Nom i cognoms:

Algorísmia i Programació 1, GCED, 16 de gener de 2020

Possibles solucions

1 Un xic de Python

- ```
  0 0 0
  2 0 2
  ```

- ```
  ["unicorn", "rainbow"]
  ```

- ```
  2
  4
  [[2, 3, 2, 3], [2, 3, 2, 3, 2, 3, 2, 3]]
  ```

- Donat un nombre natural n parell, *norf*(n) escriu totes les seqüències d'n bits amb $n/2$ zeros i $n/2$ uns.

- Donat un nombre natural n, *fog*(n) escriu totes les seqüències d'n bits que no tinguin més de tres zeros seguits ni més de dos uns seguits.

2 Especificació i invariant

- ```
  *******aaaaaaaZZZZZZZ.......**********
         |      |       |      |
        low     i       j     high
  ```

  - $low - 1 \leq i < j < high$.
  - Els elements a $v[low \ldots i]$ són més petits o iguals que p.
  - Els elements a $v[i + 1 \ldots j - 1]$ són més grans que p.

- La funció permuta els elements de $v[low \ldots high]$ i retorna un index k de tal manera que:

  - $v[k]$ conté l'element que hi havia inicialment a $v[high]$.
  - $v[low \ldots k]$ conté elements més petits o iguals que $v[k]$.
  - $v[k + 1 \ldots high]$ conté elements més grans que $v[k]$. Aquest segment podria ser buit ($k = high$).

3 Omplir matrius

- **Precondició:** n i m representen el nombre de files i de columnes d'una matriu, i les variables f i c indexen el darrer element escrit a la matriu.

  Postcondició: f i c indexen el següent element que ha de ser escrit a la matriu.

  Retorna cert si la matriu encara no estaà plena. Retorna fals si la matriu ja està plena (els valors de f i c son irrellevants en aquest cas).

- ++c; // *Incrementem columna*
  if (c == m) { // *Si estavem a la darrera columna, anem a la seguent fila*
  c = 0;
  ++f;
  }
  return $f < n$;

- **if** (f%2 == 0) {
  if (++c == m) {
  −−c; ++f;
  }
  } **else** {
  if (−−c < 0) {
  c = 0; ++f;
  }
  }
  return $f < n$;

- **if** ((f + c) % 2 == 0) { // *diagonal que baixa*
  if (f == n − 1) {
  ++c;
  } **else** {
  ++f;
  if (c > 0) {
  −−c;
  }
  }
  } **else** { // *diagonal que puja*
  if (c == m − 1) {
  ++f;
  } **else** {
  ++c;
  if (f > 0){
  −−f;
  }
  }
  }
  return $f \neq n$ **and** $c \neq m$;

4 Eficiència

Alba	$O(n^2)$
Bruna	$O(2^n)$
Clara	$O(n)$
Dafne	$O(n \log n)$

← el més eficient

Càlculs:

Ada: $T(n) = 1 + 2 + 3 + \cdots + n = \frac{n(n-1)}{2}$ $T(n) \in O(n^2)$

Bruna: $T(n) = 1 + 2 + 4 + \cdots + 2^{n-1} = 2^n - 1$ $T(n) \in O(2^n)$

Clara: $T(n) = n + \frac{2n}{3} + \frac{4n}{9} + \frac{8n}{27} + \cdots \leq n \sum_{i=0}^{\infty} \frac{2^i}{3^i} = 3n$ $\qquad T(n) \in O(n)$

Es tracta d'una sèrie geomètrica amb raó $r = \frac{2}{3} < 1$.
La seva suma convergeix a una constant.

Dafne: El bucle extern s'executa n vegades, mentre que el bucle intern s'executa $\log_2 n$ vegades. Per tant:

$T(n) = n \cdot \log_2 n$ $\qquad T(n) \in O(n \log n)$