**第九十十一章练习参考答案**

第9章

1.答：

Int selection(int \*s,double \*P,int n) { //赌轮法选择,设s[i]整数

double wheel\_pos;partsum=0;

int i=0

wheel\_pos= frandom(); //赌轮位置[0-1]

partsum=0;

do{

partsum=partsum+p[i]

i++

} while (partsum<wheel\_pos && i<n);

return s[i-1]; }

（这可看做是一个舍伍得算法。舍伍得算法其它练习如：给定问题的编码策略，求一个随机初值。如0/1背包遗传算法，解=(x1,x2,..xn)，xi∈{0,1},请随机生成一个初始解。）

2.解：

从1-365个整数中依次随机抓取25个，其可能的排列数为365.364…341=365!/340!；如果允许重复，即一个数抓出来后，再放回去继续用，则有36525种排列，二者之比为R=365!/(340!.36525)。可用随机选取25个1-365间的数，求其中没有重复数字排列所占比例R来近似计算。

double R()

{

double r;

int p[25],i,j,L,t

int m,n //总抓取次数、有重复数字次数

L=1000000;

t=0 //控制提前跳出for循环

do while m<L

{

for (i=1 to 25) {

P[i]=random(365)+1 //随机产生1-365之间的整数

for( j=1 to i-1){

if p[j]=p[i] {

n=n+1； //本组出现重复数字

t=1

break;

}

if (t=1) break;

}

m=m+1;

t=0;

}

Return (double)(m-n)/m；

}

3.解：修改第7章分枝限界算法，四个方向随机选择一个未被标记的方向扩展：以下是Las Vegas算法：(需重复调用本程序或成功，或达到一定次数返回不可达结果。第二问，修改while循环条件，可控制仅随机完成若干段布线，后边接队列式分枝限界算法完成后面的布线，参考第九章n皇后问题，及第七章电路板布线问题算法。（略）)

bool FindPath(Position start, Position finish,

int& PathLen, Position \* &path)

{ //计算从起点位置start到目标位置

//finish的最短布线路径.找到最短布

//线路径则返回true,否则返回false

if((start.row= =finish.row) &&

(start.col= =finish.col)

{PathLen=0; return true;} //start=finish

//设置方格阵列“围墙”

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

grid[0][i]=grid[n+1][i]=1;

//顶部和底部

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

grid[i][0]=grid[i][m+1]=1;

//左翼和右翼

Position offset[4]; //初始化相对位移

offset[0].row=0; offset[0].col=1;//右

offset[1].row=1; offset[1].col=0;//下

offset[2].row=0; offset[2].col=-1;//左

offset[3].row=-1; offset[3].col=0;//上

int NumOfNbrs=4;//相邻方格数

Position here, nbr;

here.row=start.row;

here.col=start.col;

grid[start.row][start.col]=2;

//标记可达方格位置

do { //标记相邻可达方格

count=0；

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

nbr.row=here.row + offset[i].row;

nbr.col=here.col+offset[i].col;

if(grid[nbr.row][nbr.col]==0)

y[count++] = i; **//该方格未被标记，记录该方向。不再用队列。**

}

if(count > 0) { **// 有布线位置** i= y[rnd.Random(count)]; **// 随机选一个方向**

nbr.row=here.row + offset[i].row;

nbr.col=here.col+offset[i].col;

grid[nbr.row][nbr.col]

=grid[here.row][here.col]+1;

}

else return //无进一步扩展位置，随机布线失败

if((nbr.row= =finish.row) &&

(nbr.col= =finish.col)) break; //若到达目标位置跳出do

//从本节点再扩展

here.row= nbr.row

here.col= nbr.col

}while(true);

//构造最短布线路径

PathLen=grid[finish.row][finish.col]-2;

path=new Position[PathLen];

//从目标位置finish开始向起始位置回溯

here=finish;

for(int j=PathLen-1; j>=0; j--){

path[j]=here; //找前驱位置

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

nbr.row=here.row+offset[i].row;

nbr.col=here.col+offset[i].col;

if(grid[nbr.row][nbr.col]==j+2)

break;

}

here=nbr;//向前移动

}

return true;

}

4.答：

(1)Las Vegas算法不会得到不正确的解。( √ )

(2)Monte Carlo算法不会得到不正确的解。( ×)

(3) Las Vegas算法总能求得一个解。(× )

(4) Monte Carlo算法总能求得一个解。(√ )

5.答：1-(1-δ)k

6.答: (2)。（反之，偏真的,答案(1)）

7.答：

（1）一般情况下，无法有效判定Las Vegas算法所得解是否肯定正确。（ × ）

（2）一般情况下，无法有效判定Monte Carlo算法所得解是否肯定正确。（√ ）

（3）虽然在某些步骤引入随机选择，但**Sherwood算法**总能求得问题的一个解，且所求得的解总是正确的。 (√ )

（4）虽然在某些步骤引入随机选择，但**Sherwood算法**总能求得问题的一个解，但一般情况下，无法有效判定所求得的解是否正确。 (×)

第10章

1. 证明：

当FF(I)=1时，显然FF(I)=OPT(I)。下面设FF(I)>1，记w=。因为任何两只箱子的重量之和大于B（否则不会开新箱子，可装到一个箱子里），因此，当FF(I)为偶数时，W>B×FF(I)/2；当FF(I)为奇数时，设最重的箱子重量为B1，则有W>B×(FF(I)-1)/2+B1>B×FF(I)/2。故总有FF(I)<2W/B。又显然OPT(I)≥W/B，得FF(I)<2OPT(I)。

1. 证明：

根据算法，每一个顶点关联的割边数大于等于关联的非割边数，对所有的顶点求和，每条边出现2次，故所有的割边数大于等于所有非割边数。从而MCUT(I)≥|E|/2。又显然OPT(I)≤|E|，得证OPT(I)≤2MCUT(I)。

1. 解：

递推公式：B(0)=0,

B(i)=B(i-1)∪{t|t-ti∈B(i-1),ti≤D},i=1,2,…n。

显然，OPT(I)=。

算法DP

输入：n个作业的处理时间t[1..n]

DP(t[]){

D=; B(0)=0;

for i=1 to n{

B(i):=B(i-1);

for t=t[i] to D

if( t-t[i]∈B(i-1)) B(i)=B(i)∪{t}；

}

t:=max B(n);J=φ;

}

输出-t ； //最优解

for (i=n to 1 step -1){

if t-t[i]∈B(i-1){

J=J∪{i};

t=t-t[i]

if( t<=0) break;

}

}

输出 J, {1,2,…n}\J；//解集合I1、I2

}

DP的时间复杂度为T(n)=O(nD)=O(n2tmax)，这是伪多项式时间算法。

以此为基础，设计近似算法：

FPTAS(t[],ε){

tmax=max{t[i]|i=1,2,…n} ;

b=max{⎣tmax/(1+1/ε)n⎦,1} ;

for(i=1 to n) t/[i]=⎡t[i]/b⎤ ;

DP(t/[]) //以t/[1..n]调用算法DP

}

FPTAS的时间复杂度T(n)=O(n2t/max)=O(n2tmax/b)=O(n3(1+1/ε))。

下面分析其近似性能。

当b=1时，FPTAS得到最优解。不妨设b>1，b(t/[i]-1)<t[i]≤bt/[i]。对任意S⊆{1,2,…,n}， ，则

 （\*）

记最优解J\*，FPTAS的近似解J,j/={1,2…,n}-J，

OPT(I)=，FPTAS(I)= ，于是

FPTAS(I)-OPT(I)= -

=(-)+(-)+(-)

由(\*)式，第一项小于等于0，J/是关于t/[1..n]的最优解，第二项也小于等于0，又显然FPTAS(I)≥tmax ,于是

FPTAS(I)-OPT(I)≤-≤bn

≤tmax/(1+1/ε)≤FPTAS(I)/(1+1/ε)

化简即得FPTAS(I)≤(1+ε)OPT(I)，得证FPTAS是完全多项式时间近似方案。

（类似0/1背包问题，但本例求最小。另外，第5章中所谓优化动态规划算法，可仿本例定义V[i]={v|v=,含义为前i项物品任意装法所得收益值的集合，按其递推公式设计的动态规划算法就是那个算法，可能更容易按动态规划思想理解。）

1. 看ppt
2. 答：

(1)旅行商问题存在多项式时间近似方案。( × )

(2)0/1背包问题存在多项式时间近似方案。( √ )

(3) 0/1背包问题的贪心算法(单位价值高优先装入)是绝对近似算法。(× )

(4)多机调度问题的贪心近似算法(按输入顺序将作业分配给当前最小负载机器)是ε-近似算法。( √ )

第11章

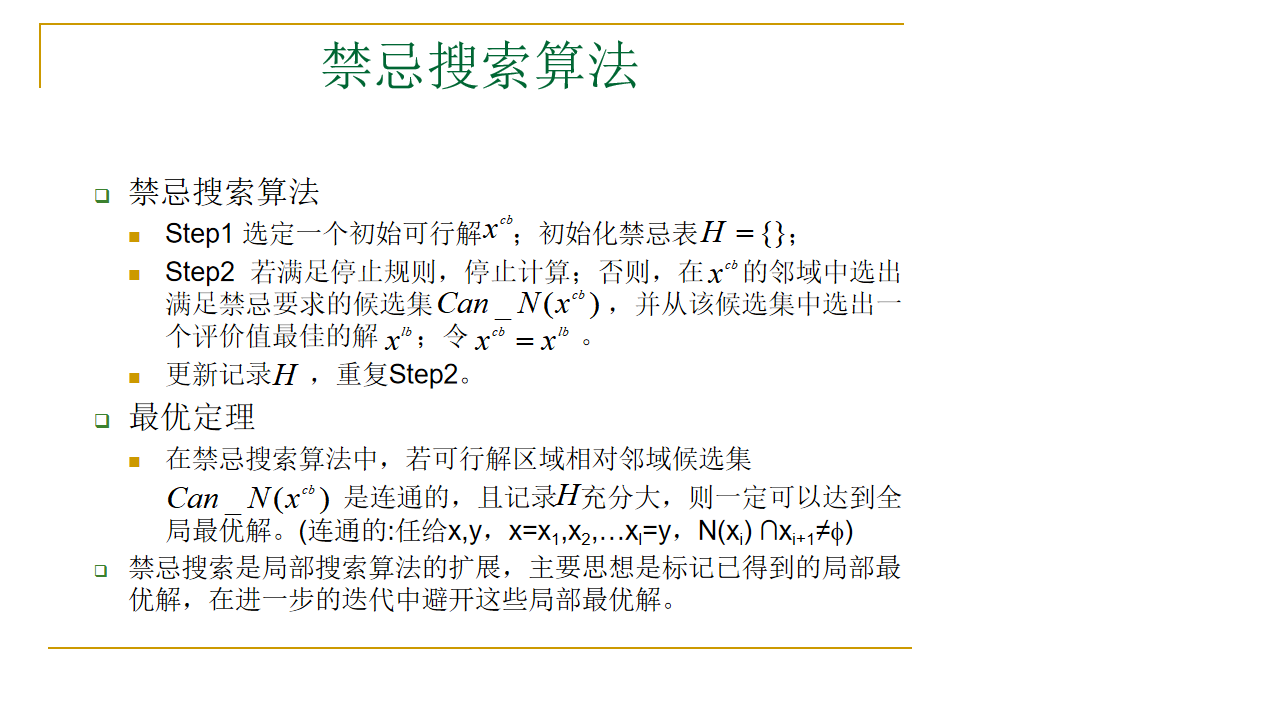
1. 解：

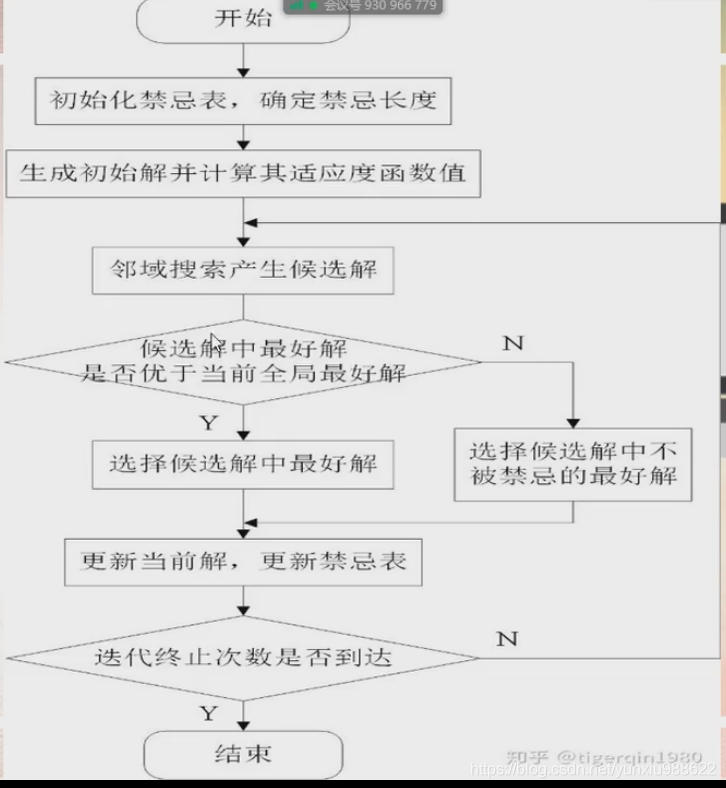
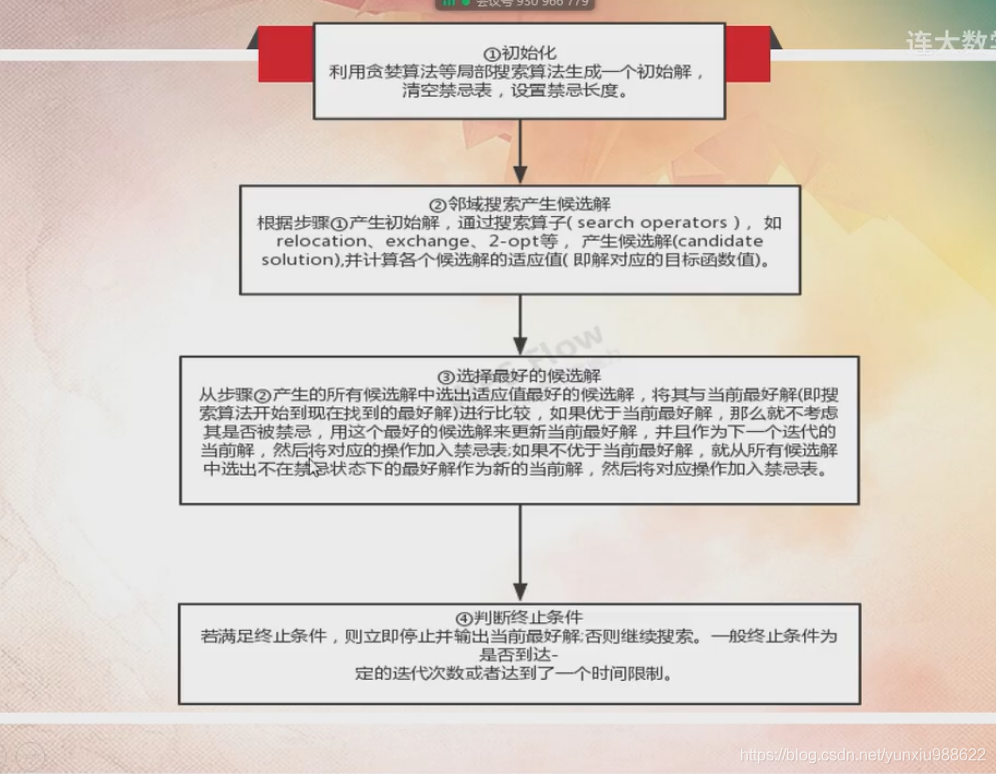
N(s)={(1324),(2134),(4123),(3214),(3421),(3142)}

1. 解：

N(x)={01001,10001,11101,11011,11000}。

1. 看ppt



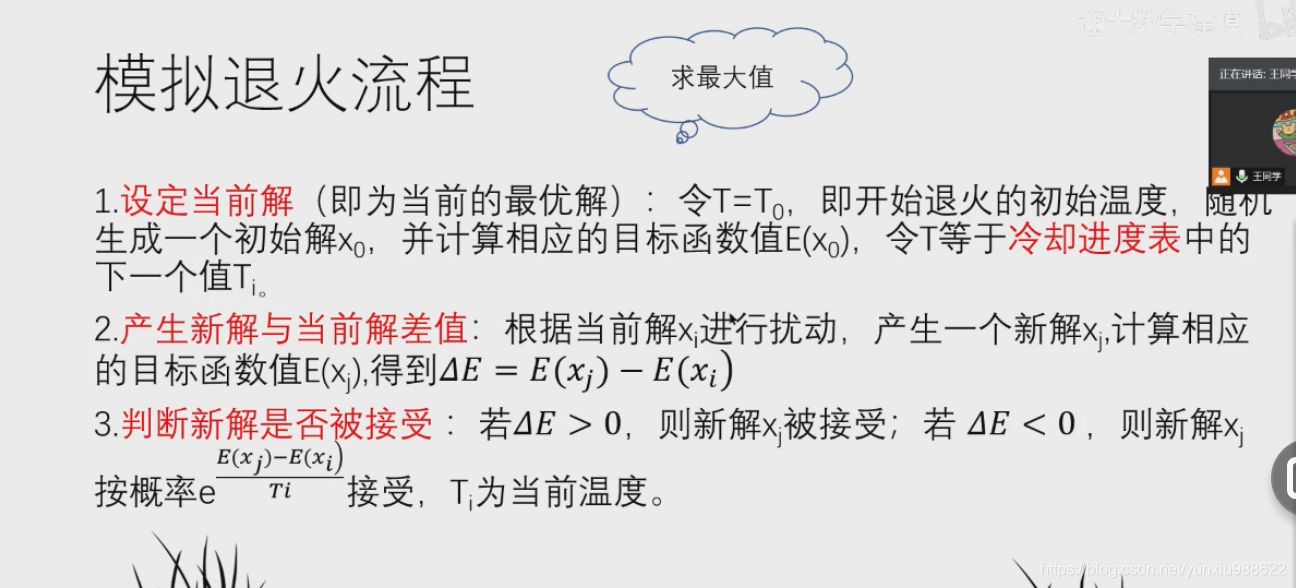
1. 解：影响力大的对象是指改变它，可以导致目标值发生较大幅度的变化(可能变坏也可能变好)。如0/1背包问题，物品重量大的对象影响力大，特赦后可以腾出较大背包容量。
2. 解：

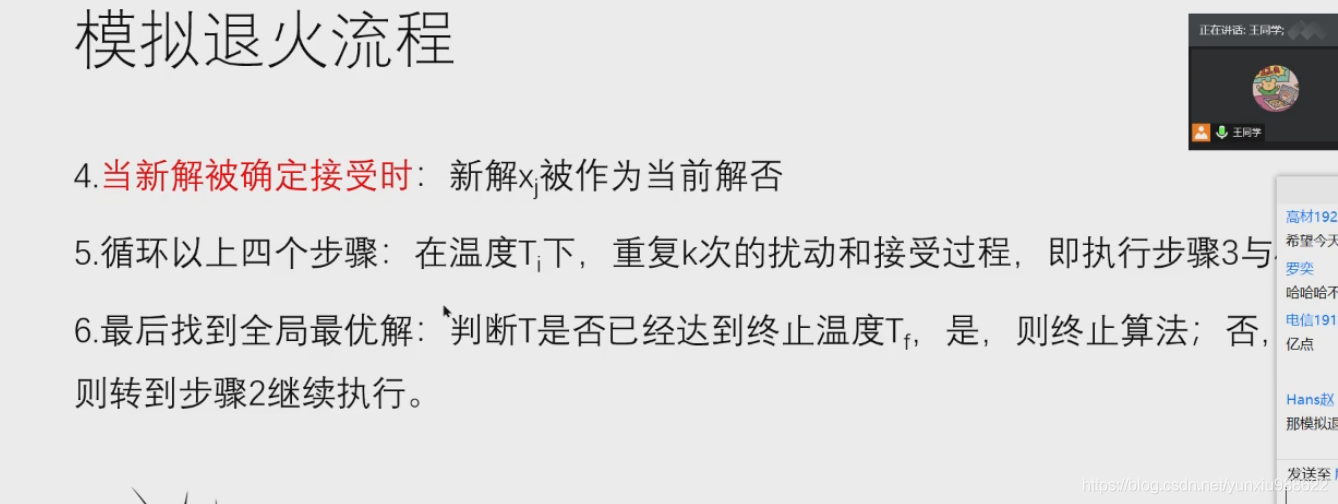
(1)禁忌搜索中，禁忌某些对象是为了避免邻域中的不可行解。( × )

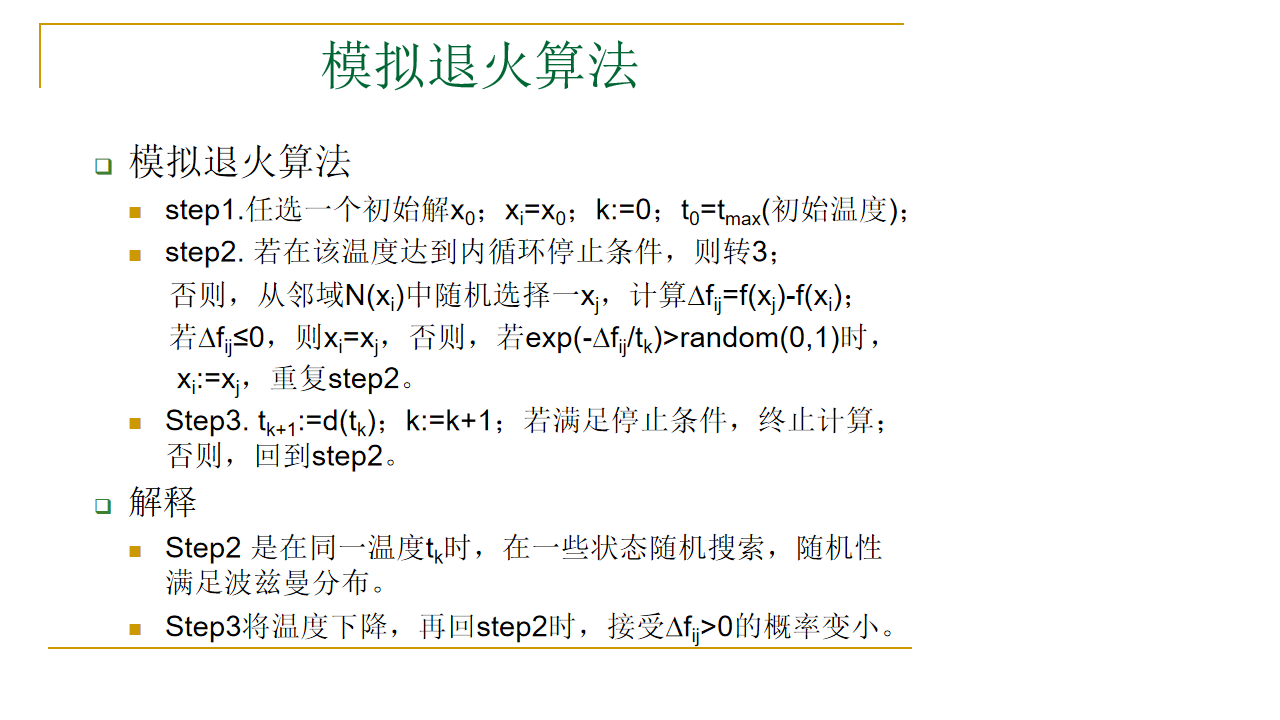
(2)禁忌长度越大越好。( × )

(3)禁忌长度越小越好。( × )

6.看ppt







7.答：tk越高接受的概率越大，Δfij越小接受退步解的概率越大。

8.答：(1)(2)(3)

9.答：排序适应函数：



线性加速适应函数：fitness(x)=αf(x)+βf(x)等。

10．简单交配可能产生非解编码(染色体)。解决办法:改变交配规则，如交配位后基因按异方基因顺序选取不重复基因、不变位法等。

11.下面属于遗传算法实现的关键技术问题的有\_\_\_\_\_\_\_\_\_\_\_。

(1)解的编码 (2)初始种群的选择 (3)邻域定义 (4)适应函数

答：（1）（4）

第8章 补充练习

1.下面说法，正确的是：\_\_\_\_\_\_1、3、\_\_\_\_\_\_.

(1)P类问题是存在多项式时间算法的问题。

(2)NP类问题是不存在多项式时间算法的问题。

(3)P类问题一定也是NP类问题。

(4)NP类问题比P类问题难解。（错误 因为子集关系？）

2.下面说法，正确的是：\_\_\_\_2\_\_\_\_\_\_.

(1)P⊂NP (2)P⊆NP (3)P=NP (4)P≠NP

3.下面说法，正确的是：\_\_2、3、4\_\_\_\_\_\_\_.

(1)NP-难问题是NP中最难的问题（NP难问题不是NP问题）

(2)NP-完全问题是NP中最难的问题

(3)NP-难不比任何NP问题容易

(4)NP-完全问题也是NP-难问题。（√）

参考答案

1.(1)(3) 2.(2) 3.(2)(3)(4)