

Aplicação de Multithreads

MC504 - Sistemas Operacionais





Áureo Henrique



Jonas Cardoso

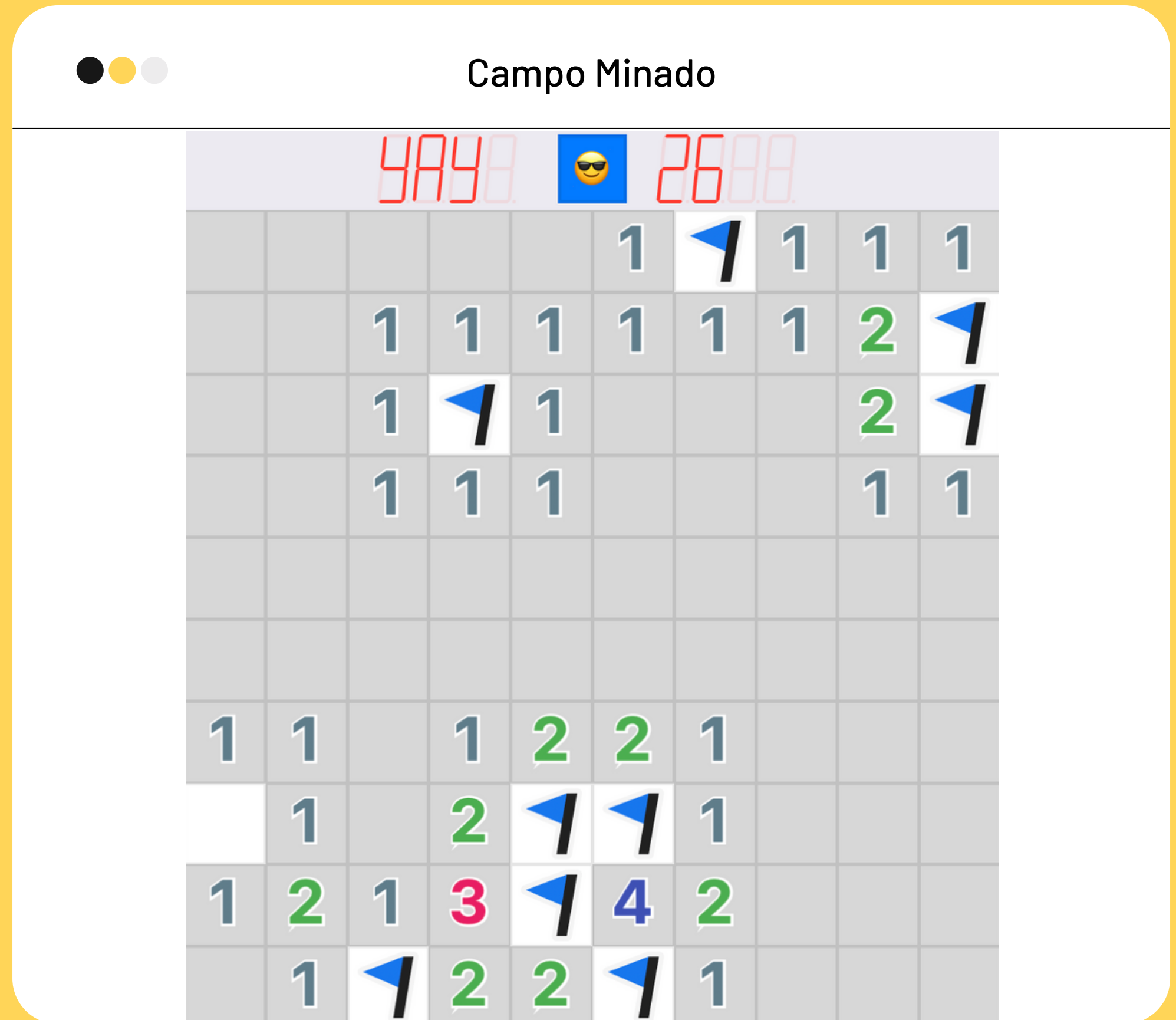


Lethycia Maia

Mecânica do Jogo

Como jogar

- 1 jogador
- Revelar minas sem detoná-las através dos campos de dicas





A ideia do projeto é desenvolver um programa que recebe uma matriz de minas e retorna uma matriz de dicas usando Threads

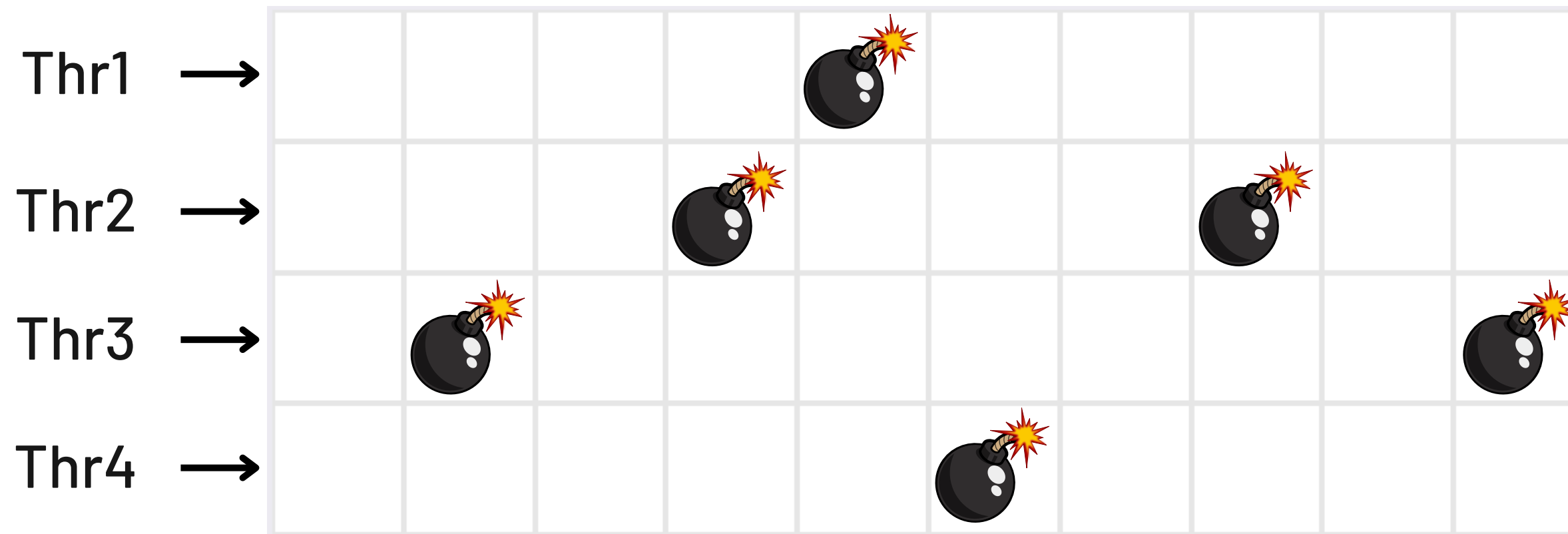










Uso das threads

- Cria-se uma thread para cada linha da matriz
- Cada thread percorre a linha e calcula a quantidade de bombas em volta de cada posição sem bomba e insere o valor nesse campo

Uso das threads



Uso das threads

Thr1 →	0	0	1	2		1	1	1	1	0
Thr2 →	1	1	2		2	1	1		2	1
Thr3 →	1		2	1	2	1	2	1	2	
Thr4 →	1	1	1	0	1		1	0	1	1

Função dicas

```
void* dicas(void *i)
{
    int linha = (int) i;

    for (int j = 0; j < L[1]; j++)
    {
        if(matriz[linha][j] == 'x')
            continue;

        int tmp = 0;

        for (int a = -1; a < 2; a++)
        {
            for (int b = -1; b < 2; b++)
            {
                if((a == 0) && (b == 0))
                    continue;
                else if (validar(a + linha, b + j) && matriz[a + linha][b + j] ==
                'x')
                    tmp++;
            }
        }
        matriz[linha][j] = tmp + '0';
    }

    return (void *) 0;
}
```



```
L = malloc (2 * sizeof (int));
scanf("%d %d ", &L[0], &L[1]);

matriz = malloc (L[0] * sizeof (char*));
for (int i = 0; i < L[0]; i++)
    matriz[i] = malloc (L[1] * sizeof (char));

preencher();

/// criação das threads
pthread_t thr[L[0]];
for (int i = 0; i < L[0]; i++)
    pthread_create(&thr[i], NULL, &dicas, (void*) i);

/// finalização das threads
for (int i = 0; i < L[0]; i++)
    pthread_join(thr[i], NULL);

imprimir();
```

Criação das Threads



Execução 1 - Entrada

8 8

0 0 0 0 0 0 0 0

0 0 X 0 0 0 0 0

0 0 0 0 0 0 X 0

0 0 0 0 X 0 0 0

0 0 X 0 0 X 0 0

0 0 0 0 0 0 0 0

0 0 X 0 0 X 0 0

0 0 0 0 0 0 0 0



Execução 1 - Saída

0 1 1 1 0 0 0 0

0 1 x 1 0 1 1 1

0 1 1 2 1 2 x 1

0 1 1 2 x 3 2 1

0 1 x 2 2 x 1 0

0 2 2 2 2 2 2 0

0 1 x 1 1 x 1 0

0 1 1 1 1 1 1 0



Execução 2 - Entrada

6 12

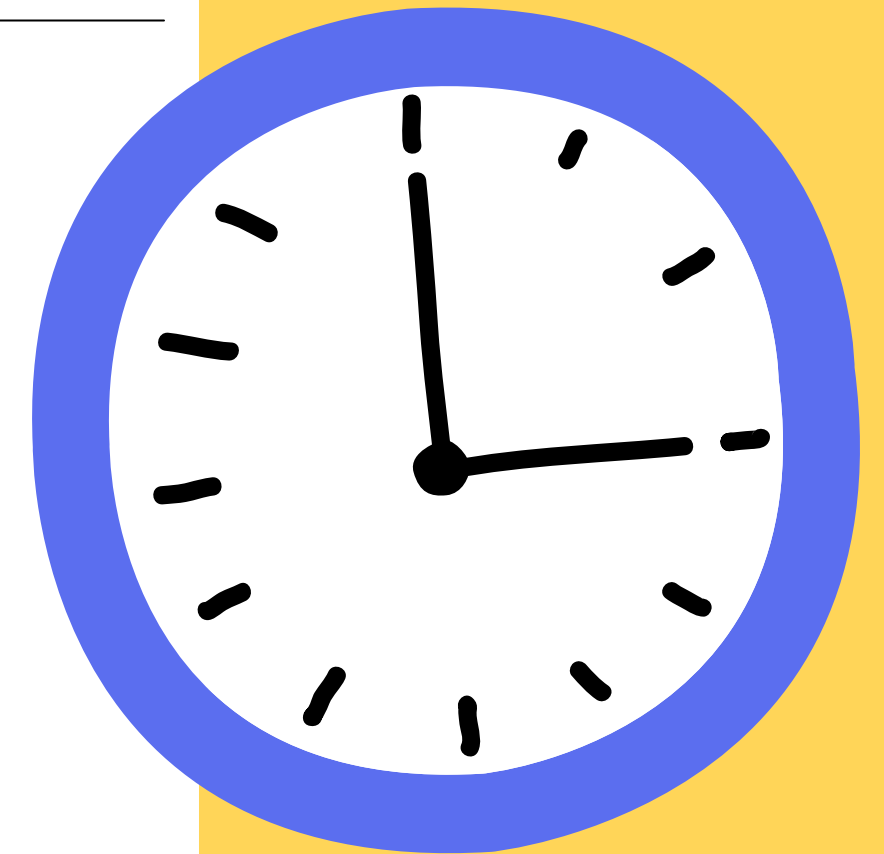
O X O O X O O O O X O O
O O X O O X O O O O O O
O O O O X O O O X O O X
O X O O X O O O O O X O
O O O O O X O X O O X O
O O O O O O O O O X O O



Execução 2 - Saída

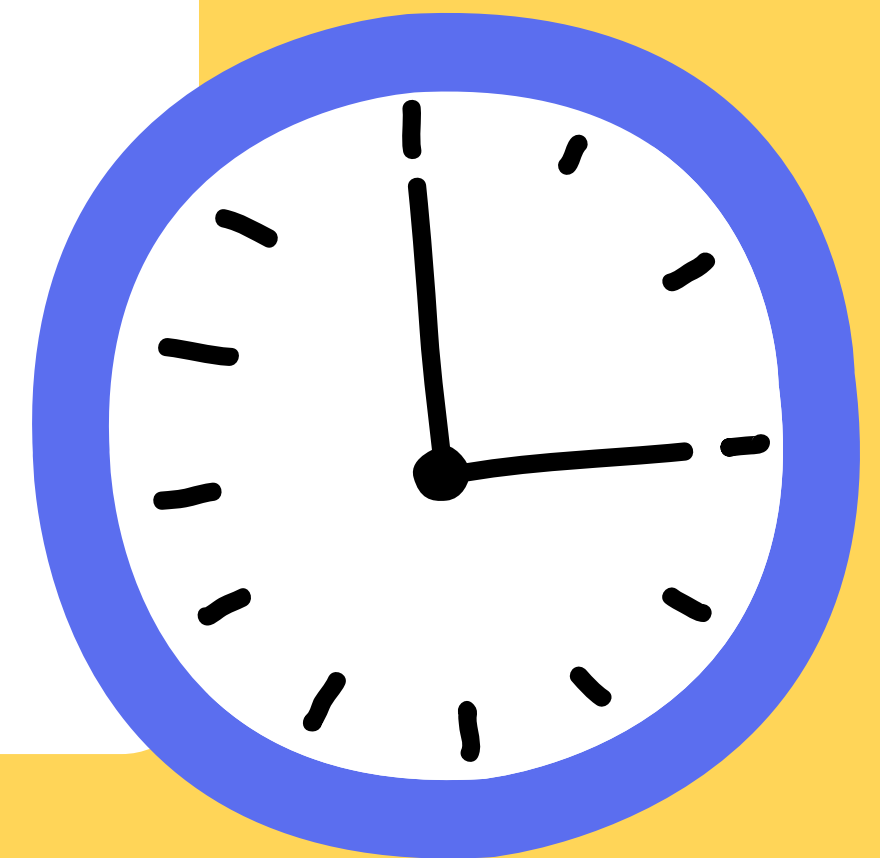
1 x 2 2 x 2 1 0 1 x 1 0
1 2 x 3 3 x 1 1 2 2 2 1
1 2 2 3 x 3 1 1 x 2 2 x
1 x 1 2 x 3 2 2 2 3 x 3
1 1 1 1 2 x 2 x 2 3 x 2
0 0 0 0 1 1 2 1 2 x 2 1

**Mas e o
tempo?**



teste de tempo de CPU

- Execução 1: matriz 8x8
 - sem thread: 0.00020 s
 - com thread: 0.00100 s
- Execução 2: matriz 6x12
 - sem thread: 0.00018 s
 - com thread: 0.00081 s
- Execução 3: matriz 1944 x 1944
 - sem thread: 0.45594 s
 - com thread: 0.84501 s



● ● ● teste de tempo de CPU

Nessa aplicação o uso de threads
foi mais custoso!





Valeu!