2016 年度 大問 5

hari64boli64 (hari64boli64@gmail.com)

2023 年 5 月 5 日

1 問題

累積積、SegmentTree

2 解答

5.py が解答。

ソースコード 1 answer

```
import math
import random
import numpy as np
from typing import List


def makeProblem(problemNo: int):
    assert 1 <= problemNo <= 4
    zeroRate = random.random()
    generator = random.random
    n = random.randint(1, 10)
    if problemNo == 1:
        zeroRate = 0.0
    if problemNo == 2:
        generator = lambda: random.randint(0, 1)
    if problemNo == 4:
        n = 16
        zeroRate = 1 / n
        generator = lambda: random.randint(1, 5)
    As = [0 if random.random() < zeroRate else generator() for _
        in range(n)]
    return n, As


```

```python
def check(n: int, As: List[int], solver: callable):
    # fがアルゴリズム，dataがデータ構造に対応する
    f, data = solver(n, As)
    # 全てのクエリに対して正しい答えが返ってくるかチェックする
    for s in range(1, n + 1):
        for t in range(s, n + 1):
            ans = f(s, t, data)
            print(s, t, ans)
            if not np.isclose(ans, math.prod(As[s - 1 : t])):
                raise Exception("Wrong Answer")
    print("ok!")


def solve1(n: int, As: List[int]) -> callable:
    """
    \prod_{i=s}^{t}A[i] =
    (\prod_{i=1}^{t}A[i]) / (\prod_{i=1}^{s-1}A[i])
    """
    Ps = [1.0]
    for i in range(n):
        Ps.append(Ps[-1] * As[i])

    def f(s: int, t: int, Ps: List[int]):
        return Ps[t] / Ps[s - 1]

    return f, Ps


def solve2(n: int, As: List[int]) -> callable:
    """
    0の個数を数えて、
    [s,t]間でその個数の増減がないかを調べれば良い
    """
    numOfZeros = [0]
    for i in range(n):
        numOfZeros.append(numOfZeros[-1] + int(As[i] == 0))

    def f(s: int, t: int, numOfZeros: List[int]):
        return 0 if numOfZeros[t] != numOfZeros[s - 1] else 1

    return f, numOfZeros


def solve3(n: int, As: List[int]) -> callable:
    """
    (1)と(2)を組み合わせるだけで良い
    Psは、0が出てきたらリセットの意味で1にする
```

```
71      """
72      Ps = [1.0]
73      for i in range(n):
74          Ps.append(Ps[-1] * As[i])
75          if Ps[-1] == 0:
76              Ps[-1] = 1
77
78      numOfZeros = [0]
79      for i in range(n):
80          numOfZeros.append(numOfZeros[-1] + int(As[i] == 0))
81
82      def f(s: int, t: int, Ps_numOfZeros: List[int]):
83          Ps, numOfZeros = Ps_numOfZeros
84          return 0 if numOfZeros[t] != numOfZeros[s - 1] else Ps[t]
                / Ps[s - 1]
85
86      return f, (Ps, numOfZeros)
87
88
89  def vis4():
90      """
91      SegmentTree、もしくは、BinaryIndexedTreeで検索
92      空間計算量は、O(logN)のように思えるが、
93      よくよく考えると、2N-1で抑えられている
94
95      出力例:
96
97      As=[3, 1, 4, 1, 3, 3, 3, 2, 1, 3, 3, 0, 5, 4, 3, 5]
98
99      0段目:    0-----------------------------------------------
100     1段目: 648----------------------  0----------------------
101     2段目:  12---------- 54---------  0---------300---------
102     3段目:   3---   4---   9---   6---   3---   0--- 20--- 15---
103     4段目:   3   1   4   1   3   3   3   2   1   3   3   0   5   4   3   5
104     """
105     n, As = makeProblem(4)
106     print(f"{As=}")
107     logN = round(math.log2(n))
108     for i in range(logN + 1):
109         print(f"{i}段目: ", end="")
110         for j in range(2**i):
111             print(
112                 f"{math.prod(As[(2**(logN-i)) * j : (2**(logN-i))
                        * (j + 1)]):>3}",
113                 end="-" * (3 * ((2 ** (logN - i) - 1))),
114             )
115         print()
```

```
116
117
118 def main():
119     for _ in range(10):
120         n, As = makeProblem(1)
121         print(f"{n=}, {As=}")
122         check(n, As, solve1)
123     for _ in range(10):
124         n, As = makeProblem(2)
125         print(f"{n=}, {As=}")
126         check(n, As, solve2)
127     for _ in range(10):
128         n, As = makeProblem(3)
129         print(f"{n=}, {As=}")
130         check(n, As, solve3)
131     vis4()
132
133
134 if __name__ == "__main__":
135     main()
```