# Relatório do EP2

## MAC0422 - Sistemas Operacionais (2024)

Gabriel Takeuchi - NUSP: 13671636

# Os ciclistas

- São $k$ ciclistas na corrida, e portanto $k$ threads rodando a função `f_ciclista`.
- Cada ciclista é uma `struct Ciclista` com os campos:
  - `int id`: identificador único (de 0 a $k$)
  - `int posicao_x, posicao_y`: metro e raia atual
  - `int velocidade`: velocidade atual
  - `int voltas`: quantas voltas deu em relação a ele
  - `int colocacao`: colocação final (alterada apenas quando ganha ou quebra)
  - `double tempo_final`: em quanto tempo em segundos ganhou a corrida (não é calculado se ele quebrou)
  - `bool na_corrida`
  - `bool atualizou_posicao`: usado para coordenar se o ciclista deve andar em um ciclo

# A pista

- A `pista` é uma matriz de dimensões $d \times 10$, sendo $d$ o tamanho da pista em metros. Acessamos o elemento no metro $x$ e raia $y$ com `pista[x][y]`.
- Mentalize a pista como o primeiro quadrante de um plano cartesiano, onde a origem é o canto inferior esquerdo.
- Para acessar a `pista`, utilizamos um mutex `mutex_pista`.

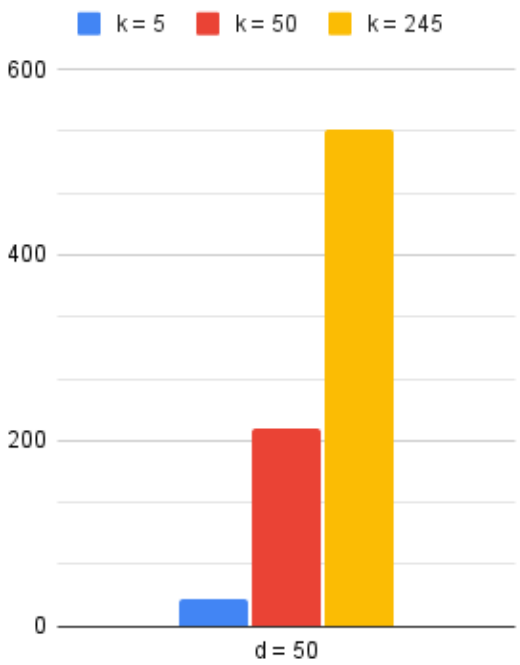
# A largada

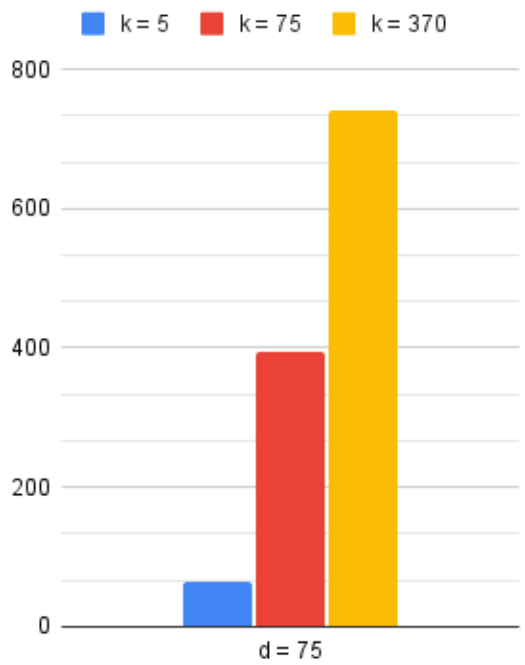
- Seja $k$ o número de ciclistas.
- A largada é organizada com grupos de 5 ciclistas lado-a-lado em cada metro
- Os ciclistas de cada grupo são inicializados nas raias 0 à 4.
- O primeiro grupo é inicializado no metro 0, o segundo no metro 1, ..., até o metro $\lceil \frac{k}{5} \rceil$.
- Resumo: temos $\lceil \frac{k}{5} \rceil$ colunas de 5 ciclistas.

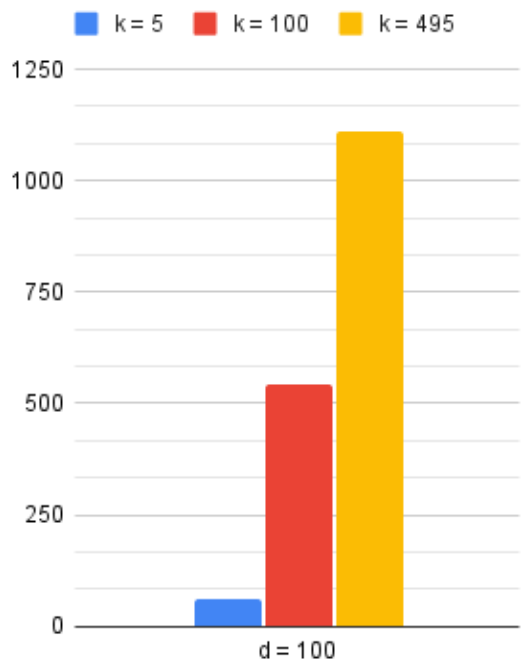
# Loop principal

- Utilizamos 2 barreiras de sincronização:
  - A `barreira_ciclistas_andaram` sincroniza as threads para executarem.
  - A `barreira_impressao` impede que as threads executem enquanto a `main` imprime e processa os dados.
- O loop principal segue a seguinte lógica:
  1. As threads executam e os ciclistas fazem suas operações.
  2. As threads sincronizam na `barreira_ciclistas_andaram`, e esperam na `barreira_impressao`.
  3. O loop principal verifica quem ganhou e quebrou e imprime a situação atual da corrida.
  4. O loop principal sincroniza na `barreira_impressao` e, caso exista ciclista na pista, libera as threads para a próxima iteração.

# Gráficos de consumo de tempo




Obs.: As medições foram feitas em segundos.

# Análise do consumo de tempo

- Consideramos $d = 50$ pequeno, $d = 75$ médio e $d = 100$ grande. Isto pois os tempo de execução ficariam absurdamente inviáveis de calcular para o intervalo $100 <= d <= 2500$.
- Os gráficos para todos os $d$ seguem uma tendência exponencial.