Devoir 7 IFT2125-A-H19

Student: Qiang Ye (20139927)

Date: 24 Jan 2019

mailto: samuel.ducharme@umontreal.ca (mailto:samuel.ducharme@umontreal.ca)

Question

Vous faites partie d'un groupe de n personnes ($n > 1$) enlevées par des ravisseurs. Au bout de quelques mois, ces derniers en ont marre de vous car vous leur compliquez la vie. Ils vous appellent un soir et vous disent que le lendemain matin, il vont se débarasser de vous. On vous alignera en file indienne de manière à ce que chacun voit ceux qui sont devant lui et chacun entend ce qui est dit. Ensuite, chacun aura un chapeau déposé sur sa tête, soit rouge, soit noir, sans que la couleur lui soit visible. Pour terminer, les ravisseurs vont commencer à poser la même question, en commençant par la personne à la fin de la queue : **Quelle est la couleur de ton chapeau?** La bonne réponse donne la liberté, la mauvaise l'exécution sur le coup.

Vous avez donc la nuit pour réfléchir à une méthode qui libérerait le plus grand nombre des hotages.

1. Combien de personnes peuvent être sauvées?
2. Comment? C'est-à-dire, écrivez un algorithme pour répondre à la question des raviseurs et expliquez - ou prouvez - pourquoi en l'exécutant le nombre d'exécutés (sic) sera minimisé. L'algorithme comprendra une boucle de 1 à n à l'intérieur de laquelle il y aura, entre autre, la réponse de la i-ème personne.

Answer

1. Au moins $n - 1$ personnes devraient être sauvées, sauf celle qui parle en premier(la dernière personne dans la file); il y a une possibilité de 50% que toutes les personnes pourraient être sauvées.
2.

Strategy:

We use index 1 to n represents the persons from the head to the tail of the line respectively.

Suppose C_i represents the number of the hats with one kind of color in front of the i_{th} person in the line. Obviously, we can define:

$$C_1 = 0$$

. We count RED hat in this problem, yet counting BLACK doesn't change the problem and the solution at all.

Let D_i be the declaration of the i_{th} person about the color of the hat on his/her head.

$$D_i \in \{0, 1\} \tag{0}$$

where 1 means the color being counted, 0 means the other color.

Let H_i be the number of hats with the color being counted in the declaration history(by the persons after current person). For $1 \leq i < n$,

$$H_i = \sum_{j=i+1}^{n} D_j \tag{1}$$

and define:

$$H_n = 0 \tag{2}$$

For the n_{th} person,

$$D_n = \mathbf{1}_{\{C_n \bmod 2 = 1\}} = C_n \bmod 2 \tag{3}$$

For the i_{th} person,

$$\begin{aligned} & D_i = \mathbf{1}_{\{(C_i + H_i - D_n) \bmod 2 \neq D_n\}} \\ \iff & D_i = \mathbf{1}_{\{(C_i + H_i) \bmod 2 = 1\}} \\ \iff & D_i = (C_i + H_i) \bmod 2 \end{aligned} \tag{4}$$

Formular (4) can also be applied to the last person based on our definition by formula (2).

Use this strategy, the last person has 50% chance to be saved or executed, wherase the rest should be saved.

Proof: Step1: If $C_n \bmod 2 = 1$, $D_n = 1$, means the last person saw there are Odd number of RED hat in front of him/her and he/she declare his/her own cat color is also RED. otherwise, he/she will declare BLACK.

Note that $H_{n-1} = D_n$. $D_{n-1} = \mathbf{1}_{\{C_i \bmod 2 \neq D_n\}}$ is very obvious, which means if both C_n and C_{n-1} are the same, then D_{n-1} should be BLACK, otherwise, RED.

Step2: Suppose the strategy is valid for i_{th} person, that is formula (4) is valid for i_{th} person; furthermore, we also have:

$$C_i = C_{i-1} + D_{i-1}, \text{ and } H_i = H_{i-1} - D_i \tag{5}$$

Taking (5) to (4), also based on **(0)**, we then come to:

$$\begin{aligned} D_i &= (C_{i-1} + D_{i-1} + H_{i-1} - D_i) \bmod 2 \\ \Longleftrightarrow 0 &= (C_{i-1} + D_{i-1} + H_{i-1}) \bmod 2 \\ \Longleftrightarrow D_{i-1} &= (C_{i-1} + H_{i-1}) \bmod 2 \end{aligned} \tag{6}$$

This proves that if the strategy is valid for i_{th} person, then it is also valid for $(i-1)_{th}$ person. As we alredy knew it's valid for the last person, we therefore can conclude it's valid for all n persons in the line.

Algorithm:

```
input: n_person
initialization: declare_history = [], a empty list with elements in {0, 1}
output: declare_history
def strategy_for_hat_color_problem():
    for person_index from 0 to n-1:
        n_red_before = number of red(1) hats before current person
        n_red_history = count number of red(1) hats in declare_history
        declare = (n_red_before + n_red_history) % 2
        declare_history.insert(0, declare)

    return declare_history
```

Codes:

In [1]:

```python
import random
```

In [2]:

```python
def declare_of_person(index, hat_colors_before, declare_history):
    """the declaration of the person with index
    params
        index: from 0 to n_person-1, int
        hat_colors_before: true hat colors before current person, [int]
        declare_history: a list of declaration of person standing behind
            current person, should have n_person-1-index elements, the first
            element is the person just behind current person, [int]
    returns
        declaration of current person: 1: red, 0: black, int

        declare_history: alsmo modified.
    """
    # count the number of red hats before himself/herself
    n_red_before = sum(hat_colors_before)
    n_red_history = sum(declare_history) if len(declare_history) > 0 else 0
    # the following three lines are equivalent.
    declare = (n_red_before + n_red_history) % 2
    #declare = 1 if (n_red_before + sum(declare_history)) % 2 == 1 else 0
    #declare = 0 if (n_red_before + sum(declare_history[:-1])) % 2 == declare_history[-1] else 1

    print("Index:{:>2}, red_before: {:<2}, Declare:{}".format(index, n_red_before, declare))
    declare_history.insert(0, declare)
    return declare
```

In [3]:

```
n_person = 20
hat_colors = [random.randint(0, 1) for _ in range(n_person)]
declare_history = []

for i in range(n_person-1, -1, -1): # from the last person to the first
    declare_of_person(i, hat_colors[0:i], declare_history)

print("true color:",hat_colors)
print("   declare:", declare_history)
print("\nCheck if the first {} persons all tell correct hat colors:".format(n_person-1), end = " ")
print(declare_history[:-1] == hat_colors[:-1]) # do not compare the last
```

```
Index:19, red_before: 9 , Declare:1
Index:18, red_before: 9 , Declare:0
Index:17, red_before: 8 , Declare:1
Index:16, red_before: 8 , Declare:0
Index:15, red_before: 7 , Declare:1
Index:14, red_before: 7 , Declare:0
Index:13, red_before: 6 , Declare:1
Index:12, red_before: 6 , Declare:0
Index:11, red_before: 5 , Declare:1
Index:10, red_before: 5 , Declare:0
Index: 9, red_before: 4 , Declare:1
Index: 8, red_before: 4 , Declare:0
Index: 7, red_before: 3 , Declare:1
Index: 6, red_before: 2 , Declare:1
Index: 5, red_before: 2 , Declare:0
Index: 4, red_before: 1 , Declare:1
Index: 3, red_before: 1 , Declare:0
Index: 2, red_before: 1 , Declare:0
Index: 1, red_before: 1 , Declare:0
Index: 0, red_before: 0 , Declare:1
true color: [1, 0, 0, 0, 1, 0, 1, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1]
   declare: [1, 0, 0, 0, 1, 0, 1, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1]

Check if the first 19 persons all tell correct hat colors: True
```