Algorithmique Projet Rapport

1.1 Première étape

Q1. On détermine si toute la ligne peut être colorée en vérifiant la valeur de T(M-1, k), où M est le nombre de colonnes et k est la longueur de la séquence. Si T(M-1, k) est vrai, cela signifie que toute la ligne li peut être colorée en utilisant la séquence entière.

Q2.

1. **Cas ℓ = 0 :** Lorsque ℓ = 0, il n'y a aucun bloc noir à placer. Par conséquent, T(j, ℓ) est toujours vrai (true) dans ce cas. Cela est dû au fait qu'une séquence vide peut toujours être placée, quel que soit la valeur de j.

2. **Cas ℓ ≥ 1 :**
   a) j < sℓ - 1 : Lorsque la valeur de j est inférieure à la longueur du bloc ℓ dans la séquence moins un, cela signifie que le nombre actuel de cases est insuffisant pour accueillir ce bloc noir. Par conséquent, dans ce cas, T(j, ℓ) est faux (false).

   b) j = sℓ - 1 : Lorsque j est égal à la longueur du bloc ℓ dans la séquence moins un, cela signifie que le nombre actuel de cases peut juste accueillir le bloc ℓ. Donc si l = 1, alors T(j, 1) est vrai. Sinon faux.

Q3. Car si on peut placer les ℓ premiers blocs noirs dans un cas où il y a une case en moins (j - 1), alors on peut certainement les placer avec le nombre actuel de cases. Ou, si on peut placer les ℓ - 1 premiers blocs noirs avec j - sℓ - 1 cases, alors on peut aussi les placer dans le cas actuel.

Q4. Voir Util.java

1.2 Généralisation

Q5.

- l = 0 : Si la ligne colorée contient des blocs noirs, alors c'est faux. Complexité : O(n)
- l ≥ 1 && j < sℓ - 1 : Faux. Complexité : O(1)
- l ≥ 1 && j = sℓ - 1 : l == 1 et la ligne colorée ne doit pas contenir de blocs blancs. Complexité : O(n)
- l ≥ 1 && j > sℓ - 1 : Si le dernier bloc est blanc : T(j - 1, ℓ). Complexité : O(1) Si le dernier bloc est noir : Si les cases du j - sℓ au j sont noires et que la case j - sl - 1 est blanche : T(j - sℓ - 1, ℓ - 1). Sinon : Faux. Complexité : O(1) Si le dernier bloc est vide : T(j - 1, ℓ) || T(j - sℓ - 1, ℓ - 1).

Q6. O(n2^j)

Q7. Voir Util.java.

1.3 Propagation

Q8. O(mnO(canColorEntireRow2))

Q9. Voir Util.java

1.4 Tests

Q10.

instant	temps(ns)
1	1322500
2	31575000
3	45291800
4	95575400
5	35931000
6	111230500
7	70432000
8	127446700
9	10014380300
10	12758393400

Instant 9:

Q11. Took 704000 ns
ne sais pas
Empty Empty Empty Empty
Empty Empty Empty Empty

Comme la stratégie adoptée est de ne colorier que si nous sommes absolument certains de la couleur, chaque case de l'instant11, prise individuellement, peut être soit noire soit blanche. Ainsi, avec la stratégie actuelle, il est impossible de déterminer une méthode de coloriage.

2 Méthode complète de résolution

Q12.
A(n) = 2A(n - 1) + 1 r^2 - 2r - 1 = 0 r = 2.414 A(n) = O(2.414^n)

Car dans chaque appel récursif, on a O(colorierEtPropager()) = O(mnO(canColorEntireRow())) . Alors on a O(O(colorierEtPropager()) * 2.414^n)

2.1 Implantation et tests

Q13. Voir Util.java.
White Black Black White
Black White White Black

Q14.

instant	temps sans énumération	temps avec énumération
12	64749900	398451300
13	1344410800	178435017100
14	55029600	6826201100
15	73411700	4270023900
16	timeout	timeout

Instant 15 sans énumération:

Instant 15 avec énumération: