吉林大学

第四章 分治法作业

1、求解①和②情况下的下列递归关系式，并简单分析：

$$T(n)=\begin{cases} g(n) & n\text{足够小} \\ 2T(n/2)+f(n) & \text{否则} \end{cases}$$

① $g(n)=O(1)$和$f(n)=O(n)$;

② $g(n)=O(1)$和$f(n)=O(1)$。

①当$g(n)=O(1)$和$f(n)=O(n)$时，

不妨设$g(n)=a$，$f(n)=bn$，a、b为常数，
令$n=2^k$，则：

$$
\begin{aligned}
T(n) &= 2T(n/2)+bn \\
&= 4T(n/4)+2bn \\
&= \cdots\ \cdots \\
&= 2^kT(n/2^k)+kbn \\
&= an+bn\log_2 n = O(n\log_2 n)
\end{aligned}
$$

②当$g(n)=O(1)$和$f(n)=O(1)$时，

不妨设$g(n)=c$，$f(n)=d$，令$n=2^k$，则：

$$
\begin{aligned}
T(n) &= 2T(n/2)+d \\
&= 4T(n/4)+2d \\
&= \cdots \quad \cdots \\
&= 2^kT(n/2^k)+kd \\
&= cn+d\log_2 n \\
&= O(n)
\end{aligned}
$$

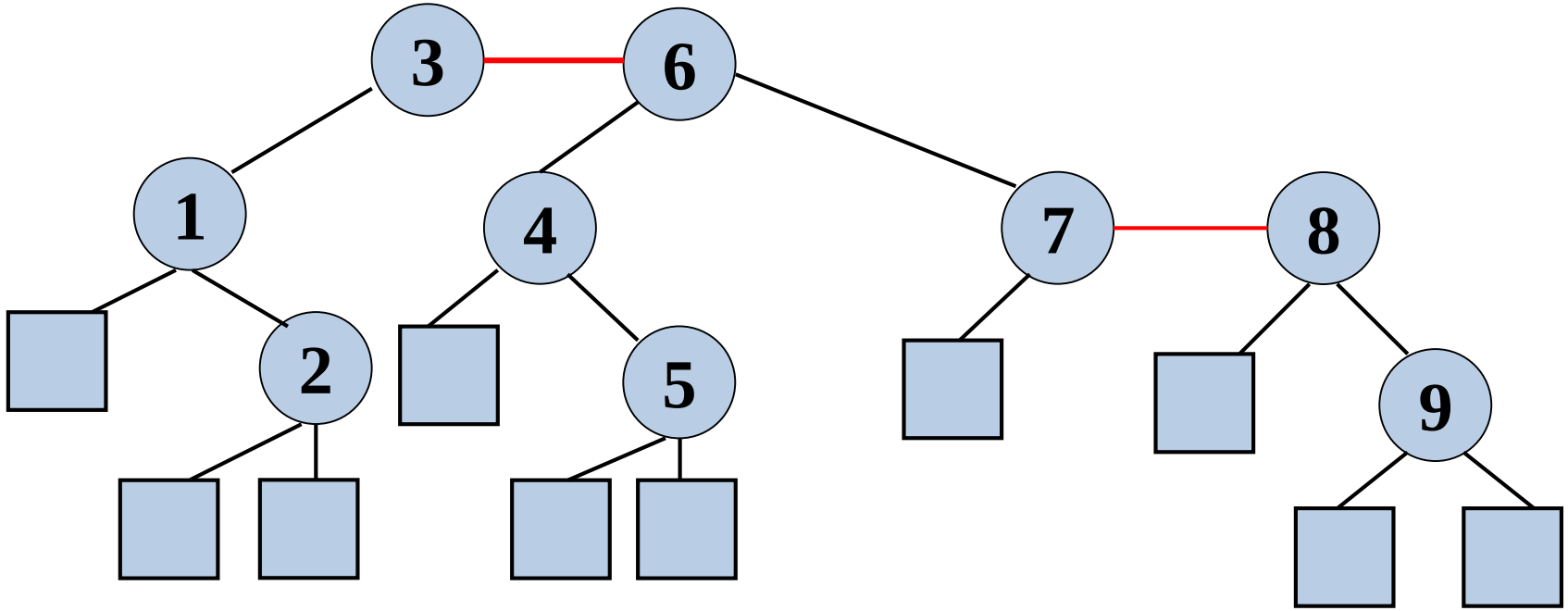
2、根据4.2节开始所给出的二分查找策略，写一个二分查找的递归过程。

```
Procedure BINSRCH2(A, low, high, x, j)
  integer  mid ;
  if low≤high then
       mid← ⌊(low+high)/2 ⌋
       case
            : x=A(mid):   j←mid;  return
            : x>A(mid):   BINSRCH2(A, mid+1, high, x, j)
            : x<A(mid):   BINSRCH2(A, low, mid–1, x, j)
       endcase
  else
        j←0;
  endif
end BINSRCH2
```

初次调用为BINSRCH2(A, 1, *n*, *x*, *j*)

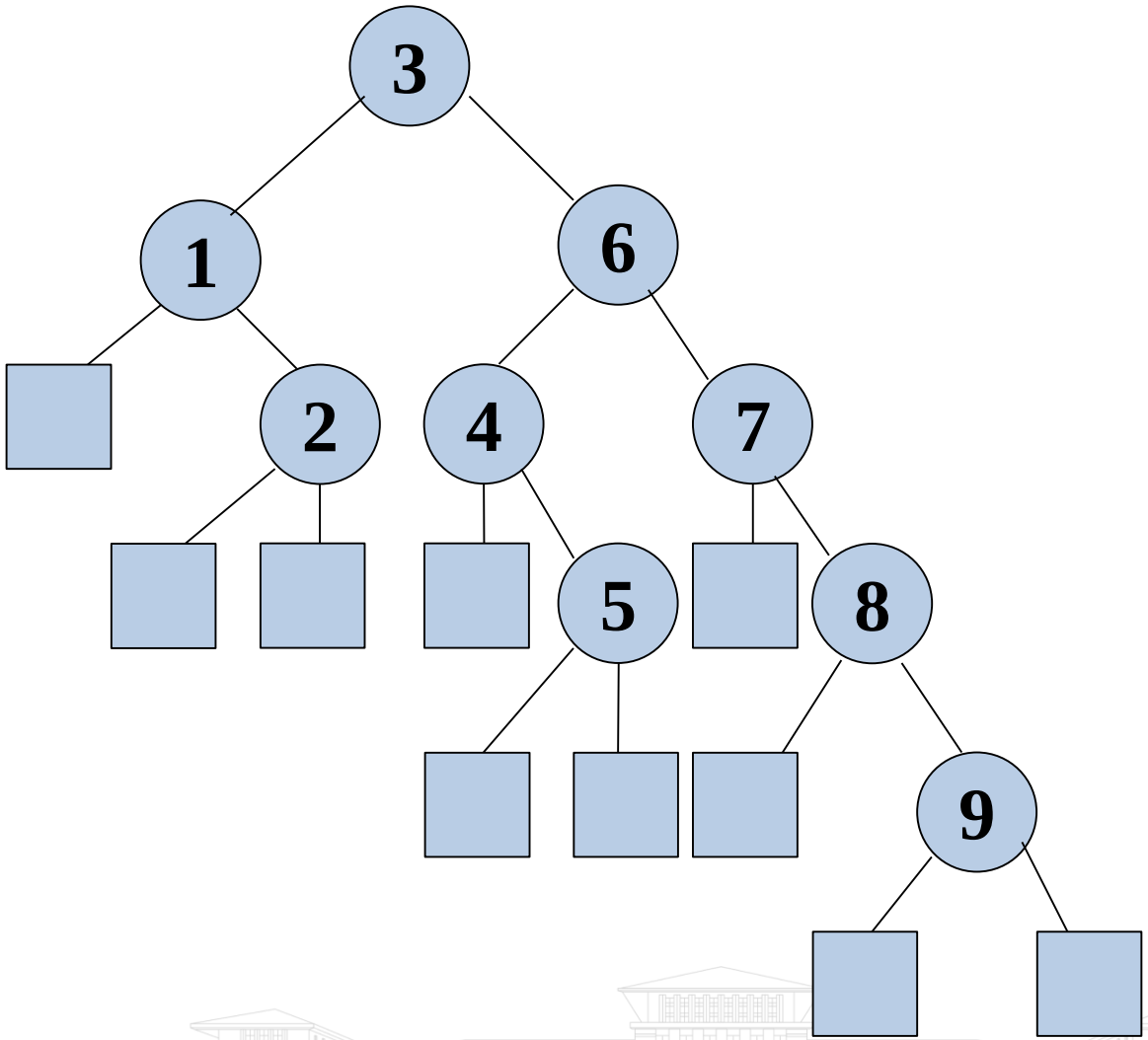
3、作一个“三分”查找算法，它首先检查$n/3$处的元素是否等于某个x的值，然后检查$2n/3$处的元素。这样，或者找到x，或者把集合缩小到原来的1/3。分析算法在各种情况下的计算复杂度。

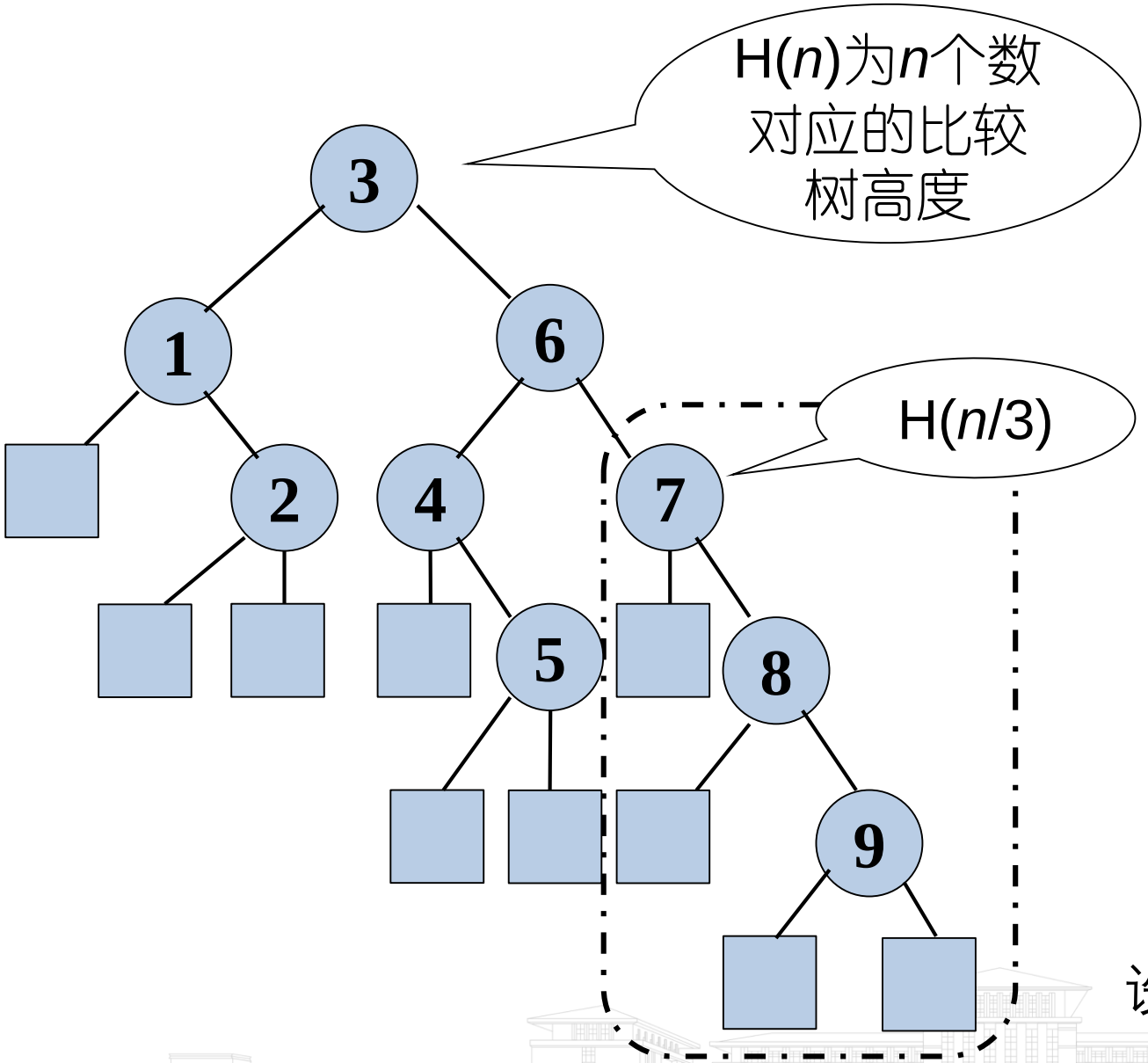
```
Procedure ThriSearch(A, n, x, j)
  integer low, high, p1, p2
  low←1;  high←n
  while low≤high do
       p1← ⌊(high+2low)/3⌋
       p2← ⌊(2high+low)/3⌋
      case
           :x=A(p1):  j←p1; return
           :x=A(p2):  j←p2; return
           :x<A(p1):  high←p1–1
           :x>A(p2):  low←p2+1
           :else:     low←p1+1;  high←p2–1
       endcase
  repeat
  j←0
end ThriSearch
```



Low	P1	P2	High
1	3	6	9
1	1	1	2
2	2	2	2

Low	P1	P2	High
7	7	8	9
9	9	9	9


3
1
6
2
4
7
5
8
9



$H(n)=H(n/3)+2$

$H(3)=3$

$H(2)=2$

$H(1)=1$

$$H(n)=\begin{cases} a & n\text{足够小} \\ H(n/3)+2 & \text{否则} \end{cases}$$

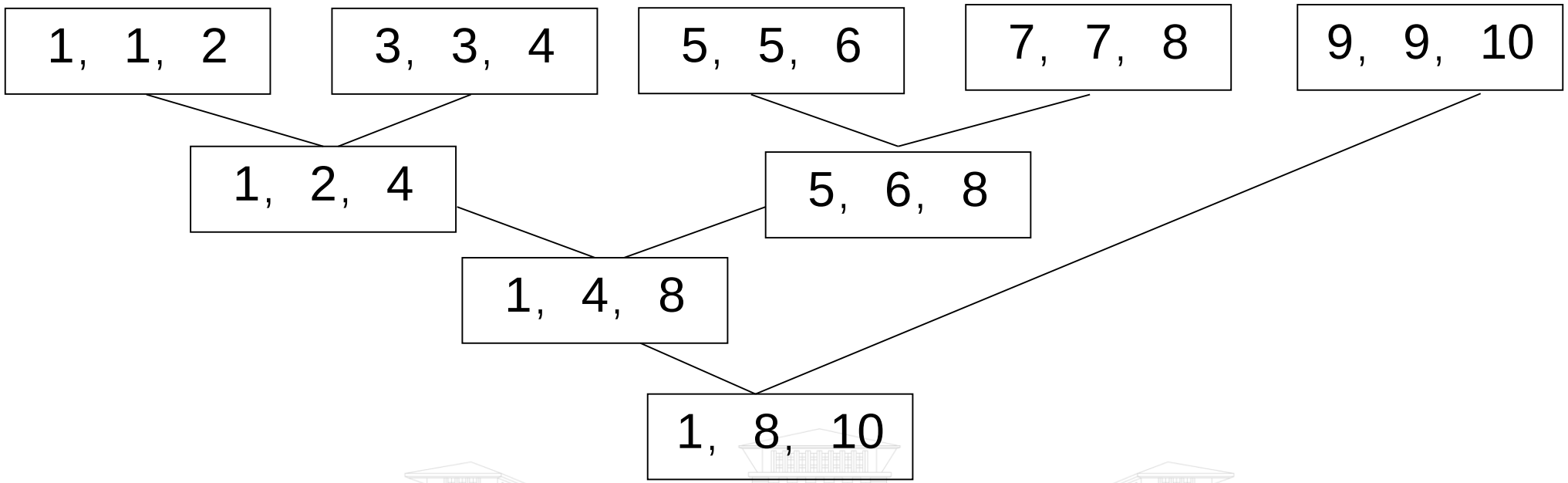
设$n=3^k$，求得$f(n)=f(3^k)=1+2\log_3 n$

三分查找的时间复杂度

成功情况下		
最好	最坏	平均
O(1)	O($\log_3(n)$)	O($\log_3(n)$)
失败情况下		
最好	最坏	平均
O($\log_3(n)$)	O($\log_3(n)$)	O($\log_3(n)$)

4、写一个“由底向上”的归并分类算法，从而取消对栈空间的利用。

实例：1，2，3，4，5，6，7，8，9，10



```
Procedure MERGESORT1(n)
  integer low, mid, high, step, k
  step ←1; k← ⌊logn⌋
  while step ≤2^k do
      low ← 1
      while low + step –1<n do
            mid ← low + step –1;
               high ← min{ mid+step, n};
          MERGE (low, mid, high);
          low ←high+1;
        repeat
       step ←step×2;
    repeat
end MERGESORT1
```

5、采用分治策略求解二维极大点问题，给出下面实例的求解过程及结果：（1,2）（2,8）（3,5）（4,1）（5,4）（6,7）（7,6）（8,3）。

理解算法，理解算法执行过程；

理解递归执行过程及执行结果。

首先将所有点按照x值排序（如果x值相等再按y值排序）

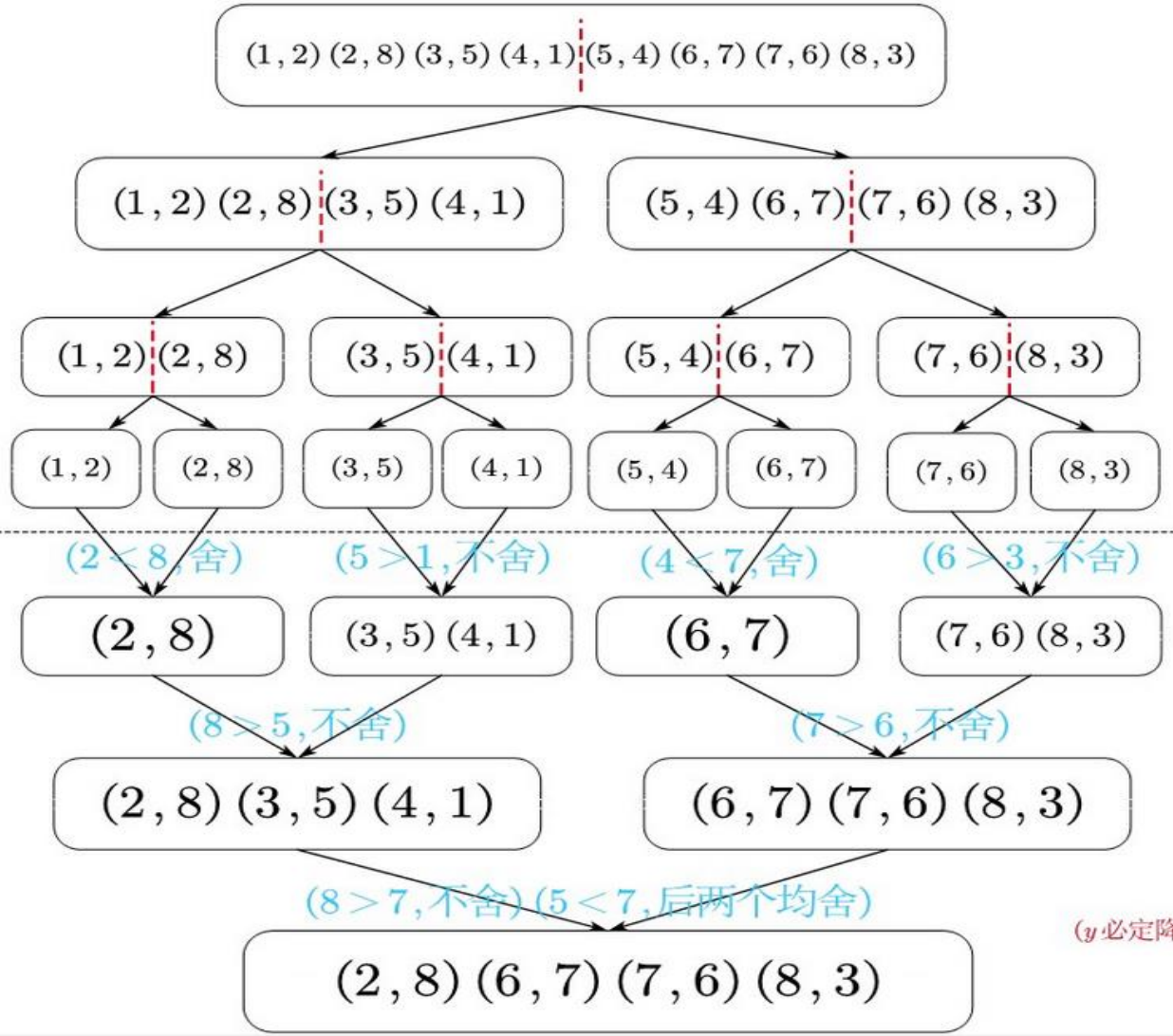
可得如下结果：（1,2）（2,8）（3,5）（4,1）（5,4）（6,7）（7,6）（8,3）

分：找出垂直于x轴的中位线l，将整个点集分为两个子集S_L和S_R；

治：分别找出S_L和S_R的极大点；

合：对于S_L中的极大点p，如果p_y小于S_R中任一极大点y的值，则p被支配，舍弃掉。

据此可得到如下过程：



设极大点集按照x值从小到大排序，则y值一定从大到小排序。

合并时：

1. 设S_R中y值最大的极大点为q，q距离中位线最近。
2. 在S_L的极大点集内查找满足$y_p>y_q$的最末元素p，删掉S_L中排在p后的极大点，合并完成。

注：查找过程可以基于y值二分查找。

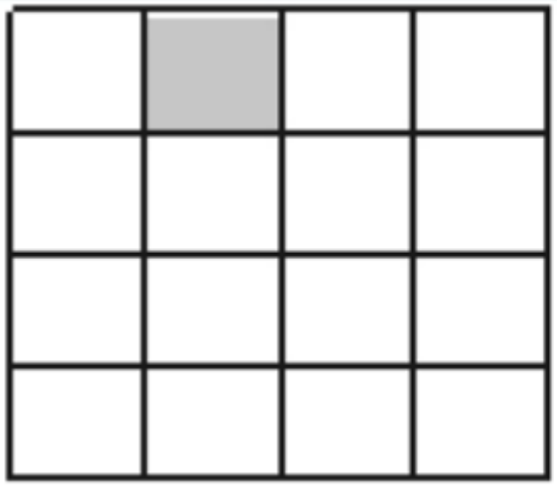
6、N个硬币放袋子里，其中一枚是假币，并且伪造的硬币比真币轻，设计一个算法找到那枚假币。要求写明算法思想，并给出算法描述（注意书写规范）。

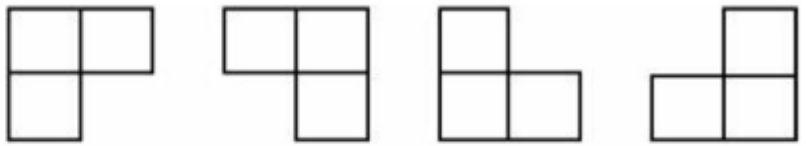
算法思想：N如果是偶数，就分为两半，每半N/2枚，称重较轻的继续二分，直到每半只有一枚，较轻者为假币；
N如果是奇数，拿出一枚，分为两半，若不一样重，就继续选轻的二分，若一样重，则拿出来的那枚就是假币。

```
procedure FAKE(c)
   if sizeof(c)==1 then return c;  //递归出口
   if sizeof(c)%2==0
       then
            (c1,c2)=partition(c);
             lighter=cmp(c1,c2);
             FAKE(lighter);
             if sizeof(lighter)=1, then return lighter; endif  //递归出口
    else   c0=takeone(c);
            c=takeoneRemain(c);
            (c1,c2)=partition(c);
            if cmp(c1,c2)==0  then return  c0;  //递归出口
                 else lighter=cmp(c1,c2);    FAKE(lighter);  endif
       endif
end
```

时间复杂度分析：T(n)=T(n/2)+1,求得T(n)=O(logn)

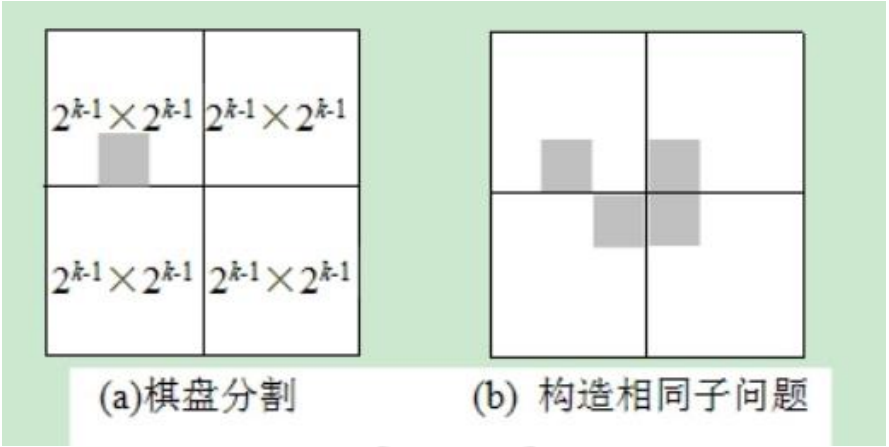
7、有一个$2^k \times 2^k$的方格棋盘，恰有一个方格是黑色的，其他为白色。你的任务是用包含3个方格的L型牌覆盖所有白色方格，且黑色方格不能被覆盖，任意一个白色方格不能同时被两个或更多牌覆盖。如图所示为L型牌的4种旋转方式。请设计算法，解决棋盘覆盖问题。要求：写清算法思路，感兴趣的同学鼓励上机编程实现。





算法思路

- 采用二分法解决该问题， $n=2^k$ ，则二分时问题规模从k降低到k-1，棋盘从2^k降低到2^{k-1}大小的四个子棋盘。用L形牌覆盖不包含黑格的三个子棋盘，使得子问题与原问题具有相同特征，问题转化为对四个包含黑格的子棋盘的覆盖问题。当问题规模减小到k=1时，覆盖策略直接可解；否则递归上述过程。



棋盘覆盖问题中的数据结构设计

- 棋盘：全局二维数组board(n,n)表示棋盘，$n=2^k$。初始时，board(i,j)=-1表示黑格，否则为0。
- 棋盘范围：基于棋盘左上角坐标+棋盘大小来表达，变量tr表示行坐标，变量tc表示列坐标，变量s表示棋盘大小。初始时，tr=1，tc=1，s=n。
- L型骨牌：全局变量t表示当前用到第t张L型骨牌，初值为1，一个n×n的棋盘中有一个黑格，所以，t最大值为$(n^2-1)/3$。

本章作业结束