数据结构第二次作业报告

王政和 17212010075

2.1

$\frac{2}{n}$, 37, $\sqrt{n}$, n, $nloglogn$, $2nlogn$, $nlog^2n$, $n^{1.5}$, n^2, n^2logn, n^3, $2^{\frac{n}{2}}$

2.5

$f(n) = cos(n)$
$g(n) = sin(n)$

2.7

(3)

$$\sum_{i=0}^{n}\sum_{j=0}^{n^2}1 = O(n^3)$$

(5)

$$\sum_{i=0}^{n}\sum_{j=0}^{i^2}\sum_{k=0}^{j}1 = \sum_{i=0}^{n}\sum_{j=0}^{i^2}j = \sum_{i=0}^{n}O(i^4) = O(n^5)$$

2.12

$O(n)$

不放设$T(n) = cn$，那么有$T(100) = 0.5ms$，当$n = 1.2 \times 10^7$时，$T(n) = 1min$

$O(nlogn)$

同理，$n \approx 3.656807360103117 \times 10^6$

$O(n^2)$

同理，$n \approx 34641$

$O(n^3)$

同理，$n \approx 4932$

2.26

先回答问题：

- 递归在当前数组长度小于等于2时结束
- 考虑两种情况：
  - 如果前$n-1$个数中已经有 `Majority` 了，那么第n个选不选无所谓
  - 如果前$n-1$个数中没有 `Majority` ，那么第n个可能带来“希望”
  - 所以n为奇数时，总是把第n个入选
- 递推为$T(n) = T(\frac{n}{2}) + O(n)$，所以是$O(n)$的算法
- 原算法需要构造$log(n)$个新数组，实际上只要留一份原来的数组，另外的新数组可以反复用同一个数组

但是题目的想法未免太过“递归”，我写的代码基于一个非常简单的思路：因为这个数字出现次数超过了一半，所以只要计数即可，遍历数组，相同加一，不同减一，减到0的时候更换候选者，次数怎么都会是正的（存在的话）。这个算法只需要$O(1)$的额外空间。

```
public void solve(int[] a) {
    int n = a.size();
    if (n <= 1) return; // ignore n<=1
    int count = 1;
    int m = a[0];
    for (int i = 0; i < n; ++i) {
        if (count == 0) {
            m = a[i];
            count = 1;
        } else if (m == a[i]) {
            ++count;
        } else {
            --count;
        }
    }
    int check = 0;
    for (int i = 0; i < n; ++i) {
        if (m == a[i])
            ++check;
    }
    if (check > (n/2))
        System.out.printf("Majority element is %d\n", m);
    else
        System.out.printf("Not found\n");
}
```

2.23

放弃提前结束的机会即可……

```
int binarySearch(AnyType[] a, AnyType x) {
    int low = 0, high = a.length - 1;
    while (low < high) {
        int mid = (low + high) / 2;
        if(a[mid].compareTo(x) < 0 )
            low = mid + 1;
        else
            high = mid;
    }
    if (a[low].compareTo(x) == 0)
        return low;
    return NOT_FOUND;
}
```