EP3 - MAC0422

Nome: Renan Ryu Kajihara
NUSP: 14605762

Variáveis auxiliares usadas em cada algoritmo

Em todos os algoritmos de alocação, tais variáveis auxiliares são utilizadas:

- int linha_comeco: representa o número da primeira linha do espaço livre analisado no momento;
- int tamanho_livre: representa o tamanho do espaço livre analisado no momento;
- int primeiro_indice_livre: representa o número do primeiro índice livre do espaço livre analisado no momento;
- int linha_atual: representa o número da linha analisado no momento atual.
- int offset: representa o deslocamento (em bytes) dentro do arquivo PGM de saída, indicando o início da linha que será modificada.

No Next-Fit, são utilizadas adicionalmente as seguintes variáveis:

- int linha_nf: variável global que representa a linha onde a busca terminou ao alocar espaço para o último processo;
- int indice_nf: variável global que representa o índice onde a busca terminou ao alocar espaço para o último processo;
- int offset_nf: variável global que representa a posição no arquivo PGM que o Next-Fit deve iniciar sua próxima busca.

No Best-Fit, são utilizadas adicionalmente as seguintes variáveis:

- int minimo_tamanho_livre: representa o tamanho do menor espaço livre que é maior ou igual ao tamanho da requisição;
- int linha_minimo_tamanho_livre: representa o número da primeira linha do menor espaço livre que é maior ou igual ao tamanho da requisição;
- int primero_indice_minimo_tamanho_livre: representa o primeiro índice livre do menor espaço livre que é maior ou igual ao tamanho da requisição.

No Worst-Fit, são utilizadas adicionalmente as seguintes variáveis:

- int maximo_tamanho_livre: representa o tamanho do maior espaço livre encontrado;
- int linha_maximo_tamanho_livre: representa o número da primeira linha do maior espaço livre encontrado;
- int primeiro_indice_maximo_tamanho_livre: representa o primeiro índice livre do maior espaço livre encontrado.

EXPERIMENTOS

Entrada dos experimentos

Para os experimentos, foram utilizados 4 arquivos trace diferentes:

- trace-firstfit: possui uma requisição de 256 unidades de alocação, seguida de uma requisição de 100 unidades de alocação, seguida de 137 requisições de 256 unidades de alocação.
- trace-nextfit: possui 1797 requisições de 20 unidades de alocação.
- trace-bestfit: possui 138 requisições de 256 unidades de alocação, seguida de 42 alocações de tamanhos variados.
- trace-worstfit: possui requisições de 50 e 256 unidades de alocação, alternadamente.

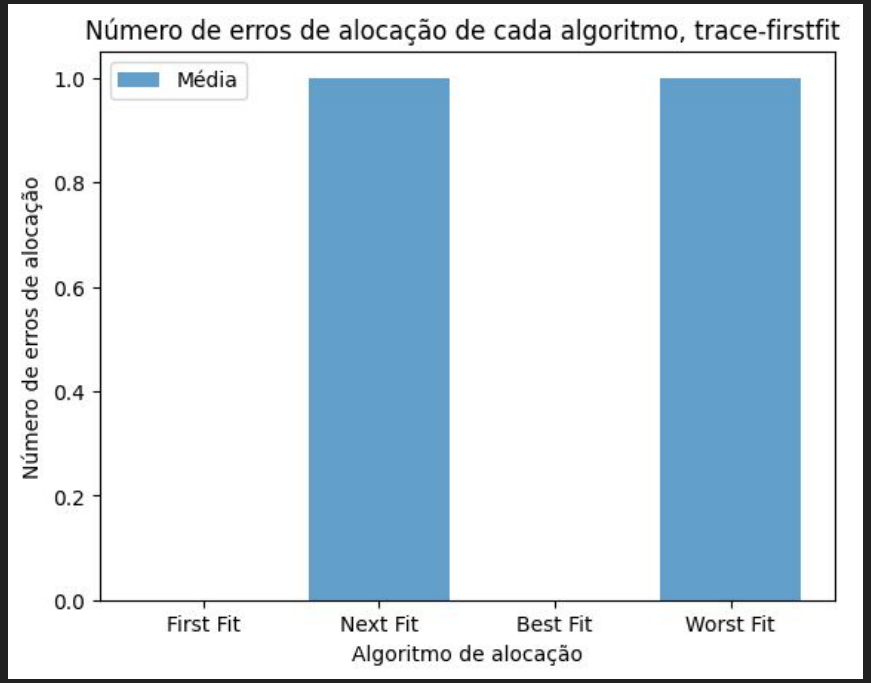
Além disso, o arquivo PGM de entrada utilizado foi o ep3.exemplo01.pgm, disponibilizado no enunciado do Exercício-Programa.

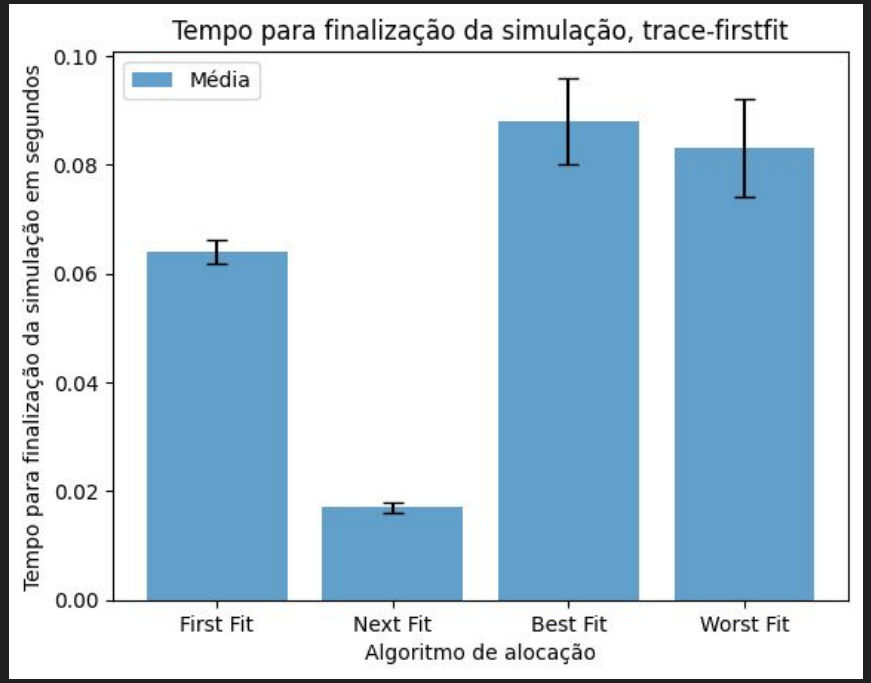
Máquina de teste

Os experimentos foram realizados em uma máquina com as seguintes especificações:

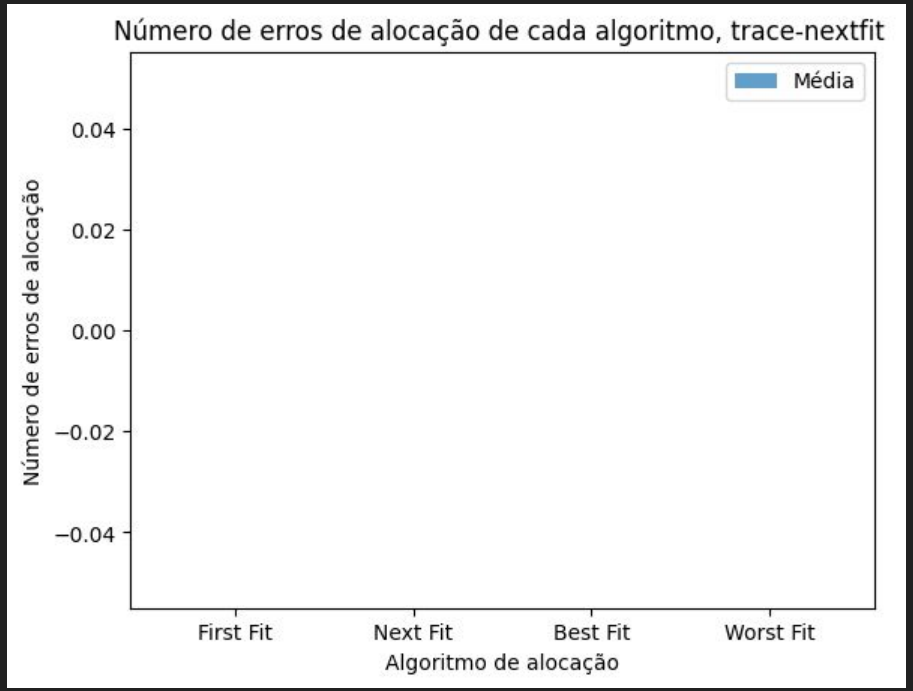
- 8 núcleos de processamento;
- processador Intel Core i7;
- Sistema Operacional Ubuntu na versão 22.04.

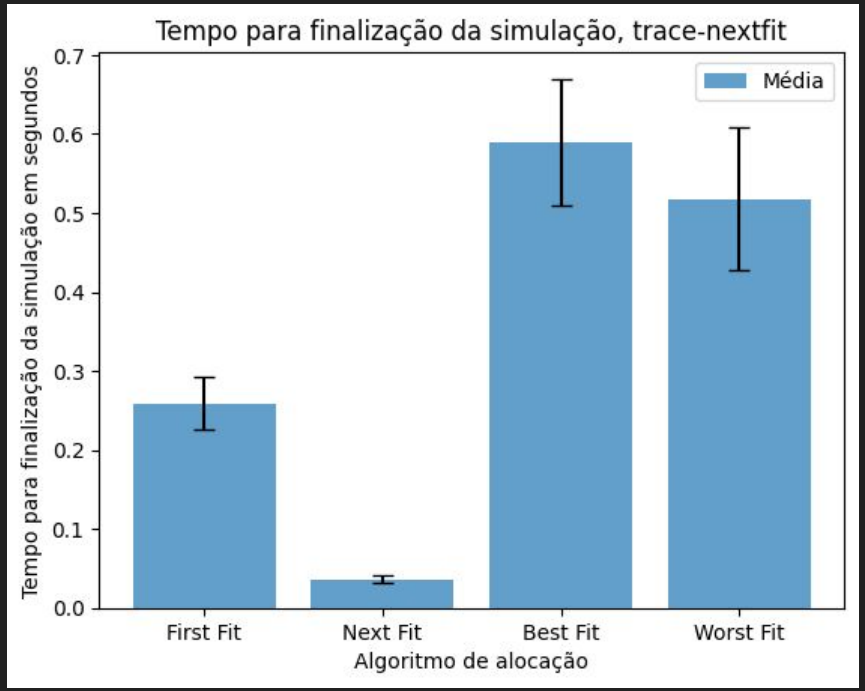
Experimentos para o arquivo trace-firstfit



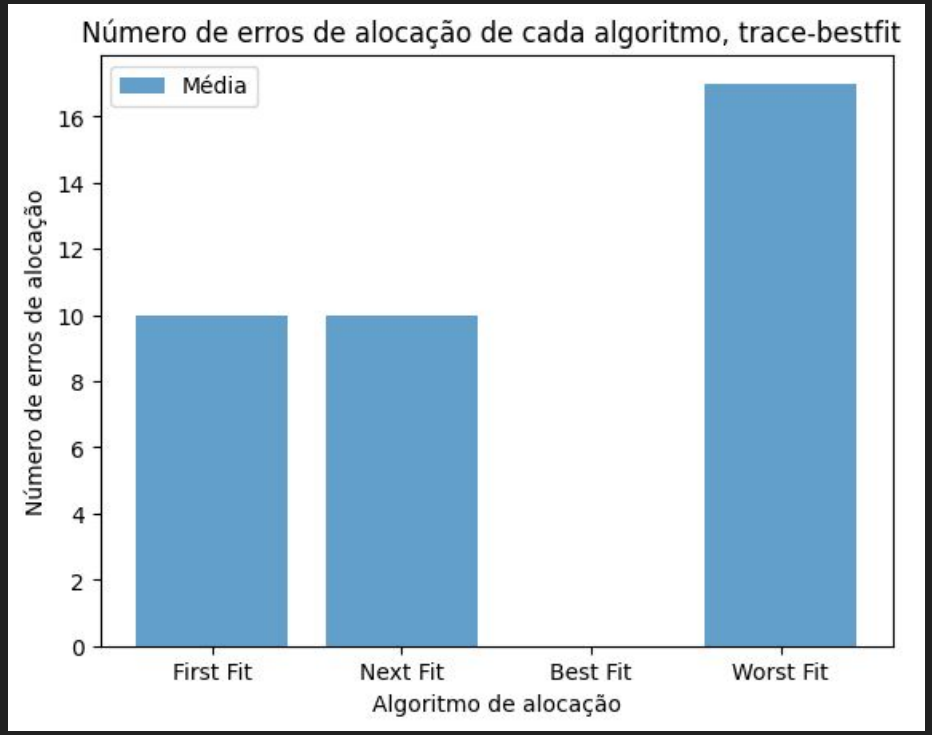


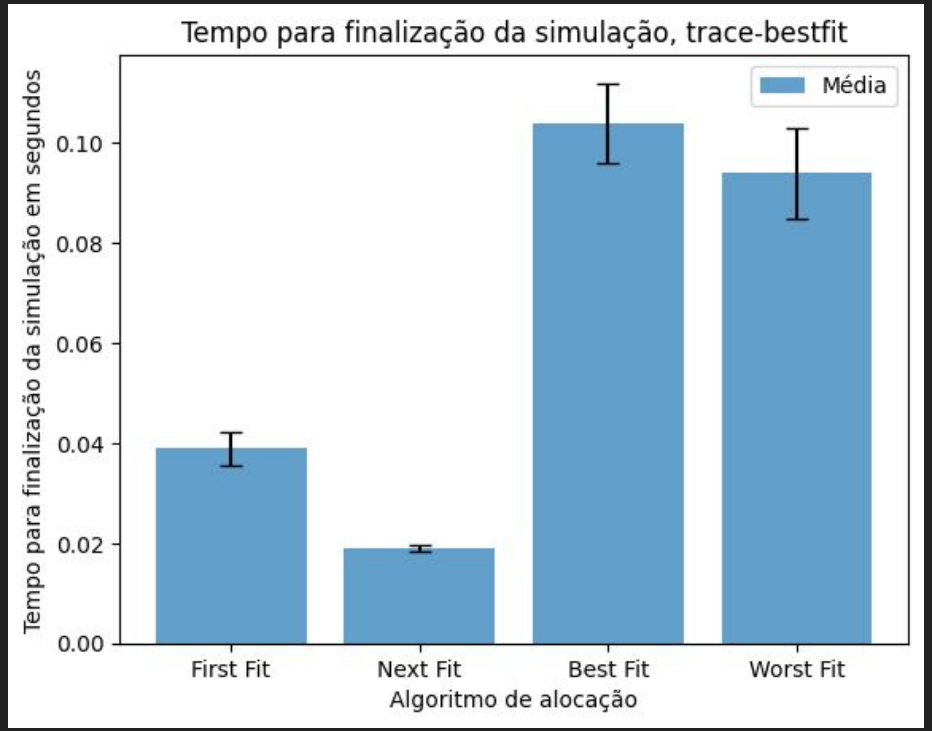
Experimentos para o arquivo trace-nextfit



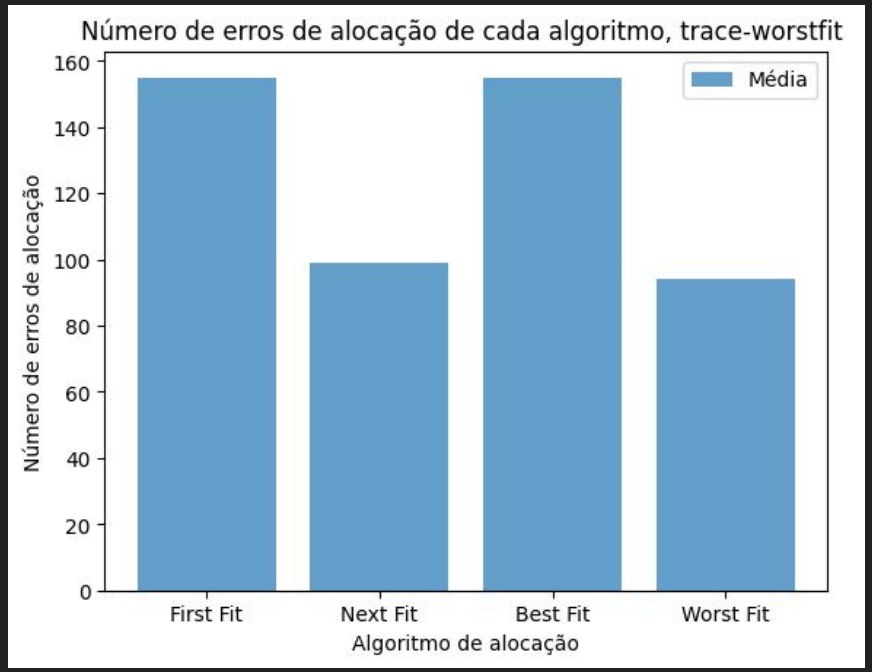


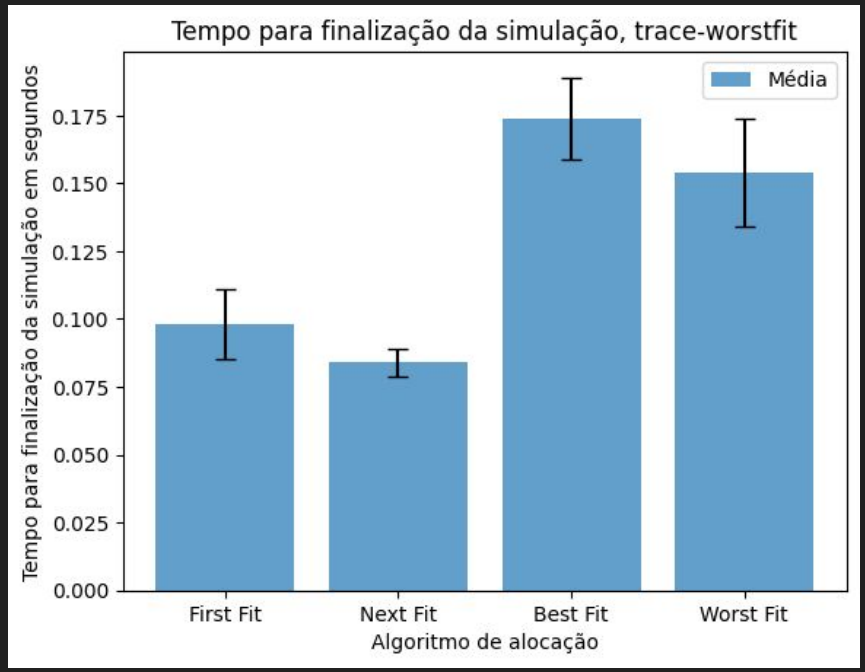
Experimentos para o arquivo trace-bestfit





Experimentos para o arquivo trace-worstfit





Análise dos experimentos em relação ao tempo

Primeiramente, é possível observar que, como esperado, o Best-Fit e o Worst-Fit são os que mais gastam tempo para fazer a simulação. Isso ocorre porque tais algoritmos necessitam que haja a varredura de toda a memória para achar o local que ocorrerá a alocação.

Além disso, é evidente que, como esperado, o Next-Fit é mais rápido do que o First-Fit. Isso ocorre porque a ideia de evitar buscar espaço onde provavelmente não há, no Next-Fit, funciona de forma efetiva, varrendo um espaço menor da memória do que o First-Fit, sendo assim, muito mais rápido.

Análise dos experimentos - erros de alocação

Primeiramente, é evidente que os algoritmos First-Fit e Next-Fit são bem mais simples, já que a partir do ponto de partida de busca, apenas buscam o primeiro espaço livre que cabe a requisição. Dessa forma, tais algoritmos tendem a ter muitos erros quando há muitas requisições seguidas e de tamanhos variados, pois fragmentam a memória com facilidade, levando a uma ineficiência no uso da memória, apesar de serem mais rápidos.

Já o Best-Fit, que evita quebrar espaços livres muito grandes, é eficiente quando há muitas requisições médias e pequenas e a memória está relativamente fragmentada, como no final do arquivo trace-bestfit. Entretanto, é evidente que em alguns casos, o Best-Fit pode levar a um mau uso da memória ao criar buracos pequenos que não servirão para nada, como no arquivo trace-worstfit.

Por fim, o Worst-Fit é eficiente quando se deseja retardar a fragmentação da memória, especialmente em cenários onde as requisições variam bastante de tamanho, como no arquivo trace-worstfit. Entretanto, é evidente que ele pode ser ineficiente quando as requisições são predominantemente pequenas, pois tende a dividir os maiores blocos em partes menores desnecessariamente, desperdiçando espaço e dificultando futuras alocações maiores.