# Homework1

刘云飞

BY1706126

## 1 第一题

### 1.1 题目描述

已知下列递推公式：

$$C(n)=\begin{cases}1 & n=1\\ 2C\left(\frac{n}{2}\right)+n-1 & n\geq 2\end{cases} \tag{1}$$

请由定理1导出C(n)的非递归表达式并指出其渐进复杂性。

**定理1**：设$a$,$c$为非负整数，$b$,$d$,$x$为非负常数，并对于某个非负整数k, 令$n=c^k$, 则以下递推式：

$$f(n)=\begin{cases}d & n=1\\ a*f\left(\frac{n}{c}\right)+b*n^x & n\geq 2\end{cases} \tag{2}$$

的解是：

$$f(n)=\begin{cases}b*n^x\log_c n+d*n^x & a=c^x\\ \left(d+\frac{b*c^x}{a-c^x}\right)*n^{\log_c a}-\left(\frac{b*c^x}{a-c^x}\right)*n^x & a\neq c^x\end{cases} \tag{3}$$

### 1.2 解决方法

根据公式1与定理1对比，令$F(n)=C(n)-1$则有：

$$F(n)=\begin{cases}0 & n=1\\ 2*C(\frac{n}{2})+n-2 & n\geq 2\end{cases} \tag{4}$$

整理有如下结果：

$$F(n) = \begin{cases} 0 & n = 1 \\ 2 * F(\frac{n}{2}) + n & n \geq 2 \end{cases} \tag{5}$$

根据定理1可以得到下面对照：$d = 0, a = 2, c = 2, b = 1, x = 1, a = cx$

故$F(n) = n * \log_2(n)$，所以$C(n) = F(n) + 1 = n * \log_2(n) + 1$ 所以$C(n)$的渐进复杂度为$O(n * log_2(n))$。

# 2 第二题

## 2.1 问题描述

由于Prim算法和Kruskal 算法设计思路的不同，导致了其对不同问题实例的效率对比关系的不同。请简要论述：

- 如何将两种算法集成，以适应问题的不同实例输入；
- 你如何评价这一集成的意义？

## 2.2 解决方法

### 2.2.1 最小生成树

在一给定的无向图$G = (V, E)$ 中，$(V, E)$ 代表连接顶点$u$ 与顶点$v$ 的边（即），而$w(V, E)$ 代表此边的权重，若存在$T$ 为$E$ 的子集（即）且为无循环图，使得的$w(T)$最小，则此$T$ 为$G$ 的最小生成树。最小生成树其实是最小权重生成树的简称。

### 2.2.2 两个算法时间效率分析

**Prim算法分析：**使用邻接矩阵来保存图的话，时间复杂度是$O(V^2)$，使用二叉堆优化Prim算法的时间复杂度为$O((V + E) * \log(V)) = O(E \log(E))$ **Kruskal算法分析：**时间复杂度为$O(E * \log(E))$

通过数据对比两者结果如下图（图片来着网络）： 不考虑中间的二叉堆优化Prim算法，仅仅比较Prim与Kruskal算法，有如下结论

- Prim在稠密图中比Kruskal优
- Prim在稀疏图中比Kruskal劣

因此考虑判别稀疏图与稠密图的边界来分类使用两算法来优化最小生成树算法。

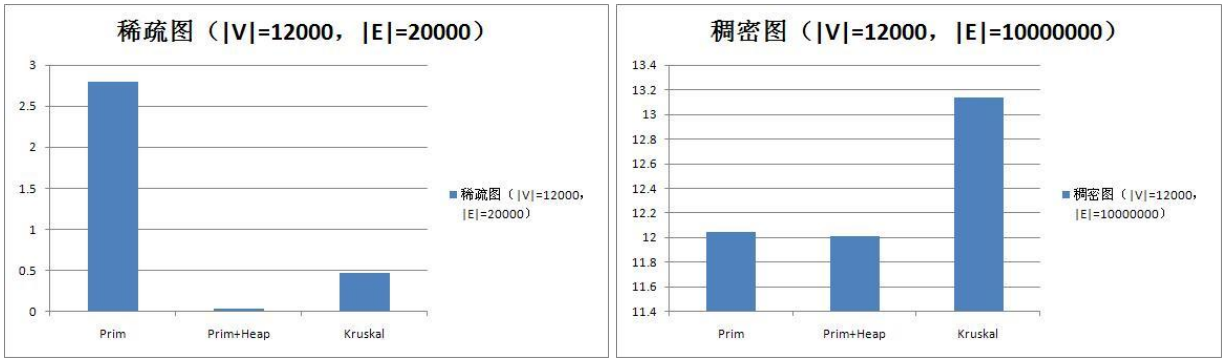


图 1: 三个算法时间复杂图比较

### 2.2.3 边界计算与优化之后的伪代码

$$v^2 = e * \log(e) \tag{6}$$

其中$v, e$分别代表$V, E$的元素个数。因此伪代码考虑如下：

**Algorithm 1** Minimal Span Tree's algorithm

1: **function** MINSPANTREE($V, E$) ▷ 优化最小生成树算法

**Require:** 无向图$(V, E)$

**Ensure:** 最小生成树$T$

2: **if** $v^2 \leq e * \log(e)$ **then**

3: $T \leftarrow$ Prim($V, E$)

4: **else**

5: $T \leftarrow$ Kruskal($V, E$)

6: **end if**

7: **end function**

### 2.2.4 对算法集成的评价

通过将两个算法进行集成，虽然增加了代码量，但是针对不同稀疏程度的无向图可以保证其最小生成树的时间控制在比较优的结果，此时时间复杂度为：$\min(V^2, E * \log(E))$。总之，没有一个算法是万能的，可以兼顾所有，此方法也是采用具体情况具体分析的方法来解决问题。

# 3 第三题

## 3.1 问题描述

分析以下生成排列算法的正确性和时间效率：

**Algorithm 2** Heap Permute's algorithm

```
1: function HeapPermute(n)                          ▷ 实现生成排列的Heap算法
Require: 一个正整数n和一个全局数组A[1...n]
Ensure: A中元素全排列
2:     if n=1 then
3:         write A
4:     else
5:         for i ← 1 to n do
6:             HeapPermute(n-1)
7:             if n is odd then
8:                 swap A[1] and A[n − 1]
9:             else
10:                swap A[i] and A[n]
11:            end if
12:        end for
13:    end if
14: end function
```

## 3.2 解决方法

采用先测试，然后总结归纳的方式来分析：

- $n = 1$时，输出$A_1$
- $n = 2$时，输出$A_1$、$A_2$，$A[2, 1]$
- $n = 3$时，for循环3次：
  1. $i = 1$时，HeapPermute(2)输出$A_1$、$A_2$、$A_3$，此时得到$A[2, 1, 3]$，交换1,3有$A[3, 1, 2]$
  2. $i = 2$时，HeapPermute(2)输出$A_3$、$A_1$、$A_2$，此时得到$A[1, 3, 2]$，交换1,3有$A[2, 3, 1]$

3. $i=3$时，HeapPermute(2)输出$A_2$、$A_3$、$A_1$，此时得到$A[3,2,1]$，交换1,3有$A[1,2,3]$

- $n=3$时，for循环4次：

1. $i=1$时，HeapPermute(3)输出$A_1$、$A_2$、$A_3$、$A_4$，$A[1,2,3]$顺序不变，交换1,4有$A[4,2,3,1]$
2. $i=2$时，HeapPermute(3)输出$A_4$、$A_2$、$A_3$、$A_1$，$A[4,2,3]$顺序不变，交换2,4有$A[4,1,3,2]$
3. $i=3$时，HeapPermute(3)输出$A_4$、$A_1$、$A_3$、$A_2$，$A[4,1,3]$顺序不变，交换3,4有$A[4,1,2,3]$
4. $i=4$时，HeapPermute(3)输出$A_4$、$A_1$、$A_2$、$A_3$，$A[4,1,2]$顺序不变，交换4,4有$A[4,1,2,3]$

此时得到$A[4,1,2,3]$，即全部输出之后数组A循环右移一位。

由此发现HeapPermute(n)输出全排列后数组元素循环右移一位。当n为奇数时，HeapPermute(n)输出全排列后数组元素顺序保持不变。只能通过归纳法证明，如下：

1. $i=1$时，显然成立。

2. $i=k$为偶数时，假设输出的是全排列，则$i=k-1$(奇数)时，$k-1$次循环中，每次前$k$个元素做全排列输出后循环右移一位，所以对换swap $A_1$and $A_n$可以保证每次将前k个元素中的一个换到$k-1$的位置，所以$k-1$次循环后输出的是$A[1,\ldots,k,1]$的全排列。

3. $i=k$为奇数时，假设输出的是全排列,则$i=k-1$(偶数)时，$k-1$次循环中，每次前$k$个元素做全排列输出后顺序保持不变，所以对换swap $A_i$and $A_n$可以保证每次将前$k$个元素中的一个换到$k-1$的位置，所以$k-1$次循环后输出的是$A[1,\ldots,k,1]$的全排列。

以上。

由此可以得到时间复杂度的递推公式为：

$$T(n)=\begin{cases}1 & n=1\\ n*[T(n-1)+2] & n\geq 2\end{cases} \tag{7}$$

改成非递归式子结果为$T(n)=n!$，时间复杂度为$O(n^{n-1})$或者为$O(n!)$

# 4 第四题

## 4.1 问题描述

对于求$n$ 个实数构成的数组中最小元素的位置问题，写出你设计的具有减治思想算法的伪代码，确定其时间效率，并与该问题的蛮力算法相比较。

## 4.2 解决方法

### 4.2.1 伪代码

如下：

**Algorithm 3** Find Min Position's algorithm

1: **function** FINDMINPOSITION($l, r$) ▷ 数组中最小元素的位置

**Require:** 含有$n$个元素的全局数组$A$，搜索起始位置$l$，以及介结束位置$r$ ▷ 初始化$l \leftarrow 0, r \leftarrow n$

**Ensure:** $A$中最小元素的值$v$及其位置$i$

2: **if** l = r **then**
3: $(v, i) \leftarrow (A_l, l)$
4: **end if**
5: $(v1, i1) \leftarrow$FindMinPosition($l, l + (r - l)/2$)
6: $(v2, i2) \leftarrow$FindMinPosition($l + (r - l)/2, r$)
7: **if** $v1 \leq v2$ **then**
8: $(v, i) \leftarrow (v1, i1)$
9: **else**
10: $(v, i) \leftarrow (v2, i2)$
11: **end if**
12: **end function**

### 4.2.2 分析

由此可以得到时间复杂度的递推公式为：

$$T(n) = \begin{cases} 1 & n = 1 \\ 2 * [T(\frac{n}{2}) + 1] & n \geq 2 \end{cases} \tag{8}$$

将此递推式子改为非递归式子有$T(n) = 2n - 1$，即时间复杂度为$O(2n - 1)$蛮力法通过一次扫描即可得到最小元素位置，时间复杂度为$O(n)$。结果显示效率不及蛮力法，因为此方法增加了比较次数。

# 5 第五题

## 5.1 问题描述

请给出约瑟夫斯问题的非递推公式$J(n)$，并证明之。其中，$n$ 为最初总人数，$J(n)$ 为最后幸存者的最初编号。

## 5.2 解决方法

已知幸存者号码的递推公式：

$$J(n)=\begin{cases}1 & n=1\\ J(\frac{n}{2})-1 & n=2k\\ J(\frac{n-1}{2})-1 & n=2k+1\end{cases} \tag{9}$$

幸存者号码的非地推公式为：设$n=2^k+b$，$J(n)=2*b+1,\text{s.t.}b\in[0,2^m),(m\geq 0)$

通过数学归纳法证明：

1. $i=1$时，$m=0$，$b=0$，$J(1)=2*b+1$，成立

2. $i\geq 1$时，

   - 当i为偶数时，设$k=\frac{i}{2}$成立，此时$J(k)=2*b+1$，而$J(i)=J(2k)=2J(k)-1=2(2b+1)-1=2(2b)+1$，即$k=i$时候上式成立。
   - 当i为奇数时，设$k=\frac{i-1}{2}$成立，此时$J(k)=2*b+1$，而$J(i)=J(2k+1)=2J(k)+1=2(2b+1)-1=2(2b+1)+1$，即$k=i$时候上式成立。

综上，式子$J(n)=2*b+1,\text{s.t.}b\in[0,2^m),m\geq 0$成立。