1. Parallel merging

假设数组A长度为m，数组B长度为n，那么并行地调用m+n次dual binary search，从A，B中找到第1，2，3 ... m+n个元素，再将这m+n个元素聚集到一个数组中，就完成了排序。

而这个算法的Work和Depth是：

W(n)=nW(1)+O(logn)

D(n)=D(1)+O(logn)

得到 W(n)=O(n), D(n)=O(logn).

2. Parentheses matching

分治法：将原长n的string分成两个长n/2的string，并求两个string中除了可以匹配的左右括号对之外，余下的左/右括号个数。同时如果原字符串可以成功配，则在每一层子问题中，最左边的子问题左括号一定比右括号多，最右边的子问题右括号一定比左括号多。

将括号string表示为数组, "(" 为-1， ")"为1，那么算法如下：

```
//array is the array of parenthese, n is the length of the array,
//ifLeft&ifRight is the mark of the most left/right sub question
match(array, n, isLeft = 1, isRight = 1){
    if(n = 1){
        return array[0];
        //if (, return -1, if ) return 1
    }
    int leftValue=match(array[0, n/2], n/2, isLeft, 0);
    int rightValue=match(array[n/2+1, n], n/2, 0, isRight);

    if((isLeft && leftValue>0)&&(isRight && rightValue<0))
        exit;

    return leftValue + rightValue;
}
```

若返回0则匹配成功

3. Efficient implementation for parallel quicksort

1. 设置多个x，用多个分治取代两个分治，降低深度
2. 用x划分数组时，将数组分成若干段，并行地进行划分，最后再merge到一起
3. 选择能让划分均匀的x