# 算法第一次作业

170617 17373126 刘萱

## 一、

1.

$$T(1) = T(2) = 1$$
$$T(n) = T(n-2) + n \quad if \quad n > 2$$

```
T(n) = T(n-2) + n
     = T(n-4) + (n-2) + n
     = T(n-6) + (n-4) + (n-2) + n
     = ...
     = T(2) + 4 + 6 + ... + n
     = 1 + 4 + 6 + ... + n
     = n² - (3n²/4 - n + 2)
     <= n²
Thus  T(n) = O(n²)
```

2.

$$T(1) = 1$$
$$T(n) = 4T(n/2) + n \quad if \quad n > 1$$

```
T(n) = 4T(n/2) + n
     = 16T(n/4) + 4(n/2) + n
     = ...
     = n(1 + 2 + 4 + 8 + ... + 2^(logn-1))
     <= n²
Thus T(n) = O(n²)
```

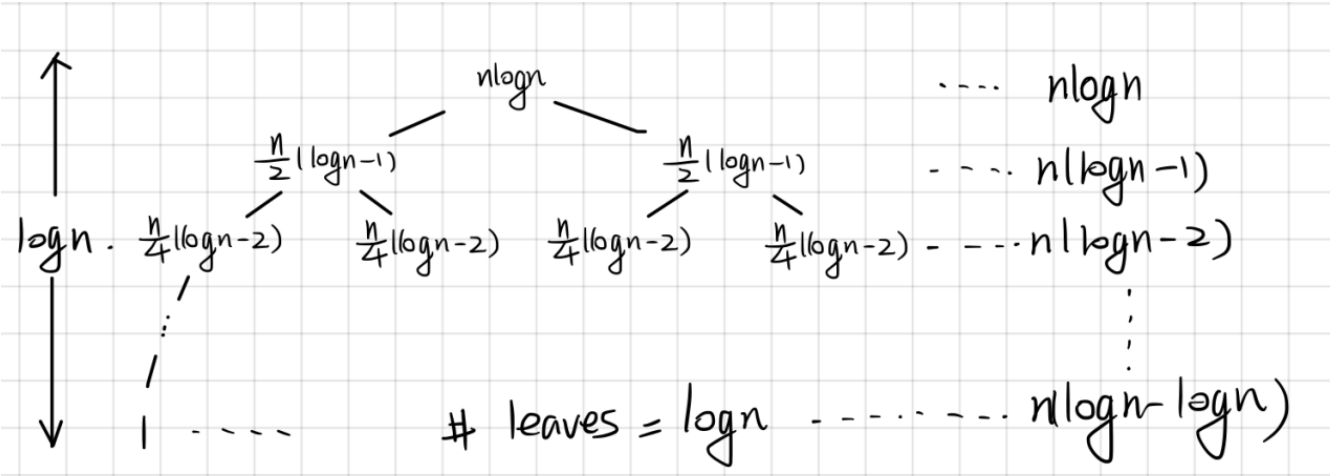
3.

$$T(1) = 1$$
$$T(n) = 2T(n/2) + n \quad if \quad n > 1$$

```
T(n) = 2T(n/2) + n
     = 2T(n/4) + 2(n/2) + n
     = 2T(n/8) + 4(n/4) + n + n
     = ...
     = nlogn
Thus T(n) = O(nlogn)
```

4.

$$T(1) = 1$$
$$T(n) = 2T(n/2) + n \log n \quad if \quad n > 1$$




```
T(n) = 2T(n/2) + nlogn
     = 2T(n/4) + n(logn - 1) + nlogn
     = 2T(n/4) + n(logn - 2) + n(logn - 1) + nlogn
     = ...
     = n(logn)² - n(1 + 2 + ... +  (logn-1))
     <= 2nlogn
Thus T(n) = O(nlogn)
```

5.

$$T(1) = 1$$
$$T(n) = 2T(n/2) + n^2 \quad if \quad n > 1$$

```
T(n) = 2T(n/2) + n²
     = 2T(n/4) + 2(n/2)² + n²
     = 2T(n/8) + 2²(n/4)² + n²/2 + n²
     = ...
     = n²(1 + 1/2 + (1/2)² + ... + (1/2)^logn)
     = 2n²
Thus T(n) = O(n²)
```

6.

$$T(1) = 1$$
$$T(n) = 3T(n/2) + n \quad if \quad n > 1$$

```
T(n) = 3T(n/2) + n
     = 3T(n/4) + 3(n/2) + n
     = 3T(n/8) + (3/2)²n + 3n/2 + n
     = ...
     = n(1 + 3/2 + (3/2)² + ... + (3/2)^(logn-1))
     <= n + (n^log3)
Thus T(n) = O(n^log3)
```

7.

$$T(1) = 1$$
$$T(n) = T(n/2) + n\log n \quad if \quad n > 1$$

```
T(n) = T(n/2) + nlogn
     = T(n/4) + n/2(logn - 1) + nlogn
     = T(n/8) + n/4(logn - 2) + n/2(logn - 1) + nlogn
     = ...
     = logn(n + n/2 + n/4 + ... + n/n)  - (n/2 + 2n/4 + 3n/8 + ... + nlogn/n)
     <= logn(nlogn/2 + n)
Thus T(n) = O(n(logn)²)
```

# 二、k路归并问题

现有 $k$ 个有序数组（从小到大排序），每个数组中包含 $n$ 个元素。你的任务是将他们合并成 1 个包含 $kn$ 个元素的有序数组。首先来回忆一下课上讲的归并排序算法，它提供了一种合并有序数组的算法 $Merge$。如果我们有两个有序数组的大小分别为 $x$ 和 $y$，$Merge$ 算法可以用 $O(x+y)$ 的时间来合并这两个数组。

1. 如果我们应用 $Merge$ 算法先合并第一个和第二个数组，然后由合并后的数组与第三个合并，再与第四个合并，直到合并完 $k$ 个数组。请分析这种合并策略的时间复杂度（请用关于 $k$ 和 $n$ 的函数表示）。(9 分)
2. 针对本题的任务，请给出一个更高效的算法，并分析它的时间复杂度。(提示：此题若取得满分，所设计算法的时间复杂度应为 $O(nk\log k)$)。(10 分)

1.

```
T(n) = O(n+n) + O(2n+n) + ... + O((k-1)n+n)
     = O(2n) + O(3n) + ... + O(kn)
     = O((k-1)(k+2)n/2)
```

2.

使用一个最小堆做k路归并排序，对于k个有序数组（升序排列）

1.建堆，分别取k个数组的第一个值建立最小堆 `O(k)`

2.取出堆顶元素，即为最小元素 `O(1)`

3.若堆顶元素所在数组不为空，取下一个元素放在堆顶，调整最小堆 `O(logk)`

若堆顶元素所在数组为空，则删除最小堆的堆顶元素，最小堆`heapSize--` `O(logk)`

重复2、3直到所有序列为空（`heapSize`为0）

总的时间复杂度为：`O(k) + O(nklogk) = O(nklogk)`

# 三、战线补给问题

`Devide(A,p,r)`

**Input**：An array **A**, the length of A **l**

**Output**：Amount of replenishment

```
sum <- 0;
L = r-p+1;
if L >= 2 then
    q <- p + L/2;
    sum1 <- Devide(A,p,q);
    sum2 <- Devide(A,q+1,r);
    if sum1 == A && sum2 == A then
        sum <- A;
    end
    else
        sum <- sum1 + sum2;
    return sum;
end
else then
    if A[p] == 0 then
        return A;
    end
    else
        return A[p]*B;
end
```

`Supply(A,n,l)`

**Input**：An array **Arr** indicates which fortress each soldier is in, the number of soldiers **n**,the index of the fortress `l`

**Output**：Amount of replenishment

```
A[] <- {0}; // arr is number of people per fortress
for i<-0 to n-1 do
    A[Arr[i]] <- A[Arr[i]] + 1;
end
sum <- Devide(A,0,2^l-1);
return sum;
```

时间复杂度：`T(n) = O( n + 2l²)`

# 四、区间计数问题

给定一个包含 $n$ 个元素的数组 $A=[a_1,a_2,\cdots,a_n]$。对数组 $A$ 中的任意区间 $[l,r](1\leq l\leq r\leq n)$，该区间的和可表示为 $S_{[l,r]}=\sum_{i=l}^{r}a_i$。
请设计一个高效的分治算法统计有多少个区间 $[l,r]$ 满足：$X\leq S_{[l,r]}\leq Y$（$X,Y$ 为给定的常数）。并分析该算法的时间复杂度。

`MergeSort(sum,lower,upper,low,high)`

**Input** : An array `sum`, lower bound **lower**,upper bound **upper**,index **low**,index **high**

**Output** : the number of interval sums

```
if high - low <= 1 then
    return 0;
end
mid <-(low + high)/2;
m <- mid;
n <- mid;
count <- 0;
count <- MergeSort(sum,lowwer,upper,low,mid) + MergeSort(sum,lower,upper,mid,high);
for i<-low to mid-1 do
    while m < high && sum[m] - sum[i] < lower do
        m++;
    end
    while n < high && sum[n] - sum[i] <= upper do
        n++;
    end
    count <- count + n - m;
end
return count;
```

`Count(num,lower,upper)`

**Input** : An array `num` contains n number,lower bound **lower**,upper bound **upper**

**Output** : the number of interval sums

```
sum <- 0;
for i<-0 to n-1 do
    sum <- sum + num[i]
    sum[i] = sum;
end
return MergeSort(sum, lower, upper, 0, n+1);
```

时间复杂度： `T(n) = O(nlogn)`

# 五、向量的最小和问题

将所有的n个向量映射到第一象限，问题可简化为求n个点中距离最小的两个间距

采用分治的思想，把n个点按照x坐标进行排序，以坐标`mid`为界限分成左右两个部分，对左右两个部分分别求最近点对的距离，然后进行合并。对于两个部分求得的最近距离`d`，合并过程中应当检查宽为`2d`的带状区间是否有两个点分属于两个集合而且距离小于`d`.

`cmpx(v1,v2)`

```
return v1.x < v2.x;
```

`cmpy(v1,v2)`

```
return v1.y < v2.y;
```

`min(a,b)`

```
return a < b ? a : b;
```

`dis(v1,v2)`

```
return sqrt((v1.x-v2.x)^2 + (v1.y - v2.y)^2);
```

`deal(p,q)`

```
tail <- 0;//计数变量
mid <- (p+q)/2;
d <- min(deal(p,mid),deal(mid+1,q));
for i<-mid to p && arr[mid].x - arr[i].x < d do
    rarr[tail++] <- arr[i];
end
for i<-mid+1 to q && arr[i].x - arr[mid].x < d do
    rarr[tail++] <- arr[i];
end
sort(br,br+tail,cmpy);
for i<-0 to tail-1 do
    for j<-i+1 to tail-1 && rarr[j].y-rarr[i].y<d do
        if d>dis(rarr[i],rarr[j]) then
            d=min(d,dis(br[i],br[j]));
        end
    end
end
return d;
```

`count(arr,n)`

```
sort(arr,arr+n,cmpx);
d <- deal(0,n);
return d;
```

时间复杂度：`T(n) = O(nlogn)`

讨论同学：刘丽君，潘林煜，唐璐