算法设计与分析
第五次作业

作业情况

教材：算法设计技巧与分析 [沙特]M. H. Alsuwaiyel
题目范围：归纳法、堆、复杂度分析
邮箱：wjyyy1@126.com
批改时间：2023 年 5 月 18 日

习题 1

Gray 码是一个长度为 2^n 的序列，序列需要满足以下三个条件，
1）序列中无相同元素，
2）每个元素都是长度为 n 位的（0,1）串，
3）相邻元素恰好只有 1 位不同。例如：
$n=1$ 时，Gray 码：$\{0,1\}$
$n=2$ 时，Gray 码：$\{00,01,11,10\}$
$n=3$ 时，Gray 码：$\{000,001,011,010,110,111,101,100\}$
请用归纳法设计一个算法，对任意的 n 构造相应的 Gray 码。
(1) 写出 $n=4$ 时的 Gray 码。
(2) 请写出归纳法的主要思想及伪代码。
(3) 写出归纳法时间复杂度的递推式，并求解?

答案 1

答案不唯一。

(1) $\{0000, 0001, 0011, 0010, 0110, 0111, 0101, 0100, 1100, 1101, 1111, 1110, 1010, 1011, 1001, 1000\}$

(2) 当我们需要长度为 2^n 的 Gray 码序列时，需要首先获得 2^{n-1} 的 Gray 码，然后将这个长为 2^{n-1} 的序列翻转后拼接到原序列后面。对这个新序列的前一半的所有码，前面加上 0，后一半的所有码，前面加上 1，即可得到。

Algorithm 1: Gray code

Input: 序列长度 2^n

Output: 长度为 2^n 的 Gray 码

```
Gray (n)

Function Gray(n):
    if n = 1 then
        g ← [];
        g[1] ← 0;
        g[2] ← 1;
        return g;
    end
    g ← Gray (n − 1);
    f ← g.Reverse()
    for i ← 1 to 2^(n−1) do
        g[i] =' 0' + g[i];
        f[i] =' 1' + f[i];
    end
    return 连接 g, f;
```

(3) $T(n) = T(n-1) + 2^n = O(2^{n+1})$（或 $T(n) = T(n-1) + n \cdot 2^n = O(n \cdot 2^{n+1})$）

解析：

对于 $n = 4$ 时的 Gray 码，需要基于 $n = 3$ 时的 Gray 码进行扩展。因为 $n = 3$ 的 Gray 码已经满足所述条件，因此先考虑直接增加一位。但是增加之后需要考虑如何连接。因为前面一段和后面一段一定有最高位的区别，所以连接处必须有这个不同位。所以要把 Gray 码进行反转然后拼接，见算法1。

$T(n)$ 的计算首先是调用 $T(n-1)$，然后将 0 和 1 拼接。考虑到拼接的复杂度不同，答案可能为 $O(2^{n+1})$ 和 $O(n \cdot 2^{n+1})$。

习题 2

有 n 个函数，分别为 $F_1, F_2, \ldots, F_n$。定义 $F_i(x) = A_i x^2 + B_i x + C_i (x \in \mathbb{N}^*)$。给定这些 A_i、B_i 和 C_i，请求出所有函数的所有函数值中最小的 m 个（如有重复的则都输出）。

(1) 给定 $F_1(x) = x^2 + 3x + 4$，$F_2(x) = 2x^2 - 4x + 1$，$F_3(x) = 3x^2 + 5$，求所有函数值中最小的 4 个。

(2) 写出算法的伪代码，分析复杂度。

答案 2

(1) $\{-1, 1, 7, 8\}$

(2) 因为所有函数值中的最小值，一定从其中一个函数的最小值得到，所以我们每次维护所有函数的最小值（共 m 个），然后从这 m 个里面取出最小值并更新。

Algorithm 2: Find min

Input: 函数数量 n 和函数信息 A, B, C

Output: 最小的 m 个值 ans

$ans \leftarrow []$;
$q \leftarrow Heap()$;
for $i \leftarrow 1$ to n **do**
  $mid \leftarrow -B[i]/(2 * A[i])$;
  if $mid < 0$ **then** $q.push((A[i] * x * x + B[i] * x + C, i, mid, 1))$;
  else $q.push((A[i] * x * x + B[i] * x + C, i, mid, ceil(mid)))$;
end
for $i \leftarrow 1$ to m **do**
  $(func, num, mid, x) \leftarrow q.pop()$;
  $ans[i] = func$;
  if $x > mid$ **then** $x \leftarrow mid - (x - mid)$;
  else $x \leftarrow mid + (mid - x) + 1$;
  if $x \leq 0$ **then** $x \leftarrow mid + (mid - x) + 1$;
  $q.push(((A[i] * x * x + B[i] * x + C, i, mid, x)))$
end

复杂度为 $O((n + m) \log n)$。

解析：使用堆来维护这 n 个函数的最小值，每取出一个就改为次小值。在上述伪代码中，q 是一个小根堆，按照键值从前到后的顺序从小到大排序。

查找的过程分别是：mid，$mid + i$，$mid - i$，$mid + i + 1$，……

当 $mid - i \leq 0$ 时，则跳过这个值。

习题 3

有 n 个人参加一门舞蹈课。每个人的舞蹈技术由整数来决定。在舞蹈课的开始，他们从左到右站成一排。当这一排中至少有一对相邻的异性时，舞蹈技术相差最小的那一对会出列并开始跳舞。如果不止一对，那么最左边的那一对出列。一对异性出列之后，队伍中的空白按原顺序补上（即：若队伍为 ABCD，那么 BC 出列之后队伍变为 AD）。舞蹈技术相差最小即是 a_i 的绝对值最小。
任务是模拟以上过程，确定跳舞的配对及顺序。
(1) 舞蹈技术从左到右分别为 $[5, 1, 8, 9, 6]$，给出计算过程。
(2) 写出算法的伪代码，分析复杂度。

答案 3

(1) 最开始，相差最小的为 $(8, 9)$，序列变为 $[5, 1, 6]$，接着相差最小的是 $(5, 1)$，序列变为 $[6]$，所以跳舞的配对分别是 $(8, 9), (5, 1)$
(2) 使用堆来维护所有相邻的异性对。当一对异性出堆时，它旁边的两组信息也需要更新，这个时候需要维护一个删除标记。只有两个异性都没有被删除时才能选择它们出列。

Algorithm 3: Dance pairing

Input: 人数 n 和跳舞技术 a
Output: 跳舞的配对 ans

$ans \leftarrow []$;
$used \leftarrow []$;
$q \leftarrow Heap()$;
for $i \leftarrow 1$ to $n-1$ **do** $q.push(abs(a[i] - a[i+1]), i)$;
$cnt \leftarrow 1$;
while $!q.empty()$ **do**
  $(x, place) \leftarrow q.pop()$;
  if $used[place]$ or $used[place+1]$ **then continue**;
  $ans[cnt] \leftarrow (a[place], a[place+1])$;
  $used[place] \leftarrow$ true;
  $used[place+1] \leftarrow$ true;
end

维护大小为 $O(n)$ 的堆，时间复杂度为 $O(n \log n)$

解析：本题题目有不清晰的地方，默认男生两边都是女生。但同学们可以进行自定义性别的探索。

习题 4

(1) 试证明若 $f_1(n) = O(g_1(n))$ 并且 $f_2(n) = O(g_2(n))$，
那么 $f_1(n) + f_2(n) = O(\max\{g_1(n), g_2(n)\})$；
(2) 试证明 $O(f(n)) + O(g(n)) = O(\max\{f(n), g(n)\})$

答案 4

(1) 根据给定条件，$f_1(n) \le c_1 g_1(n), f_2(n) \le c_2 g_2(n)$，
所以 $f_1(n) + f_2(n) = c_1 g_1(n) + c_2 g_2(n) \le \max\{c_1, c_2\} \max\{g_1(n), g_2(n)\}$。
令 $c = \max\{c_1, c_2\}$，则

$$f_1(n) + f_2(n) \le c \max\{g_1(n), g_2(n)\} = O(\max\{g_1(n), g_2(n)\})$$

(2) 对于任意 $f_1(n) \in O(f(n))$ ，存在 c_1 和 n_1，使得对所有 $n \ge n_1$，有 $f_1(n) \le c_1 f(n)$。
类似地，对于任意 $g_1(n) \in O(g(n))$，存在 c_2 和 n_2，使得对所有 $n \ge n_2$，有 $g_1(n) \le c_2 g(n)$。
令 $c_3 = \max\{c_1, c_2\}$，$n_3 = \max\{n_1, n_2\}$，$h(n) = \max\{f(n), g(n)\}$。则对所有的 $n \ge n_3$，有：

$$f_1(n) + g_1(n) \le c_1 f(n) + c_2 g(n) \le c_3 f(n) + c_3 g(n) = c_3(f(n) + g(n)) \le$$

$$c_3 2 \max\{f(n), g(n)\} = O(\max\{f(n), g(n)\})$$

注意：参考答案中多为最优做法，如果同学们写的复杂度较高，也会算正确。但不可以太高，否则会被扣分。

作业改的过程比较快，批错的大概率错了，批对的不一定全对。

需要注意的是，堆操作的复杂度是每次 $O(\log n)$。