算法基础

第一次作业参考答案（DDL：2024 年 9 月 x 日 23:59）

Q1.（$15+20=35$ 分）化学学院的小明同学制备了一种新型材料，并用该材料制成了两颗大小、形状、硬度完全相同的实心球。同时，有一栋 $n=128$ 层高的新理化大楼。现在需要确定它们的抗摔能力参数 x（它们从高度大于等于 x 层落下时会破碎，而从高度小于等于 $x-1$ 层落下时不会破碎）。假设 $x\in\{1,2,\ldots,127,128\}$，且 x 的取值满足均匀分布。为此，同实验室的小方设计了两种测定抗摔能力参数的方法：

1. （伪二分法）第一次试验在 $n/2=64$ 层进行，若球破碎，则从第 1 层开始逐层试验，直至确定抗摔能力参数 x。反之，则在 $n/2+n/4=96$ 层进行第二次试验，若球破碎，则从第 $n/2+1=65$ 层开始逐层试验，直至确定抗摔能力参数 x。反之，则在 $n/2+n/4+n/8=112$ 层进行第三次试验，以此类推。
2. （改良法）第一次试验在 $k=16$ 层进行，若球破碎，则从第 1 层开始逐层试验，直至确定抗摔能力参数 x。反之，则在 $k+(k-1)=31$ 层进行第二次试验，若球破碎，则从第 $k+1=17$ 层开始逐层试验，直至确定抗摔能力参数 x。反之，则在 $k+(k-1)+(k-2)=45$ 层进行第三次试验，以此类推。

函数 $f(x)$ 和 $g(x)$ 定义为，当新型材料球的抗摔能力参数为 x 时，分别采用伪二分法和改良法所需的试验次数（例如，$f(17)=18$，$g(17)=3$）。请分别计算以上两种方法确定抗摔能力参数 x 所需要的期望试验次数 $E(f(x))$ 和 $E(g(x))$。

Solution:

1. 首先，分析 x 与 $f(x)$ 的关系：

$$f(x)=\begin{cases} x+t-\frac{(2^{t-1}-1)n}{2^{t-1}} & ,x\in[\frac{(2^{t-1}-1)n}{2^{t-1}}+1,\frac{(2^t-1)n}{2^t}-1]\ (t\in[1,6]) \\ \frac{n}{2^t}+t-1 & ,x=\frac{(2^t-1)n}{2^t}\ (t\in[1,7]) \\ \lg n & ,x=n \end{cases}$$

则

$$E(f(x))=\frac{1}{128}\sum_{i=1}^{128}f(x)=23.7578125$$

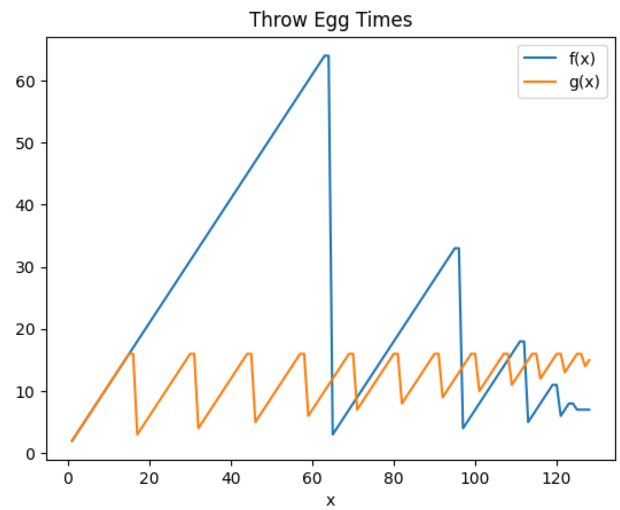

图 1: 函数 $f(x)$ 和 $g(x)$

2. 首先，分析 x 与 $g(x)$ 的关系：

$$g(x)=\begin{cases} t & ,x=\frac{t(2k-t+1)}{2}\ (t\in[1,15]) \\ t+x-\frac{(t-1)(2k-t)}{2} & ,x\in[\frac{(t-1)(2k-t)}{2}+1,\frac{t(2k-t+1)}{2}-1]\ (t\in[1,15]) \end{cases}$$

则

$$E(g(x))=\frac{1}{128}\sum_{i=1}^{128}g(x)=11.6328125$$

Q2.（$5\times 2=10$ 分）判断以下命题的正误。若正确，请给出证明；若错误，请给出反例。

1. 若 $f(n)=\Theta(g(n))$，则有 $\lg(f(n))=\Theta(\lg g(n))$。
2. 若 $f(n)=\sum_i^l b_i n^i$，且 $b_l>0$，则有 $f(n)=\Theta(n^l)$。

Solution:

1. **F**。假设 $f(n)=g(n)=1/n$，则 $\lg f(n)$ 不满足渐进非负。
2. **T**。令 $c_1=\frac{b_l}{2}$，$c_2=b_l+1$，

$$n_1=\lceil l\max(b_{l-1},\sqrt{b_{l-2}},\dots,\sqrt[l]{b_1})\rceil$$

，则当 $n > n_1$ 时，

$$f(n) \leq c_2 g(n)$$

$$n_2 = \lceil \frac{lb_l}{2} \max(b_{l-1}, \sqrt{b_{l-2}}, \ldots, \sqrt[l]{b_1}) \rceil$$

，则当 $n > n_2$ 时，

$$f(n) \geq c_1 g(n) \geq 0$$

，故当 $n > n_0 = \max(n_1, n_2)$ 时，

$$0 \leq c_1 g(n) \leq f(n) \leq c_2 g(n)$$

，即 $f(n) = \Theta(n^l)$。

Q3.（$3 \times 5 = 15$ 分）假设你有以下列出的五种运行时间的算法。(假设这些是作为输入大小 n 函数执行的确切操作次数。）假设你有一台计算机，每秒可以执行 10^{10} 次操作，你需要在最多一个小时的计算时间内得到结果。对于每种算法，你能在一小时内得到结果的最大输入大小 n 是多少?

1. n^4
2. $100n^2$
3. $n \log n$
4. 2^n
5. 2^{2^n}

Solution:

1.

$$3.6 \times 10^{13} \geq n^4$$

$$n = \lfloor \sqrt[4]{3.6 \times 10^{13}} \rfloor = 2449$$

2.

$$3.6 \times 10^{13} \geq 100n^2$$

$$n = \lfloor \sqrt{3.6 \times 10^{11}} \rfloor = 6.0 \times 10^5$$

3.

$$3.6 \times 10^{13} \geq n \log n$$

```python
import numpy as np
# given x
# return the max n, let n\log n less than x
def get_n(x):
    # use binary search to find the max n
    upper = x
    lower = 1
    while lower < upper:
        mid = (upper + lower) // 2
        if mid * np.log2(mid) < x:
            lower = mid + 1
        else:
            upper = mid
    return lower - 1
```

$$n = 906316482853$$

4.

$$3.6 \times 10^{13} \geq 2^n$$

$$n = \lfloor \lg(3.6 \times 10^{13}) \rfloor = 45$$

5.

$$3.6 \times 10^{13} \geq 2^{2^n}$$

$$n = \lfloor \lg \lg(3.6 \times 10^{13}) \rfloor = 5$$

Q4.（20 分）将下列函数按照增长顺序排列。找出一个满足 $g_1 = O(g_2)$，$g_2 = O(g_3)$，$g_3 = O(g_4)$，$g_4 = O(g_5)$，$g_5 = O(g_6)$，$g_6 = O(g_7)$，$g_7 = O(g_8)$ 的函数排列 $g_1, g_2, g_3, g_4, g_5, g_6, g_7, g_8$。

$$f_1(n) = n^\pi$$

$$f_2(n) = \pi^n$$

$$f_3(n) = \binom{n}{5}$$

$$f_4(n) = \sqrt{2^{\sqrt{n}}}$$

$$f_5(n) = \binom{n}{n-4}$$
$$f_6(n) = 2^{\log^4 n}$$
$$f_7(n) = n^{5(\log n)^2}$$
$$f_8(n) = n^4\binom{n}{4}$$

Solution:

$$f_1(n), f_5(n), f_3(n), f_8(n), f_7(n), f_6(n), f_4(n), f_2(n)$$

Q5.（$5+15=20$ 分）中秋佳节即将到来，科大糕点厂决定给 n 个科大幼儿园的小朋友分发科大定制月饼。小朋友们排成一队依次领取，从队头数起第 $i(1 \le i \le n)$ 个小朋友有 a_i 朵小红花，分到 c_i 枚月饼。同时，为了奖励小红花多的小朋友，幼儿园园长制定了以下分发规则：

- 每个小朋友至少分到 1 枚月饼（$c_i \ge 1$）；
- 若小朋友的小红花数多于相邻的小朋友的小红花数，则其分到的月饼数也多于相邻的小朋友（$a_i > a_j (1 \le i, j \le n, |i-j| = 1) \Rightarrow c_i > c_j$）；

1. 若 $n=5$，$a_1=1$，$a_2=3$，$a_3=3$，$a_4=5$，$a_5=2$，请给出一个满足以上规则的月饼分配方案。
2. 现在 n, a_i 已知，请设计满足以上规则的分发方案（算法），输给出每个小朋友分到的月饼数 c_i，同时使得 $\sum_{i=1}^{n} c_i$ 最小。给出该算法的伪代码。（不需要证明算法的最优性）

Solution:

1. 答案不唯一：
$$[c_1\ c_2\ c_3\ c_4\ c_5] = [1\ 2\ 1\ 2\ 1]$$
2.

Algorithm 1: Minimize $\sum c_i$

Input: $n,\ a_i\ (1 \leq i \leq n)$

Output: $c_i\ (1 \leq i \leq n)$

for $j \leftarrow 1$ *to* n **do**
    $c_j \leftarrow 1$
end
for $j \leftarrow 1$ *to* $n-1$ **do**
    if $a_{j+1} > a_j$ **then**
        $c_{j+1} \leftarrow c_j + 1$
    end
end
for $j \leftarrow n$ *to* 2 **do**
    if $a_j < a_{j-1}$ *and* $c_j \geq c_{j-1}$ **then**
        $c_{j-1} \leftarrow c_j + 1$
    end
end
return $[c_i]$