

算法课第 2 次作业

作业得分：

	题目 1	题目 2	题目 3	题目 4	题目 5	总分
分数						
阅卷人						

题目 1:

答：不能保证总存在稳定时间表。以下选择任务(b)来求解。

设：当前共有 $n=2$ 个时段，A 网的节目列表 $Pa=\{a_0, a_1\}$ ，B 网的时间表 $Pb=\{b_0, b_1\}$ ，而对于这些电视节目而言，收视率关系有 $a_0 > b_0 > a_1 > b_1$ 。

以下分两种时间表的情况进行讨论：

(1) 若：给出时间表 $(S, T) = \{<a_0, b_0>, <a_1, b_1>\}$ ，则对于 B 网，目前能赢得的时段有 0 个，若 B 网单方面改变时间表为 $(S, T') = \{<a_0, b_1>, <a_1, b_0>\}$ ，则可赢得 1 个时段，是一种更优的策略。

(2) 若：给出时间表 $(S, T) = \{<a_0, b_1>, <a_1, b_0>\}$ ，则对于 A 网，目前能赢得的时段有 1 个，若 B 网单方面改变时间表为 $(S, T') = \{<a_0, b_1>, <a_1, b_0>\}$ ，则可赢得 2 个时段，是一种更优的策略。

综上所述，对每组节目和收视率，不能保证总存在稳定时间表。

题目 2:

答：1 小时=60 分钟=3600 秒=3600 $\times 10^{10} = 3.6 \times 10^{13}$ 次运算。

$$(a) \quad n^2 = 3.6 \times 10^{13} \rightarrow n = \sqrt{3.6 \times 10^{13}} \rightarrow n = 6 \times 10^6$$

$$(b) \quad n^3 = 3.6 \times 10^{13} \rightarrow n = \sqrt[3]{3.6 \times 10^{13}} \rightarrow n = 33015$$

$$(c) \quad 100n^2 = 3.6 \times 10^{13} \rightarrow n^2 = 3.6 \times 10^{11} \rightarrow n = \sqrt{3.6 \times 10^{11}} \rightarrow n = 6 \times 10^5$$

$$(d) \quad n \log n = 3.6 \times 10^{13} \rightarrow 2^{n \log n} = 3.6 \times 10^{13} \rightarrow n \times 2^n = 2^{3.6 \times 10^{13}}$$

$$(e) \quad 2^n = 3.6 \times 10^{13} \rightarrow n = \log(3.6 \times 10^{13}) \rightarrow n = 45$$

(f) 由于 2^{2^n} 增长速度过快，以下采用枚举法进行计算。最终得到 n 不应超过 5。

表 1 运算结果

n	2^n	2^{2^n}
1	2	4
2	4	16
3	8	256
4	16	65536
5	32	4294967296
6	64	18446744073709551616L

以下展示部分关键运算结果截图（本计算由 python2.7 实现）：

```
python
WARNING: Python 2.7 is not recommended.
This version is included in macOS for compatibility with legacy software.
Future versions of macOS will not include Python 2.7.
Instead, it is recommended that you transition to using 'python3' from within Terminal.

Python 2.7.16 (default, Jun  5 2020, 22:59:21)
[GCC 4.2.1 Compatible Apple LLVM 11.0.3 (clang-1103.0.29.20) (-macos10.15-objc-
on darwin
Type "help", "copyright", "credits" or "license" for more information.
>>> a = 3600000000000000
>>> pow(a, 1.0/3)
33019.27248894625
>>> pow(33019, 3)
35999108745859
>>> pow(33020, 3)
36002379608000
>>>
```

图 1 (b) 运算结果

```
python
WARNING: Python 2.7 is not recommended.
This version is included in macOS for compatibility with legacy software.
Future versions of macOS will not include Python 2.7.
Instead, it is recommended that you transition to using 'python3' from within Terminal.

Python 2.7.16 (default, Jun  5 2020, 22:59:21)
[GCC 4.2.1 Compatible Apple LLVM 11.0.3 (clang-1103.0.29.20) (-macos10.15-objc-
on darwin
Type "help", "copyright", "credits" or "license" for more information.
>>> a = 3600000000000000
>>> import math
>>> math.log(a, 2)
45.033062140090664
>>>
```

图 2 (e) 运算结果

```
python
WARNING: Python 2.7 is not recommended.
This version is included in macOS for compatibility with legacy software.
Future versions of macOS will not include Python 2.7.
Instead, it is recommended that you transition to using 'python3' from within Terminal.

Python 2.7.16 (default, Jun  5 2020, 22:59:21)
[GCC 4.2.1 Compatible Apple LLVM 11.0.3 (clang-1103.0.29.20) (-macos10.15-objc-
on darwin
Type "help", "copyright", "credits" or "license" for more information.
>>> pow(2, 4)
16
>>> pow(2, 8)
256
>>> pow(2, 16)
65536
>>> pow(2, 32)
4294967296
>>> pow(2, 64)
18446744073709551616L
```

图 3 (f) 运算结果

题目 3:

答:

1. 显然 $n < 2^n$ 故有 $2^n < 2^{2^n}$ 。

$$2. \lim_{n \rightarrow \infty} \frac{2^n}{n^{\log n}} = \lim_{n \rightarrow \infty} 2^{\log \frac{2^n}{n^{\log n}}} = \lim_{n \rightarrow \infty} 2^{\log 2^n - \log n^{\log n}} = \lim_{n \rightarrow \infty} 2^{\log 2^n - \log n^{\log n}} = \lim_{n \rightarrow \infty} 2^{n - (\log n)^2} \infty。$$

故有 $n^{\log n} < 2^n$ 。

3. 显然常数小于 $\log n$ ，故有 $n^{\frac{4}{3}} < n^{\log n}$ 。

$$4. \lim_{n \rightarrow \infty} \frac{n(\log n)^3}{n^{\frac{4}{3}}} = \lim_{n \rightarrow \infty} \frac{(\log n)^3}{n^{\frac{1}{3}}} = \lim_{n \rightarrow \infty} \frac{\log n}{n^{\frac{1}{9}}} = 0，故有 n(\log n)^3 < n^{\frac{4}{3}}。$$

5. 由于 $2^{\sqrt{\log n}} < 2^{\log n} = n < n(\log n)^3$ ，故有 $2^{\sqrt{\log n}} < n(\log n)^3$ 。

综上所述：按照增长率递增的顺序排列函数如下：

$$2^{\sqrt{\log n}} < n(\log n)^3 < n^{\frac{4}{3}} < n^{\log n} < 2^n < 2^{2^n}$$

题目 4:

(a) 由于要求在小于线性复杂度的时间内完成计算，因此朴素的二分查找+顺序遍历的思想就不符合要求，因为这种方法仅仅是将 $O(n)$ 变成了 $O\left(\frac{1}{2^k}n\right)$ 其中 $\frac{1}{2^k}$ 仍是一个常数，即还是一个 $O(n)$ 复杂度的计算问题。

为了降低计算复杂度，可以尝试采用开方的形式进行求解，即第一个瓶子的尝试位置序列可以表示为： $i \lfloor \sqrt[n]{n} \rfloor, i \in [1, \lfloor \sqrt[n]{n} \rfloor]$ ，假设第一个瓶子在掷出第 k 次后碎掉了，那么第二个瓶子将从第 $(k-1)\lfloor \sqrt[n]{n} \rfloor + 1$ 个位置开始尝试。

对于上述方法，第一个瓶子最多尝试 $\lfloor \sqrt[n]{n} \rfloor$ 次，第二个瓶子同样最多尝试 $\lfloor \sqrt[n]{n} \rfloor$ 次，整体时间复杂度为 $O(2\lfloor \sqrt[n]{n} \rfloor) = O(\sqrt[n]{n}) < O(n)$ 。

(b) 受到(a)的启发，我们在掷第 k 个瓶子时，可以将下次查找的区间缩小至 $n^{\frac{k-1}{k}}$ 的范围内，即对于第 k 个瓶子，我们的尝试位置序列可以表示为： $i \lfloor n^{\frac{k-1}{k}} \rfloor, i \in [1, \lfloor n^{\frac{1}{k}} \rfloor]$ ，该步最多尝试了 $\lfloor n^{\frac{1}{k}} \rfloor$ 次。

接着，对于剩下的 $k-1$ 个瓶子，我们的尝试位置序列可以表示为： $i \lfloor n^{\frac{k-2}{k-1}} \rfloor, i \in [1, \lfloor n^{\frac{1}{k-1}} \rfloor]$ 。

递归进行上述步骤直至剩下 2 个瓶子，该问题回到了问题(a)，对 k 个瓶子的尝试次数进行累加得总尝试次数为 $(k-1)\lceil k\sqrt{n} \rceil$ ，即使用该方法的计算复杂度为 $O(k^k\sqrt{n})$ 。

对于 $f_k(n) = k^k\sqrt{n}$ 而言，有： $\lim_{n \rightarrow \infty} \frac{f_k(n)}{f_{k-1}(n)} = 0$ 。符合题目要求。

题目 5:

答：命题成立。

证明：采用反证法，假设满足要求的图 G 不是联通的，则图 G 必然有 n 个 ($n \geq 2$) 联通子图。对于这 n 个子图，必有其中一个子图 S 的节点数 $|S| \leq \frac{n}{2}$ 。而对于 S 中的任一节点 v ，其度数之最大值 $d_{\max} \leq \frac{n}{2} - 1 < \frac{n}{2}$ 。这与我们的前提图 G 的每个节点度数至少为 $\frac{n}{2}$ 是矛盾的。

故假设不成立，原命题成立。

此命题证毕。

郑重声明

本作业由作者独立完成。抄袭行为在任何情况下都是不能容忍的 (COPY is strictly prohibited under any circumstances)！由抄袭所产生的一切后果由抄袭者承担，勿谓言之不预也。

陈麒先