 0 | 5 | 9 | 10 |
| 14 | 0 | 4 | 7 |
| 10 | 15 | 0 | 7 |
| 12 | 17 | 21 | 0 |

表 5 D（4）