MAC0422

SISTEMAS OPERACIONAIS

EP3

Thainara de Assis Goulart
13874413

Variáveis Auxiliares Utilizadas

First Fit

livres

Conta quantas UAs (unidades de alocação) livres (pixel == 255) foram encontradas consecutivamente até o momento.

Inicialmente vale 0. Quando **livres** == **m** (tamanho do bloco buscado), significa que existe um bloco de **m** UAs livres.

pos

Guarda o índice inicial do bloco livre que está sendo contado.

É atualizado quando **livres** passa de 0 para 1 (primeira UA livre após um ocupada) e permanece fixo até que **livres** == **m**, quando retornamos **pos**.

Variáveis Auxiliares Utilizadas

Next Fit

nfPos

É uma variável global que marca a posição onde a próxima busca de bloco deve começar.

Inicia em 0 e após uma alocação bem-sucedida de tamanho m em pos, faz

nfPos = (pos + m) % UATOTAL

para “continuar” dali na próxima iteração.

livres e pos

Funcionam como no First Fit.

Variáveis Auxiliares Utilizadas

Next Fit

bestTam

Armazena o menor tamanho de bloco encontrado até agora que seja >= **m**.

Ao fechar cada bloco (quando encontra uma UA ocupada ou chega ao fim), se

livres >= m && livres < bestTam

atualiza **bestTam = livres** e **bestPos = pos**.

bestPos

Guarda o índice inicial do bloco de tamanho **bestTam**, sendo retornado ao fim do loop, após percorrer a memória inteira procurando pelo espaço mais justo para alocação.

livres e pos

Funcionam como no First Fit, porém identificam e contam cada buraco livre à medida que percorremos toda a memória (mesmo após encontrar um candidato).

Variáveis Auxiliares Utilizadas

Worst Fit

worstTam

Armazena o maior tamanho de bloco encontrado que seja >= **m**.

Ao fechar cada bloco, se

livres >= m && livres > worstTam

atualiza **worstTam = livres** e **worstPos = pos**.

worstPos

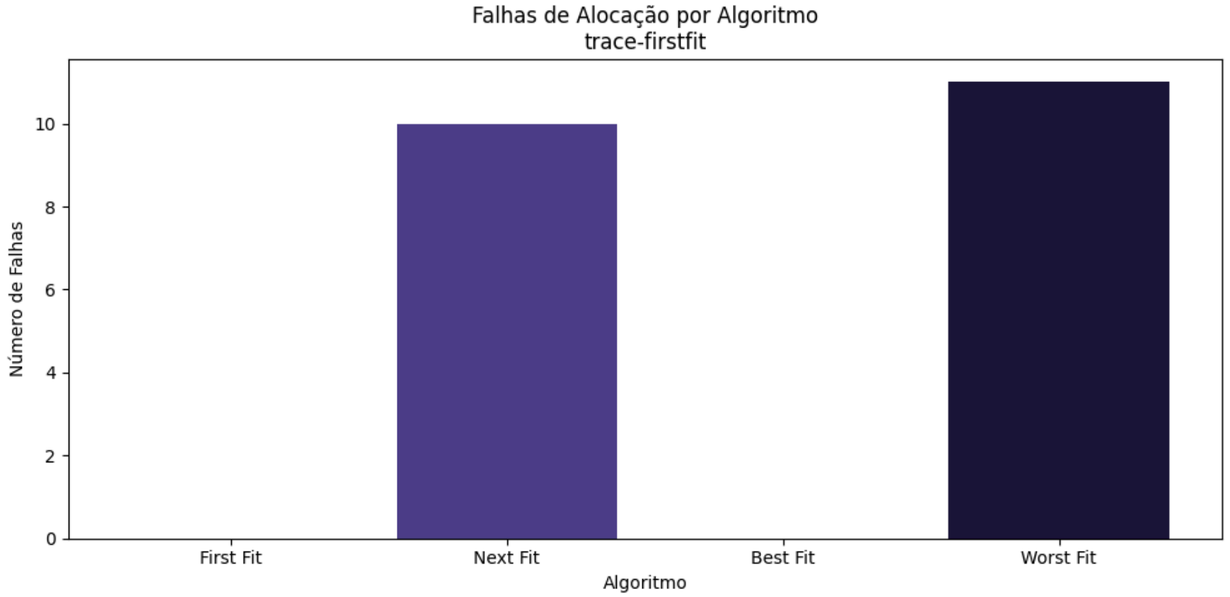
Guarda o índice inicial do bloco de tamanho **worstTam**, sendo retornado ao fim do loop, após percorrer a memória inteira procurando pelo maior espaço para alocação.

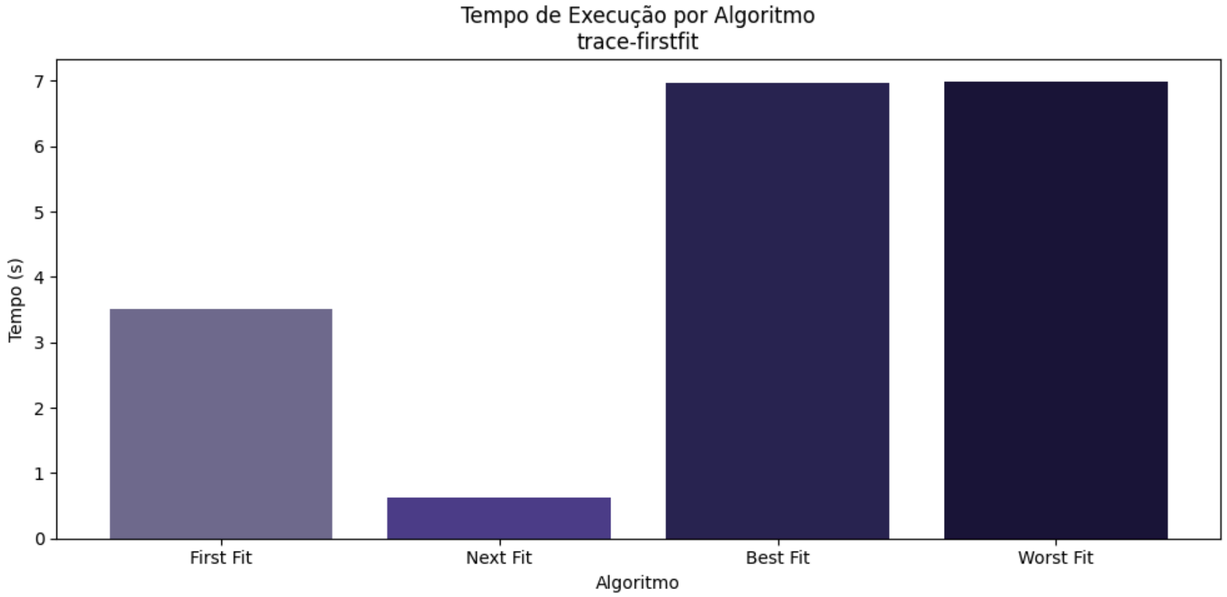
livres e pos

Idênticos ao Best Fit, para contabilizar cada bloco livre.

Resultados

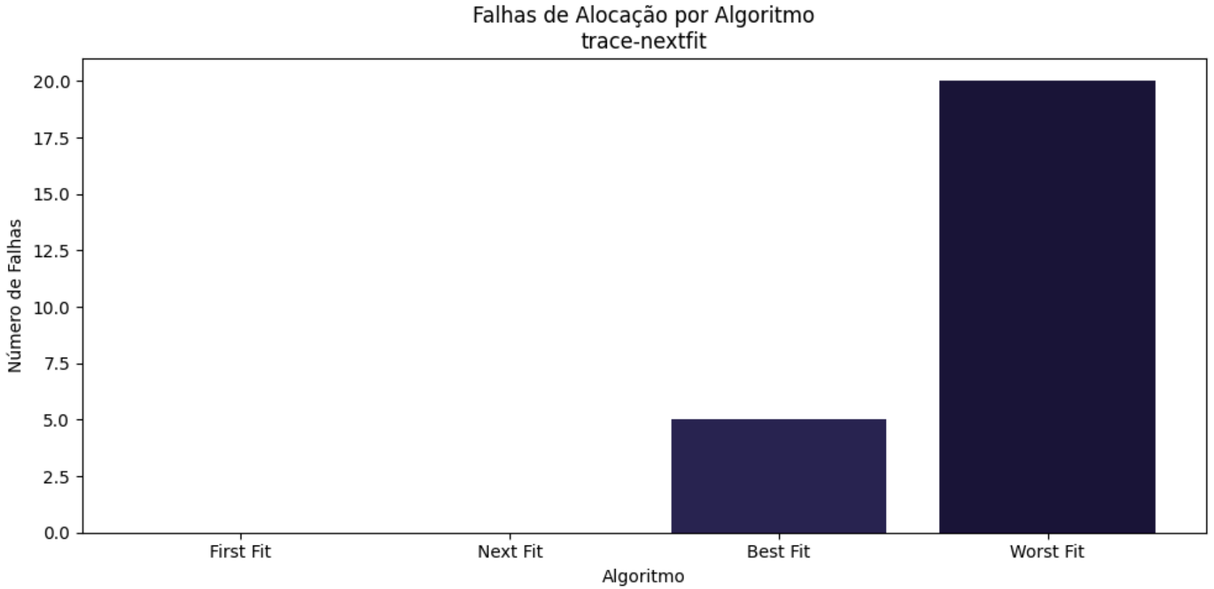
trace-firstfit

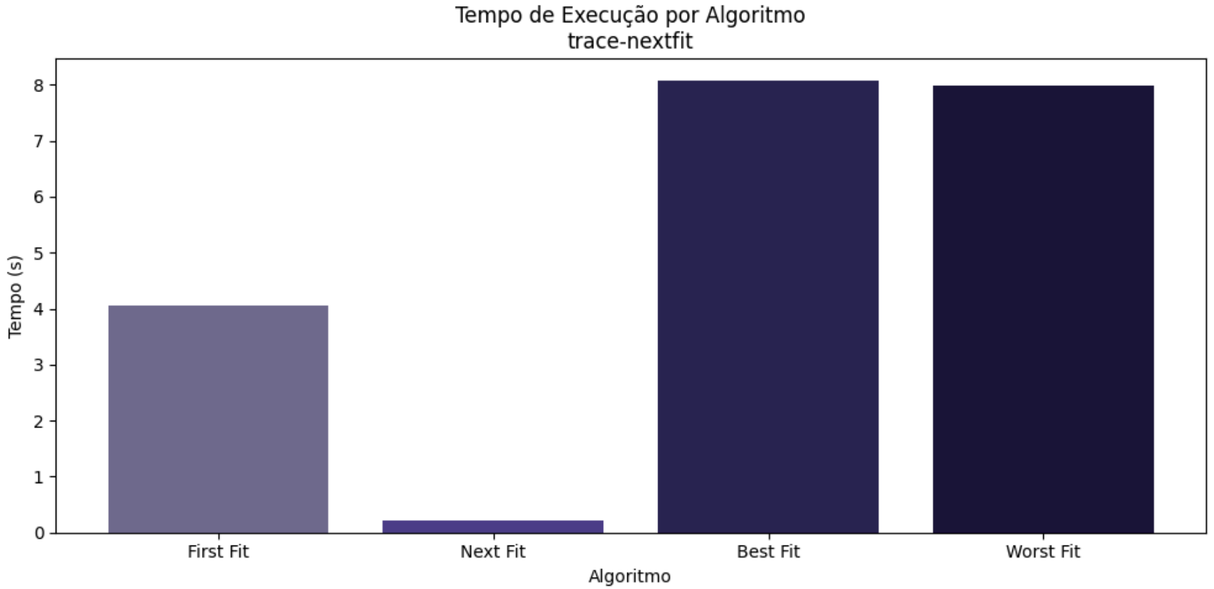




Resultados

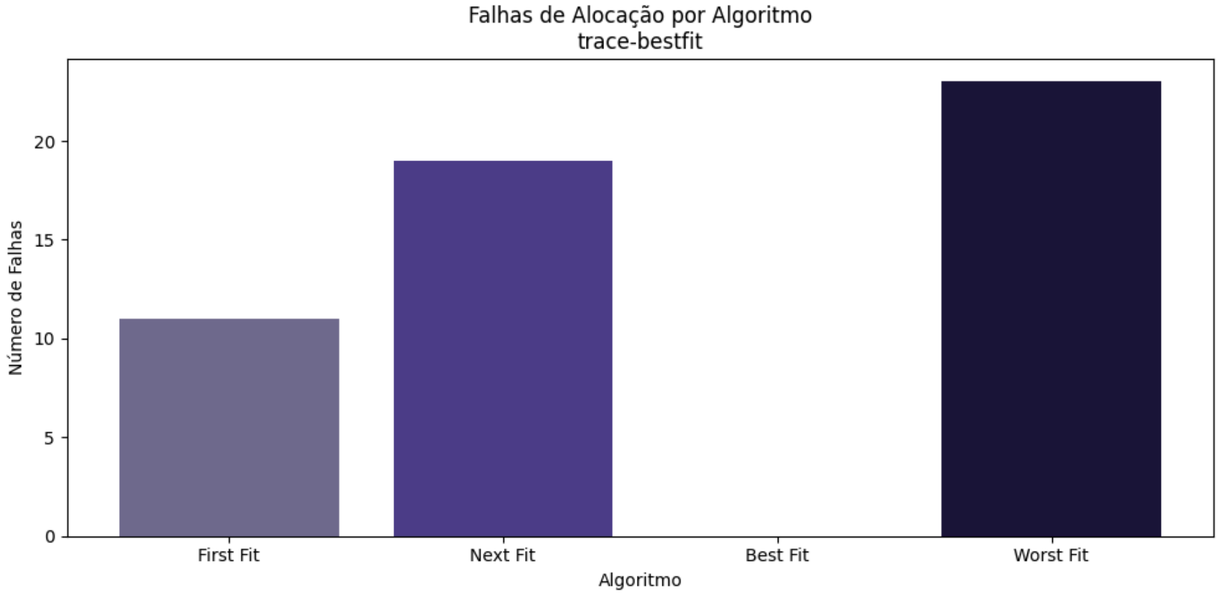
trace-nextfit

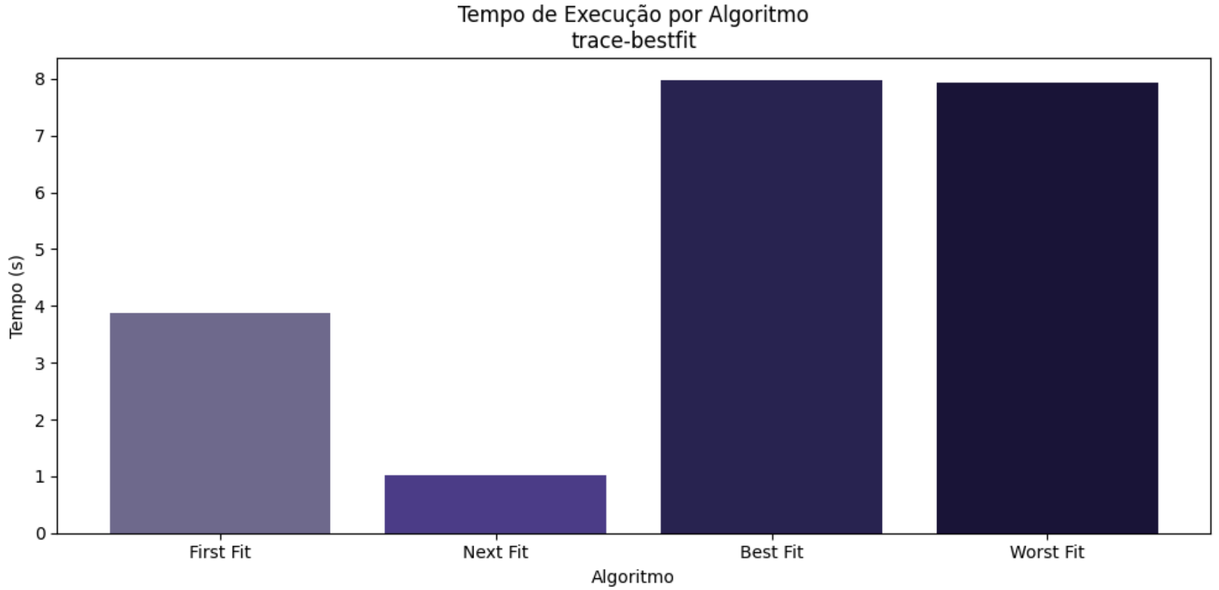




Resultados

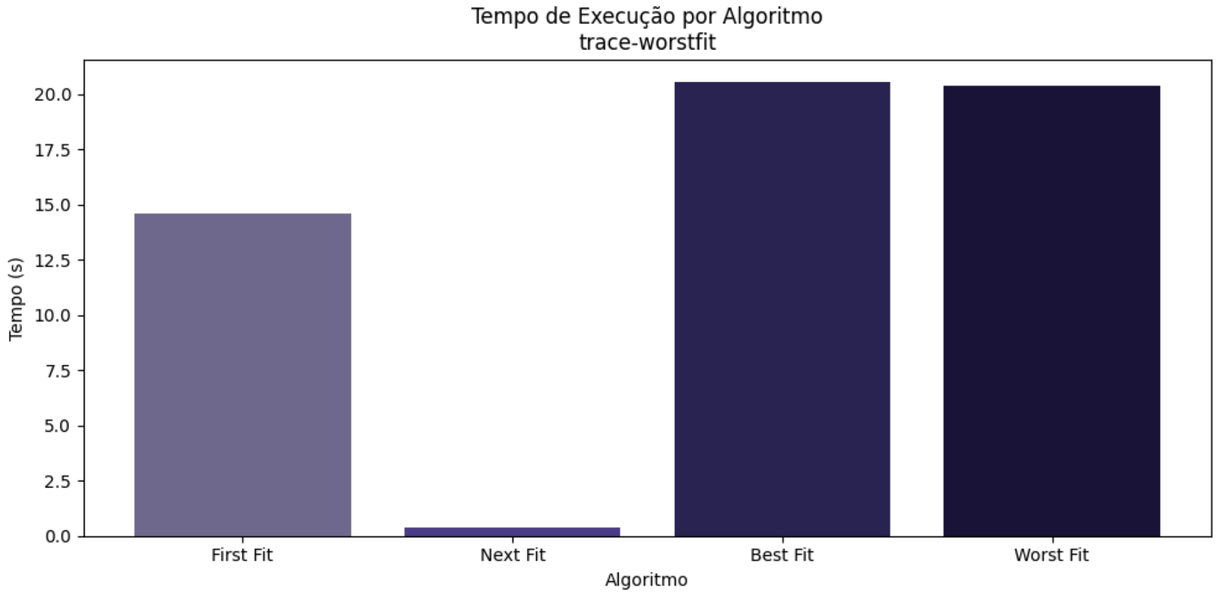
trace-bestfit





Resultados

trace-worstfit



Análise

trace-firstfit

First Fit se comporta como esperado: aproveita o primeiro espaço e não falha, além de ter um melhor tempo de execução em relação ao BF e WF, mostrando um bom desempenho do first fit.

Best Fit também se comporta como esperado, aproveitando o menor espaço adequado e não falha.

Next Fit falha em 10 requisições: continua de onde parou e “pula” alguns buracos iniciais que FF/BF usariam, porém ainda possui o melhor tempo.

Worst Fit falha em 11, sendo o pior, pois preserva sempre o maior buraco, deixando fragmentos que não servem aos pedidos do trace.

trace-nextfit

Next Fit faz exatamente o previsto: nenhuma falha, já que nosso trace está ocupando a memória na ordem exata de espaços livres, e melhor tempo, já que NF não revisita início.

First Fit também não falha, mas demora mais, pois sempre volta ao início, como esperado.

Best Fit falha 5 vezes: trace com muitos blocos grandes e pequenos (relativamente) alternados cria cenários em que o menor adequado não existe.

Worst Fit falha 20 vezes, sendo o pior, pois o trace é projetado com pedidos que cabem em buracos médios, mas WF preserva sempre os maiores.

Análise

trace-bestfit

Best Fit se comporta como esperado e não falha, pois escolhe sempre o espaço mais justo para cada pedido, uma vez que o trace foi montado com valores crescentes, usando espaços pequenos com alocações pequenas, tendo espaço ao final para alocações grandes.

First/Next Fit falham dezenas de vezes, pois acabam fragmentando mais. Entretanto, o Next Fit ainda é o algoritmo mais rápido.

Worst Fit é o pior, pois tenta usar sempre o maior espaço e ignora espaços menores que cabem nos pedidos iniciais, não sobrando espaços para alocações maiores no fim.

trace-worstfit

Este trace foi elaborado para que nenhum algoritmo falhe, inclusive o Worst Fit. Ao ordenar as requisições em ordem decrescente, o Worst Fit sempre consome primeiro os maiores espaços de memória, deixando apenas pequenos fragmentos ao final. Dessa forma, mesmo os pedidos menores conseguem ser atendidos nos espaços remanescentes, e o algoritmo atinge sua melhor performance nesse cenário.

Em relação ao tempo , o **Next Fit** continua o mais rápido; **Best/Worst Fit** os mais lentos; **First Fit** intermediário, como sempre e esperado.