入选理由

- 一个难度不小的分治。需要一点点数学知识。

标签

- 分治

难度

- 中等

题目地址(932. 漂亮数组)

https://leetcode-cn.com/problems/beautiful-array/

题目描述

对于某些固定的 N，如果数组 A 是整数 1, 2, ..., N 组成的排列，使得：

对于每个 i < j，都不存在 k 满足 i < k < j 使得 A[k] * 2 = A[i] + A[j]。

那么数组 A 是漂亮数组。

给定 N，返回任意漂亮数组 A（保证存在一个）。

示例 1：

```
输入：4
输出：[2,1,4,3]
```

示例 2：

```
输入：5
输出：[3,1,2,5,4]
```

提示：

```
1 <= N <= 1000
```

前置知识

- 分治

公司

- 暂无

思路

由数字的奇偶特性，可知：**奇数 + 偶数 = 奇数** 。

因此如果要使得：**对于每个 i < j，都不存在 k 满足 i < k < j 使得 A[k] * 2 = A[i] + A[j]** 成立。由于 A[k] * 2 一定为偶数，因此我们可以令 A[i] 和 A[j] 一个为奇数，另一个为偶数即可。

另外漂亮数组还有两个非常重要的性质，也是本题的突破口。

- 性质 1：如果数组 A 是 漂亮数组，那么将 A 中的每一个数 x 进行 kx + b 的映射，其仍然为漂亮数组。其中 k 为不等于 0 的整数， b 为整数。
- 性质 2：如果数组 A 和 B 分别是**不同奇偶性**的漂亮数组，那么将 A 和 B 拼接起来仍为漂亮数组。

举个例子。我们要求长度为 N 的漂亮数组，那么一定有 N / 2 个偶数 和 N - N / 2 个奇数。

> 这里的除法为地板除。

假设长度为 N / 2 和 N - N/2 的漂亮数组被计算出来了。那么我们只需要对长度为 N/2 的漂亮数组通过性质 1 变换成全部为偶数的漂亮数组，并将长度为 N - N/2 的漂亮数组也通过性质 1 变换成全部为奇数的漂亮数组。接下来利用性质 2 将其进行拼接即可得到一个漂亮数组。

刚才我们**假设长度为 N / 2 和 N - N/2 的漂亮数组被计算出来了**，实际上我们并没有计算出来，那么其实可以用同样的方法来计算。其实就是分治，将问题规模缩小了，问题本质不变。递归的终点自然是 N == 1，此时可直接返回 [1]。

关键点

- 利用数学基本性质**奇数 + 偶数 = 奇数**
- 对问题进行分解，分解为规模相同的同等问题
- 仿射变换

代码

- 语言支持：Python3，CPP，JS，Java

Python3 Code:

```python
class Solution:
    def beautifulArray(self, N: int) -> List[int]:
        @lru_cache(None)
        def dp(n):
            if n == 1:
                return [1]
            ans = []
            # [1,n] 中奇数比偶数多1或一样
            for a in dp(n - n // 2):
                ans += [a * 2 - 1]
            for b in dp(n // 2):
                ans += [b * 2]
            return ans

        return dp(N)
```

CPP Code:

```cpp
class Solution {
public:
    vector<int> beautifulArray(int n) {
        if (n == 1) return {1};
        vector<int> res;
        auto leftArr = beautifulArray((n+1)/2);
        auto rightArr = beautifulArray(n/2);
        for (auto& x : leftArr)
            res.push_back(2*x - 1);
        for (auto& x : rightArr)
            res.push_back(2*x);

        return res;
    }
};
```

JS Code:

```
const beautifulArray = function (n) {
  let arr = [];
  arr.push(1);
  while (arr.length < n) {
    let tmp = [];
    for (const i of arr) if (i * 2 - 1 <= n) tmp.push(i * 2 - 1);
    for (const i of arr) if (i * 2 <= n) tmp.push(i * 2);
    arr = tmp;
  }

  return arr;
};
```

Java Code:

```
class Solution {
    Map<Integer, int[]> memo = new HashMap();
    public int[] beautifulArray(int n) {
        memo.put(1, new int[]{1});
        return dp(n);
    }
    private int[] dp(int n) {
        if (memo.get(n) != null) {
            return memo.get(n);
        }
        int[] res = new int[n];
        int i = 0;
        for (int x : dp((n + 1) / 2)) {
            res[i++] = 2 * x - 1;
        }
        for (int x : dp(n / 2)) {
            res[i++] = 2 * x;
        }
        memo.put(n, res);
        return res;
    }
}
```

复杂度分析

令 n 为数组长度。

- 时间复杂度：$O(nlogn)$
- 空间复杂度：$O(n + logn)$

关于复杂度大家可以利用递推公式推出来，这里留给大家自己推导试试。

> 此题解由 **力扣刷题插件** 自动生成。

力扣的小伙伴可以**关注我**，这样就会第一时间收到我的动态啦~

以上就是本文的全部内容了。大家对此有何看法，欢迎给我留言，我有时间都会一一查看回答。更多算法套路可以访问我的 LeetCode 题解仓库：https://github.com/azl397985856/leetcode 。 目前已经 45K star 啦。大家也可以关注我的公众号《力扣加加》带你啃下算法这块硬骨头。

关注公众号力扣加加，努力用清晰直白的语言还原解题思路，并且有大量图解，手把手教你识别套路，高效刷题。

上一页 下一页