EP2 – MAC0422 SISTEMAS OPERACIONAIS

INTEGRANTES / NUSP:

BRUNO MAZETTI SAITO | 11221838

WILLIAN HIROSHI TAKIHI | 11221755

ROTEIRO

- Decisões de Projeto;
- Testes;
- Gráficos;
- Resultados e Conclusão.

Decisões de Projeto

- Tratamento de *Deadlock*;
- Disposição Inicial dos Ciclistas;
- *Threads* do Programa;
- Seção Crítica;
- Biblioteca Adicional;
- Reorganização da Pista;
- Barreira de Sincronização;
- Modo de Eliminação dos Ciclistas.
- Modo de Ultrapassagem.
- Últimas Duas Voltas

Tratamento do *Deadlock*

- O programa exige que, em cada faixa da pista, exista pelo menos um espaço livre para que os ciclistas possam andar de maneira adequada.
- A disposição inicial dos ciclistas escolhida no EP é feita para prevenir que toda uma faixa na pista esteja cheia.
- O modo de ultrapassagem também previne que esse tipo de situação aconteça durante a execução do programa.

Disposição Inicial dos Ciclistas

- No início da corrida os ciclistas são colocados na pista com a ordem aleatória, seguindo o seguinte padrão:
  - Os números representam cada um dos ciclistas;
  - A corrida ocorre da esquerda para direita.

REPRESENTAÇÃO DA PISTA					
15		14		23	
5		11		3	
16		20		16	
26		24		18	
28		27		6	
	9		4		7
	17		25		12
	13		19		22
	21		1		2
	12		10		8

Threads do Programa

- O EP consiste das seguintes *threads* principais:
  - Coordenador;
  - "*workCiclista*".
- O coordenador é encarregado de:
  - sincronizar as *threads* de cada um dos ciclistas;
  - eliminar os últimos ciclistas de cada volta;
  - reorganizar a pista a cada volta;
  - organizar a matriz de semáforos para controle das seções críticas do programa.

Threads do Programa

- O “*workCiclista*” representa cada um dos ciclistas, desempenhando as funções de:
  - andar na pista;
  - atualizar a sua velocidade quando necessário;
  - verificar e quebrar o ciclista;
  - colocá-lo no vetor de listas ligadas de colocações de cada volta.

Seção Crítica

- As seções críticas do EP consistem em duas partes:
  - pista de ciclistas.
  - vetor de listas ligadas de colocações de cada volta;
- Na pista de ciclistas, foi utilizado uma matriz de semáforos de tamanho idêntico ao da pista. Cada semáforo é utilizado para alterar as posições da pista de maneira atômica. Assim, nenhum ciclista disputará pelo mesmo espaço.
- É necessário um semáforo adicional para incluir no vetor de listas ligada para que não ocorra condição de corrida entre ciclistas.

Biblioteca Adicional

- A biblioteca *lista* foi criada para manejar as listas ligadas presentes no vetor de listas ligadas de colocações de cada volta.

Reorganização da Pista

- A cada volta realizada, a pista é reorganizada da seguinte maneira: os ciclistas são colocados nas faixas mais internas de modo a deixar pelo menos um espaço livre em cada faixa da pista;

Barreira de Sincronização

- O programa ocorre em turnos, alternando entre o coordenador e as *threads* dos ciclistas.
- O coordenador utiliza um vetor de estados chamado *arrive* para esperar a execução das *threads*, para que em seguida ele possa realizar suas funções*;*
- O coordenador retoma a execução das *threads* com outro vetor de estados denominado *continuar.*
- As checagens dos vetores de estado são feitas em intervalos de tempo de 1 milissegundo.

Modo de Eliminação dos Ciclistas

- A cada volta par completada, o coordenador utiliza o vetor de listas ligadas de colocações para retirar o ciclista que completou a volta em último.
- A cada volta múltiplo de 6 completada por um ciclista, ele próprio decide se quebrará ou não e se retira da corrida caso necessário.
- Caso um ciclista se apresente em uma volta posterior à volta em que deveria ter sido eliminado, por conta de retardatários, e venha a quebrar, o coordenador não elimina nenhum ciclista. Pois, é suposto que ele deveria ter sido eliminado por estar em último.
- Caso dois ciclistas ou mais terminem uma volta por último ao mesmo tempo, cabe ao escalonador decidir quem será eliminado.

Modo de Ultrapassagem

- Cada *thread* ciclista espera, ou a posição imediatamente à frente estar vazia, ou o ciclista à frente executar para poder realizar uma ultrapassagem ou andar.
- Os ciclistas realizam ultrapassagens para faixas mais externas da pista, de modo a deixar pelo menos um espaço vazio em cada faixa. Isso para evitar casos de *deadlock*.
- No programa, é permitido que ciclistas de mesma velocidade ultrapassem.

Últimas Duas Voltas

- Como ciclistas podem quebrar no meio da corrida, não é possível saber quais são as duas últimas voltas.
- Apenas quando há 5 ciclistas e não podem ocorrer mais quebras, podemos determinar em qual volta a corrida acabará.
- Em casos em que o primeiro colocado já completou ou está correndo as últimas duas voltas, a mudança de velocidade para 90km/h é ignorada.
- Caso ainda venham a completar as duas últimas voltas, um dos dois primeiros colocados pode ser sorteado para correr a 90km/h.

Testes

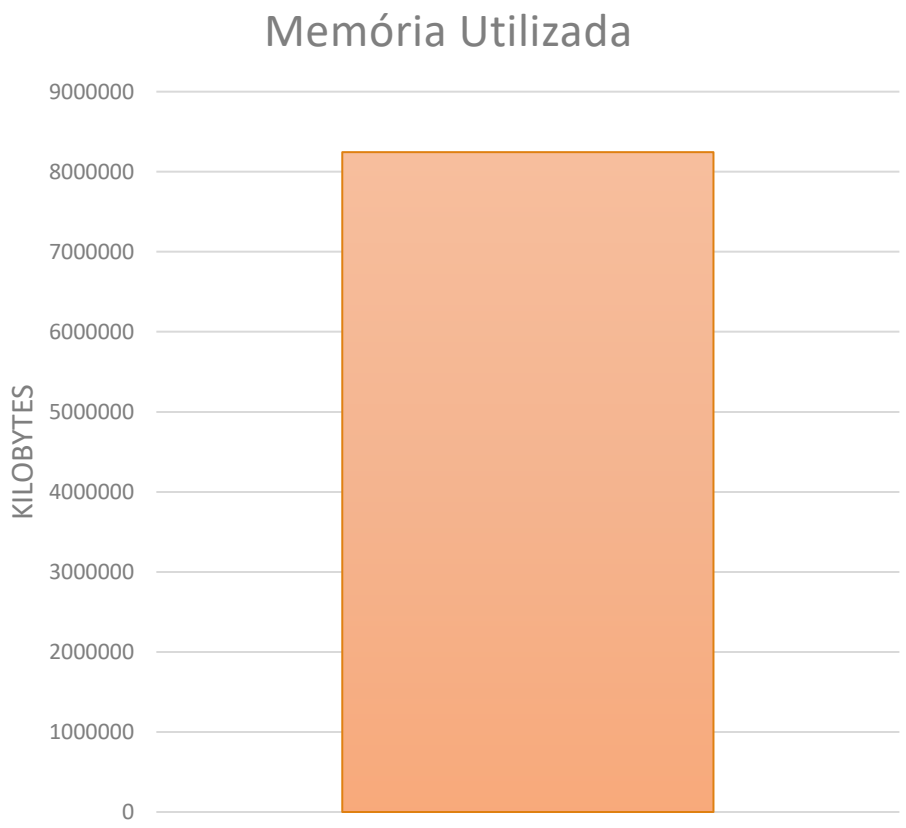
- Os testes foram feitos com os seguintes valores de D (comprimento da pista) e N (número de ciclistas):
  - Valores de D
    - Pequeno: 250
    - Médio: 400
    - Grande: 500
  - Valores de N
    - Pequeno: 50
    - Médio: 500
    - Grande: 2000
- Nos testes, foram registrados o uso de memória e o tempo de execução *(wall-clock time).*
- Para testar em relação à mudança de D, foi utilizado o número médio de ciclistas.
- Para testar em relação à mudança de N, foi utilizado o comprimento médio de pista.

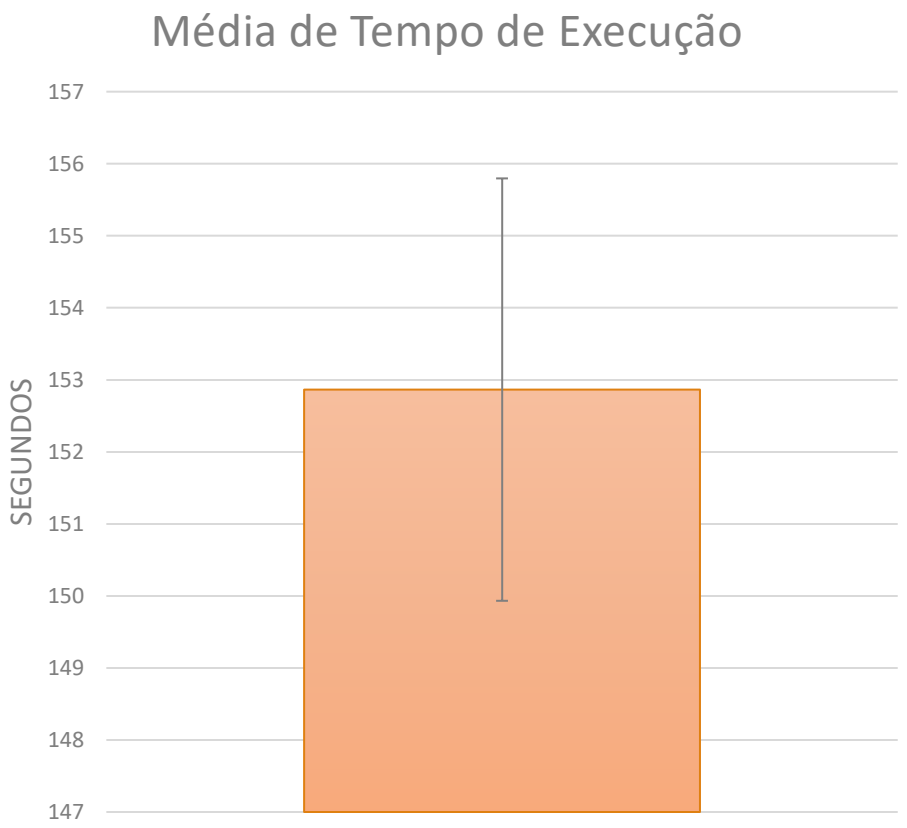
Testes

- Para cálculo do uso de memória, foi utilizado o comando: pmap <PID_DO_PROCESSO>.
- Para cálculo do tempo, foi utilizado o comando: time ./ep2 <D> <N>
- Foram realizados 30 testes de cada uma das configurações.
- Os testes foram realizados em duas máquinas diferentes, consequentemente, é possível que os resultados apresentem diferenças maiores entre si.

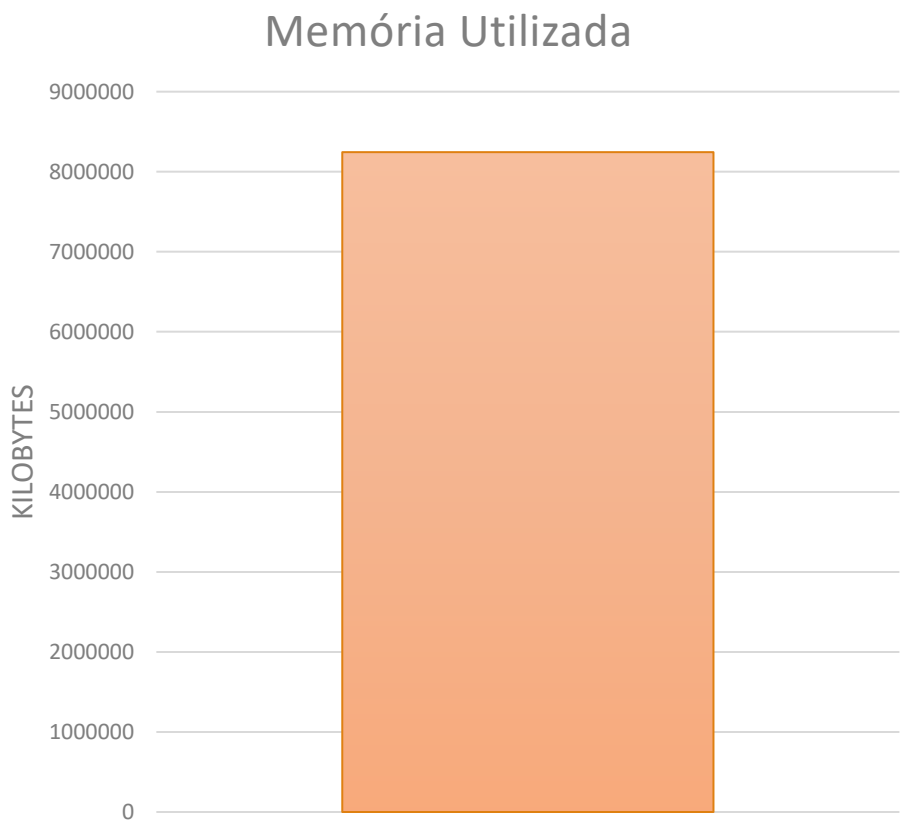
- Observação: Baseado na escolha dos valores de D e N realizadas nos testes, os gráficos 2 e 5 acabaram por ser iguais. Mantivemos ambos para facilitar a visualização dos resultados.

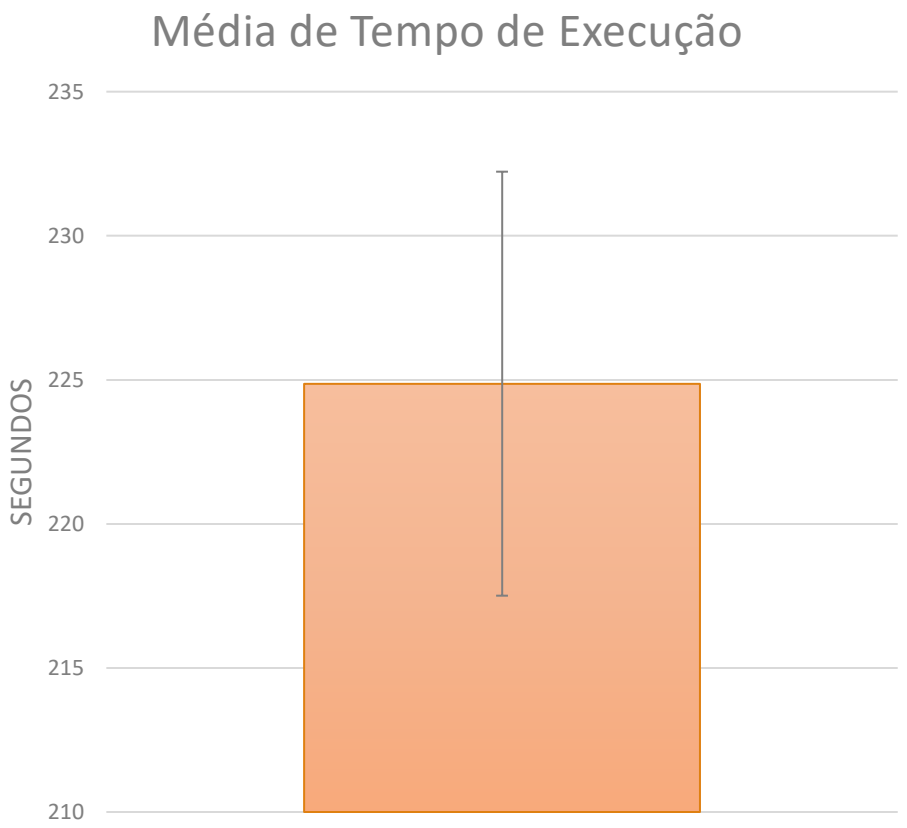
Gráficos 1 – D = 250 e N = 500



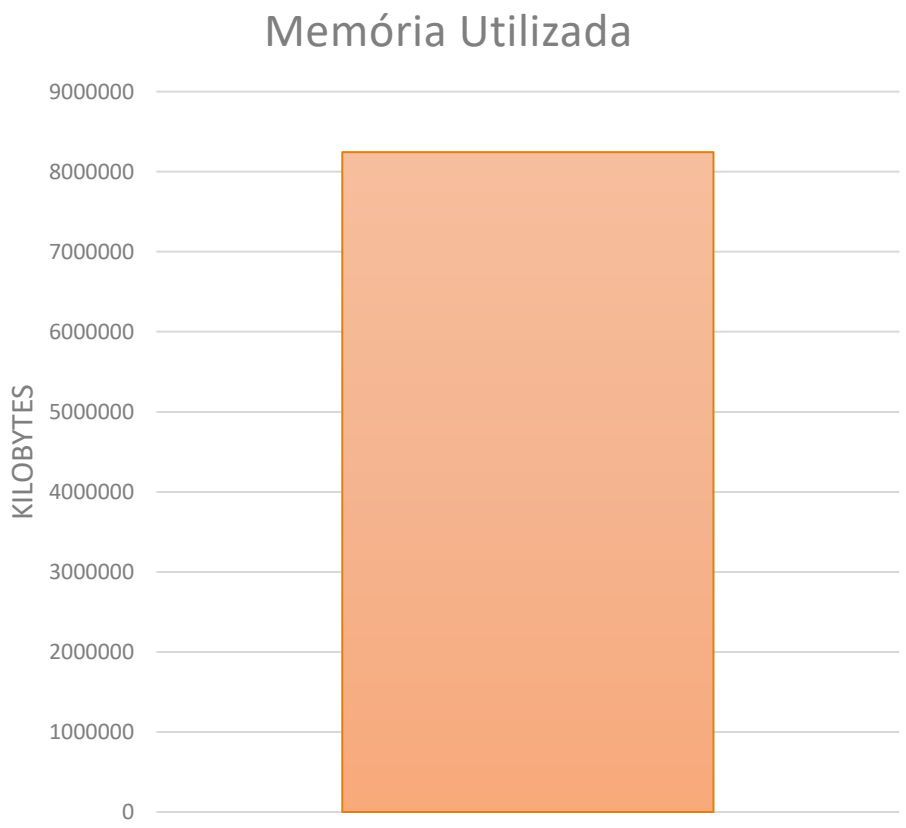


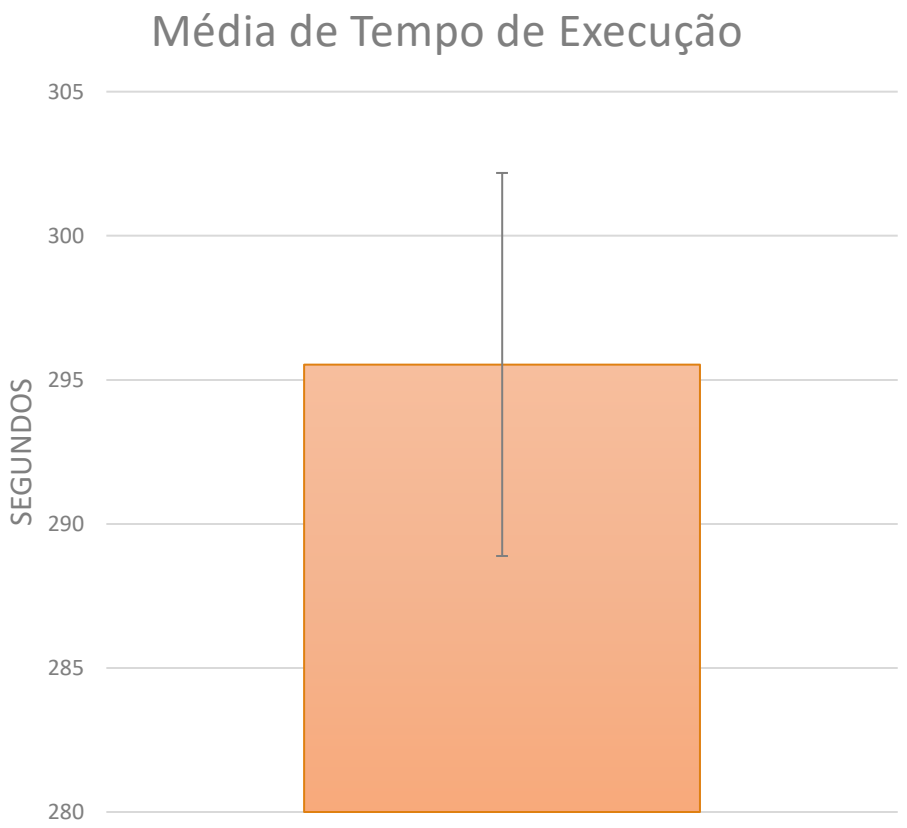
Gráficos 2 – D = 400 e N = 500



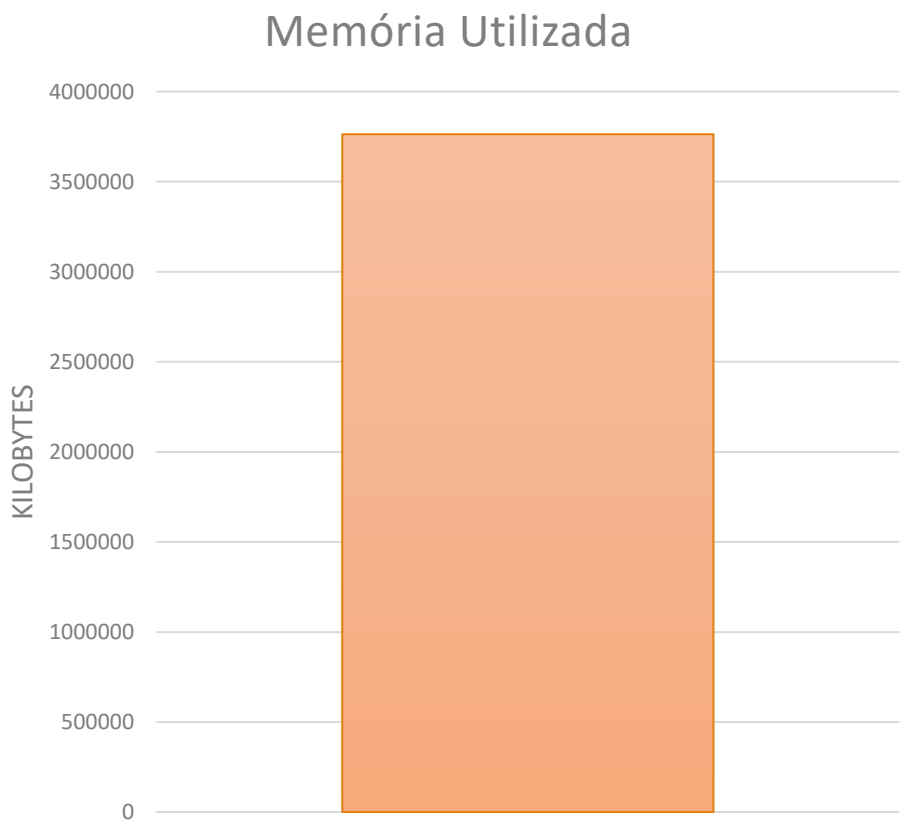


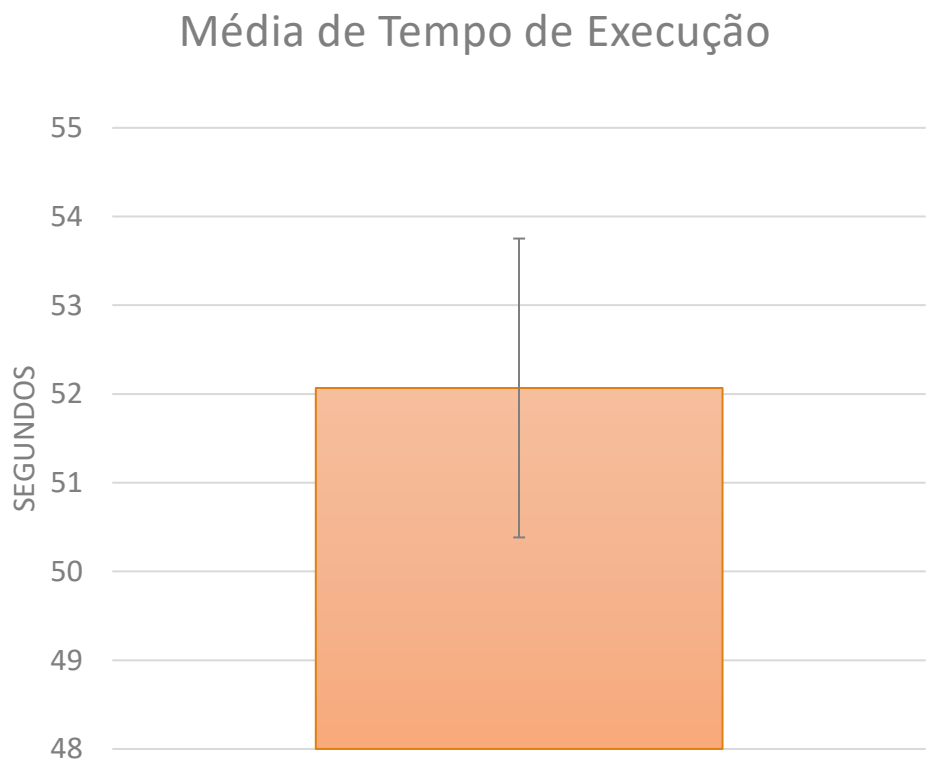
Gráficos 3 – D = 500 e N = 500



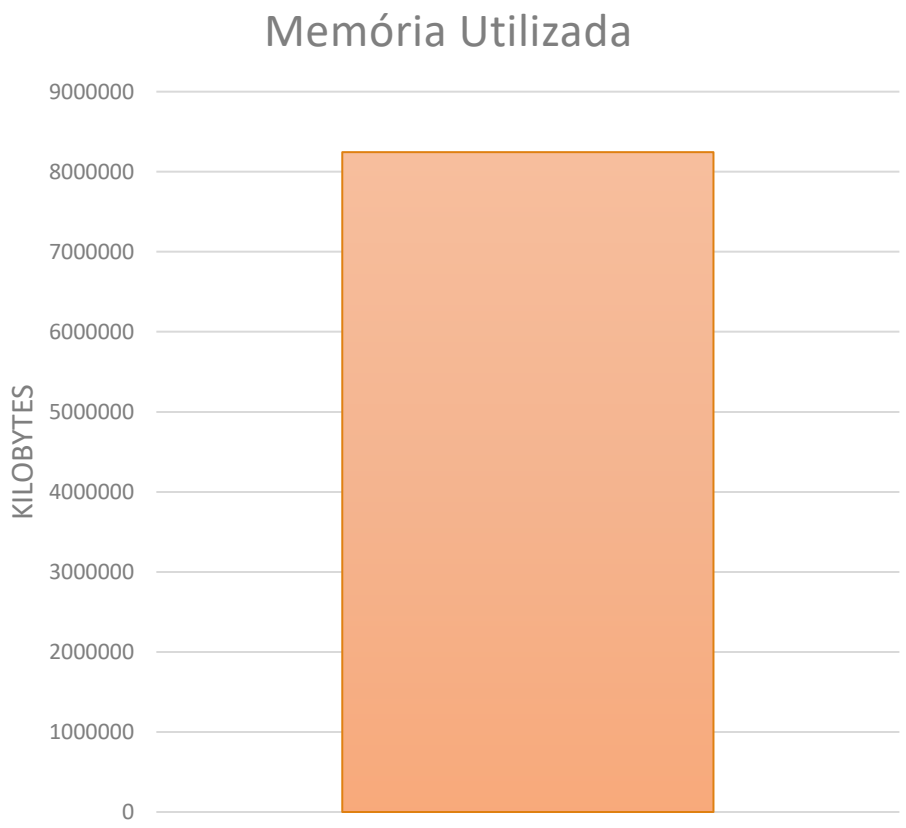


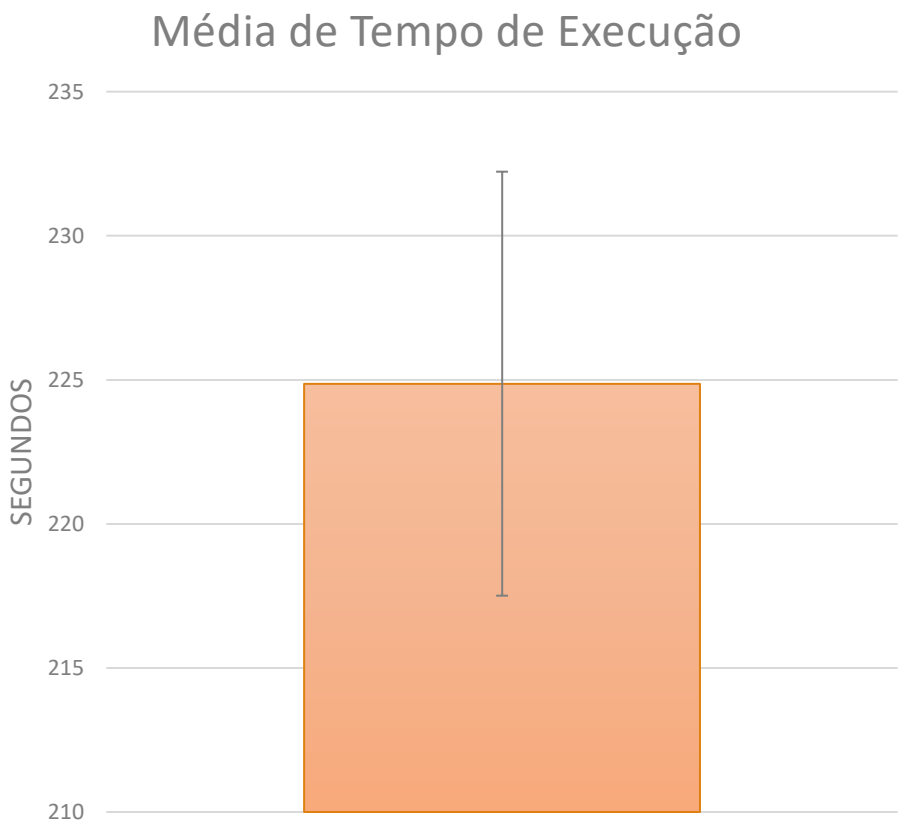
Gráficos 4 – D = 400 e N = 50



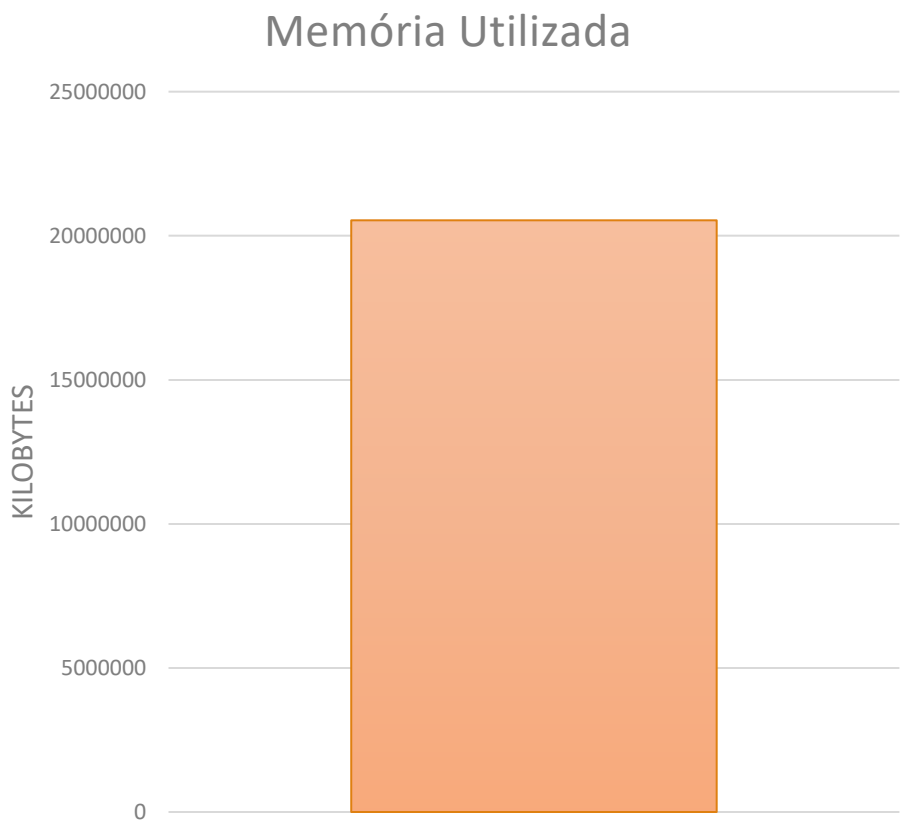


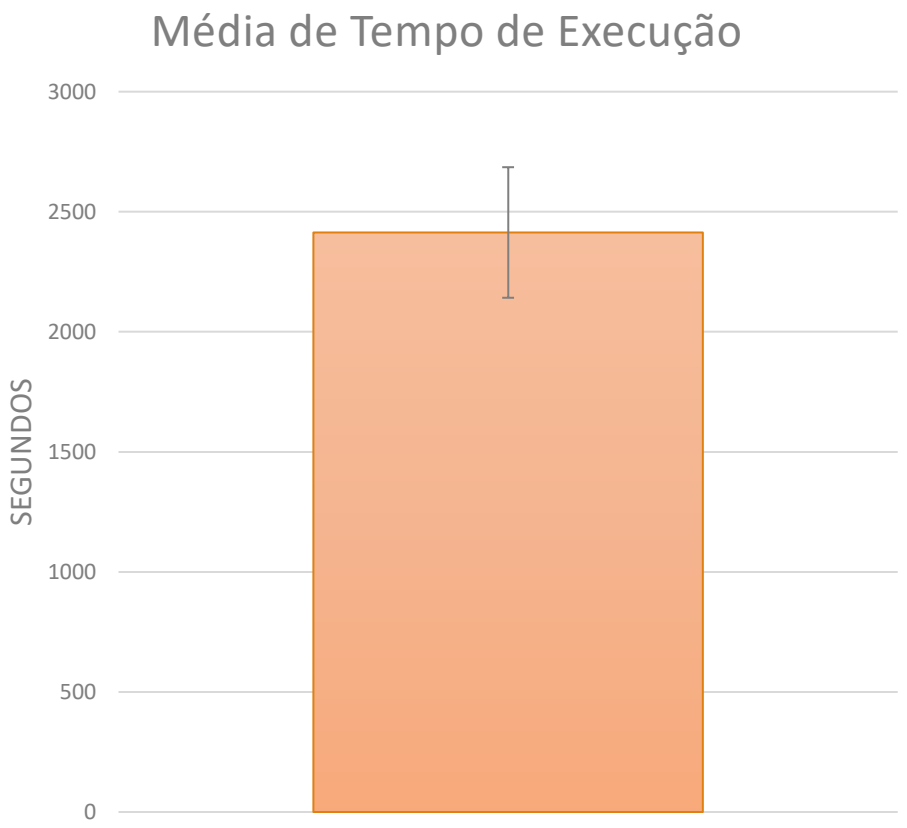
Gráficos 5 – D = 400 e N = 500





Gráficos 6 – D = 400 e N = 2000





Resultados e Conclusão

- O resultado esperado do programa é:
  - O tempo de execução é diretamente proporcional ao comprimento da pista, haja vista que é necessário executar mais iterações para completar as voltas e, eventualmente, retirar os ciclistas da corrida.
  - O uso de memória é diretamente proporcional ao número de ciclistas, pois é necessário criar mais variáveis e *threads* para representação de cada ciclista.
- Podemos observar esses dois fenômenos nos testes realizados:
  - A medida que o comprimento da pista aumenta, o uso de memória cresce de maneira pouco perceptível, mas o aumento no tempo de execução é significante.
  - O aumento do número de ciclistas resultou em mais consumo de memória e tempo de execução.