ALGORITMO PARA GERENCIAMENTO DINÂMICO DE MEMÓRIA

Giuliana Leon, Guilherme Goulart, Thamires Sampaio e João Antunes.

PLANO DE TESTES

TABELA DE REQUISITOS

Requisito	Prioridade	Descrição
RF001	Essencial	O usuário do sistema deve ser capaz de configurar o tamanho da heap para funcionamento do sistema.
RF002	Essencial	O usuário do sistema deve ser capaz de configurar o intervalo de valores (mínimo e máximo) para geração randômica do tamanho das variáveis a cada requisição produzida pelo gerador.
RF003	Essencial	O algoritmo de alocação deve alocar a memória para cada variável na heap sempre de forma contígua.
RF004	Essencial	O algoritmo de alocação deve monitorar o índice de porcentagem de espaços ocupados na heap, iniciando a execução de um desalocador de memória para liberar espaço na heap **sempre** que a porcentagem estiver acima do percentual máximo informado pelo usuário.

RF005	Essencial	O algoritmo de desalocação deve selecionar, de forma randômica, variáveis a serem desalocadas (simulando um fluxo natural de um sistema real com desalocações).
RF006	Essencial	O algoritmo de desalocação deve executar de forma paralela com o alocador.
RF007	Essencial	O número de requisições de alocação a serem alocadas na memória deve ser informado pelo usuário do sistema.
RF008	Essencial	O usuário deve informar o limite máximo de ocupação de memória como um percentual, acima do qual o algoritmo de desalocação de memória deve ser executado.
RF009	Essencial	O usuário deve informar o percentual mínimo de espaço livre na memória.
RF010	Essencial	O desalocador deve liberar memória até que o percentual de memória livre fique igual ou abaixo da porcentagem mínima informada.
RF011	Essencial	O vetor deve ser implementado como uma fila circular.
RF012	Essencial	Cada requisição deve informar ao algoritmo de alocação de memória o tamanho da variável dinâmica a ser alocada, bem como seu respectivo identificador.

RF013	Essencial	Um gerador de requisições randômicas deve alimentar o vetor de requisições.
RF014	Essencial	O alocador de memória e o gerador de requisições randômicas devem compor duas threads do sistema (uma thread é o alocador de memória e outra thread é o gerador de requisições).
RF015	Essencial	implementar um algoritmo de compactação para eliminar a fragmentação externa que se forma à medida que as variáveis vão sendo desalocadas.

DETALHAMENTO DA ABORDAGEM DE TESTES

Tipo de teste:	Funcional
Subtipo de teste:	Requisitos
Requisitos que motivaram:	RF001
Objetivo do teste:	Testar se o tamanho da heap informado pelo usuário está sendo armazenado corretamente em sua respectiva variável.

Tipo de teste:	Funcional
Subtipo de teste:	Requisitos
Requisitos que motivaram:	RF001

Objetivo do teste:	Testar se a heap está realmente sendo gerada com o tamanho especificado.

Tipo de teste:	Funcional
Subtipo de teste:	Requisitos
Requisitos que motivaram:	RF002
Objetivo do teste:	Testar se o intervalo de valores (mínimo e máximo) para geração randômica informados pelo usuário estão sendo armazenados em suas respectivas variáveis.

Tipo de teste:	Funcional
Subtipo de teste:	Requisitos
Requisitos que motivaram:	RF002
Objetivo do teste:	Testar se o intervalo de valores (mínimo e máximo) informados pelo usuário estão sendo utilizados da forma correta, gerando variáveis que possuem tamanho de valores existentes entre o mínimo e o máximo informado.

Tipo de teste:	Funcional
Subtipo de teste:	Requisitos
Requisitos que motivaram:	RF003
Objetivo do teste:	Verificar se o algoritmo de alocação está realizando a alocação da heap a cada

	requisição removida do vetor de requisições.

Tipo de teste:	Funcional
Subtipo de teste:	Requisitos
Requisitos que motivaram:	RF003
Objetivo do teste:	Verificar se o algoritmo de alocação está alocando corretamente as informações da requisição (Tamanho e ID)

Tipo de teste:	Funcional
Subtipo de teste:	Requisitos
Requisitos que motivaram	RF003
Objetivo do teste:	Verificar se o algoritmo de alocação está realizando a alocação de forma contígua.

Tipo de teste:	Funcional
Subtipo de teste:	Requisitos
Requisitos que motivaram:	RF004
Objetivo do teste:	Testar se o algoritmo de alocação está inicializando a execução de um desalocador de memória quando a porcentagem de ocupação da heap atinge um valor igual ou maior que seu percentual máximo informado pelo usuário.

Tipo de teste:	Funcional
Subtipo de teste:	Requisitos
Requisitos que motivaram:	RF005
Objetivo do teste:	Testar se o desalocador está realizando sua função de liberar espaço na heap sempre que for executado pelo algoritmo de alocação.

Tipo de teste:	Funcional
Subtipo de teste:	Requisitos
Requisitos que motivaram:	RF006
Objetivo do teste:	Verificar se o algoritmo de desalocação e o alocador estão sendo executados de forma paralela.

Tipo de teste:	Funcional
Subtipo de teste:	Requisitos
Requisitos que motivaram:	RF007
Objetivo do teste:	Testar se o número de requisições de alocação informado pelo usuário está sendo armazenado em sua respectiva variável.

Tipo de teste:	Funcional
Subtipo de teste:	Requisitos

Requisitos que motivaram:	RF007
Objetivo do teste:	Testar se o número de requisições de alocação informado pelo usuário está sendo utilizado de forma correta, ou seja, se o vetor realmente possui este número específico de requisições armazenadas.

Tipo de teste:	Funcional
Subtipo de teste:	Requisitos
Requisitos que motivaram:	RF008
Objetivo do teste:	Verificar se o percentual máximo definido pelo usuário para o limite de ocupação da heap está sendo armazenado em sua respectiva variável.

Tipo de teste:	Funcional
Subtipo de teste:	Requisitos
Requisitos que motivaram:	RF008
Objetivo do teste:	Verificar se o percentual definido pelo usuário para o limite de ocupação da heap está sendo utilizado de forma correta, ou seja, se o alocador está ativando a desalocação quando o percentual atual de ocupação da heap atinge o valor máximo definido.

Tipo de teste:	Funcional
Subtipo de teste:	Requisitos

Requisitos que motivaram:	RF009
Objetivo do teste:	Verificar se o percentual definido pelo usuário para o valor mínimo de espaço ocupado na heap está sendo armazenado em sua respectiva variável.

Tipo de teste:	Funcional
Subtipo de teste:	Requisitos
Requisitos que motivaram:	RF009
Objetivo do teste:	Verificar se o percentual definido pelo usuário para o valor mínimo de espaço ocupado na heap está sendo utilizado corretamente na desalocação, ou seja, se o desalocador está realizando a desalocação dos espaços na heap até que atinja uma porcentagem de ocupação menor ou igual ao mínimo definido pelo usuário.

Tipo de teste:	Funcional
Subtipo de teste:	Requisitos
Requisitos que motivaram:	RF012
Objetivo do teste:	Verificar se cada requisição está informando ao algoritmo de alocação o tamanho de sua variável dinâmica a ser alocada, bem como seu respectivo identificador.

Tipo de teste:	Funcional
Subtipo de teste:	Requisitos
Requisitos que motivaram:	RF013
Objetivo do teste:	Verificar se a geração de cada requisição está sendo feita de forma randômica.

Tipo de teste:	Funcional
Subtipo de teste:	Requisitos
Requisitos que motivaram:	RF013
Objetivo do teste:	Verificar se cada requisição está realmente sendo armazenada no vetor de requisições.

Tipo de teste:	Funcional
Subtipo de teste:	Requisitos
Requisitos que motivaram:	RF015
Objetivo do teste:	Verificar se o algoritmo de compactação está sendo executado quando detecta-se a necessidade, ou seja, quando o somatório retorna um valor maior que o tamanho da requisição a ser alocada.

Tipo de teste:	Funcional
Subtipo de teste:	Requisitos
Requisitos que motivaram:	RF015

Objetivo do teste:	Verificar se o algoritmo de compactação está fazendo o que deveria, ou seja, realocando todos os espaços para que exista apenas um buraco contíguo na heap ao invés de diversos menores divididos pela heap.

CASOS DE TESTE

Caso de uso	ID	Passos	Resultado Esperado
UC001 - Testar se a heap está realmente sendo gerada com o tamanho especificado (RF001).	1	Na interface do programa, coloque o tamanho da heap em 10 e preencha os outros dados normalmente.	
	2	Clique em “Adicionar dados” e após isso clique em “imprimir heap”.	Tanto a impressão da heap paralela quanto da heap sequencial devem mostrar que suas páginas vão de 0 a 9.

UC002 - Testar se o intervalo de valores (mínimo e máximo) informados pelo usuário estão sendo utilizados da forma correta, gerando variáveis que possuem tamanho de valores existentes entre o	1	Na interface do programa, coloque o tamanho mínimo da requisição em 1 e o tamanho máximo em 5, preencha os outros dados normalmente.	
	2	Clique em “Adicionar dados” e após isso clique em “imprimir vetor de requisições”.	Os valores informados de cada requisição deverá ser um número existente de 1 a 5.

mínimo e o máximo informado (RF002).			

UC003 - Verificar se o algoritmo de alocação está realizando a alocação da heap a cada requisição removida do vetor de requisições (RF003).	1	Rode a versão de testes do programa, preencha as informações como bem entender, contanto que não utilize valores = 0.	
	2	Ao executar o programa sequencialmente e paralelamente, compare o print da heap, lista de ocupados e vetor de requisições a cada alocação realizada.	A cada ciclo em que uma requisição é removida do vetor de requisições, esta deverá aparecer em algum espaço da heap.

UC004 - Verificar se o algoritmo de alocação está alocando corretamente as informações da requisição (Tamanho e ID) (RF003).	1	Rode a versão de testes do programa, preencha as informações como bem entender, contanto que não utilize valores = 0.	
	2	Ao executar o programa sequencialmente e paralelamente, compare o print da heap, lista de ocupados e vetor de requisições a cada alocação realizada.	Quando uma requisição é removida do vetor de requisições, seu tamanho deverá ser armazenado na lista de ocupados, juntamente com sua ID e a posição inicial em que ela está armazenada atualmente na heap.

UC005 - Verificar se o algoritmo de alocação está realizando a alocação de forma contígua (RF003).	1	Rode a versão de testes do programa, coloque uma heap de tamanho 10, 5 requisições, porcentagem máxima de 90, mínima de 40 e o tamanho da variável dinâmica sendo algo entre 1 e 2.	
	2	Ao executar o programa sequencialmente e paralelamente, compare o print da heap, lista de ocupados e vetor de requisições a cada alocação realizada.	Quando uma requisição é removida do vetor de requisições, a mesma deverá ser alocada de forma contígua na memória, ou seja, enquanto não houver buracos na heap o software deverá alocar de forma contígua cada uma das variáveis, respeitando seus respectivos tamanhos.

UC006 - Testar se o algoritmo de alocação está inicializando a execução de um desalocador de memória quando a porcentagem de ocupação da heap atinge um valor igual ou maior que seu percentual máximo	1	Rode a versão de testes do programa, coloque uma heap de tamanho 10, 5 requisições, porcentagem máxima de 50, mínima de 30 e o tamanho da variável dinâmica sendo 1 (tanto o máximo quanto o minimo).	
	2	Ao executar o programa sequencialmente e paralelamente, compare o print da heap, lista de ocupados e vetor de requisições a cada alocação realizada.	Quando o print mostrar que 5 ou mais espaços estão ocupados na heap, no próximo ciclo um print deverá aparecer na tela com a

informado pelo usuário (RF004 e RF008).			mensagem “O desalocador foi chamado, porcentagem de ocupação da heap ultrapassou os 50%”

UC007 - Testar se o desalocador está realmente desalocando a heap quando é chamado, além de verificar se o percentual definido pelo usuário para o valor mínimo de espaço ocupado na heap está sendo utilizado corretamente na desalocação (RF005 e RF009).	1	Rode a versão de testes do programa, coloque uma heap de tamanho 10, 5 requisições, porcentagem máxima de 50, mínima de 30 e o tamanho da variável dinâmica sendo 1 (tanto o máximo quanto o minimo).	
	2	Ao executar o programa sequencialmente e paralelamente, compare o print da heap, lista de ocupados e vetor de requisições a cada alocação realizada.	Quando o print mostrar que 5 ou mais espaços estão ocupados na heap, anote o ID destes e compare com o próximo print do vetor de ocupados, ao menos um ID do ciclo anterior deverá ter desaparecido da lista (desalocado) e seus respectivos blocos estarão vazios. A desalocação apenas irá ser finalizada quando a heap possuir apenas 3 ou menos blocos ocupados (30%)

UC007 - Testar se o número de requisições de alocação informado	1	Na interface do programa, coloque o número de requisições em 10, preencha os outros dados	

pelo usuário está sendo utilizado de forma correta, ou seja, se o vetor realmente possui este número específico de requisições armazenadas (RF007).		normalmente.	
	2	Clique em “Adicionar dados” e após isso clique em “imprimir vetor de requisições”.	O vetor de requisições impresso deverá possuir exatamente 10 espaços, com todos preenchidos com requisições de tamanhos distintos.

UC008 - Verificar se cada requisição está informando ao algoritmo de alocação o tamanho de sua variável dinâmica a ser alocada, bem como seu respectivo identificador (RF012).	1	Na interface do programa, coloque o número de requisições em 10, tamanho da requisição entre 1 e 5, preencha os outros dados normalmente.	
	2	Clique em “Adicionar dados”, depois disso rode o programa em modo paralelo e modo sequencial, comparando o print da heap, vetor de ocupados e vetor de requisições a cada ciclo.	Veja se o tamanho da variável da primeira requisição removida do vetor de requisições está na lista de ocupados após a primeira alocação, juntamente com seu identificador.

UC009 - Verificar se o somatório está realizando o cálculo corretamente.	1	Na interface do programa, coloque o número de requisições em 10, tamanho da requisição entre 1 e 5, preencha os outros dados normalmente.	
	2	Clique em “Adicionar dados”, depois disso rode o programa em modo paralelo e modo sequencial.	Analisando o print da heap, conte quantos blocos vazios ela

			possui (blocos com valor = 0), após isso, compare este valor com o valor informado pelo somatório, ambos precisam ser iguais.

UC010 - Verificar se a compactação está sendo realizada de forma correta (RF015)	1	Rode a versão de testes do programa, coloque uma heap de tamanho 10, 5 requisições, porcentagem máxima de 50, mínima de 30 e o tamanho da variável dinâmica sendo algo entre 1 e 5.	
	2	Clique em “Adicionar dados” e rode o programa tanto de forma sequencial quanto paralela	Uma mensagem falando “compactação realizada!” deverá aparecer na tela caso uma situação que necessite de compactação ocorra, compare o print da heap realizado previamente à compactação e o print atual, verifique se todos os espaços ocupados agora estão contíguos e os buracos (fragmentação externa) agora se tornaram apenas um grande buraco contíguo.

UC011 - Verificar se a interface mostra um aviso de erro caso o usuário não preencha algum campo da interface.	1	Na interface do programa, preencha todos os dados exceto um (você precisará fazer isso 6 vezes, uma para cada campo).	
	2	Clique em “Adicionar dados”.	Uma mensagem falando “Há campos de textos vazios, insira dados” deverá aparecer na tela.

UC012 - Verificar se a interface mostra um aviso de erro caso o usuário coloque um tamanho de variável máximo maior que o tamanho da heap informado	1	Na interface do programa, preencha todos os dados normalmente, porém coloque o tamanho máximo da variável da requisição maior que o tamanho da heap.	
	2	Clique em “Adicionar dados”.	Uma mensagem falando “Tamanho da requisição maior que o permitido” deverá aparecer na tela.

UC013 - Verificar se a interface mostra um aviso de erro caso o usuário coloque um tamanho de variável mínimo maior que o tamanho de variável máximo informado	1	Na interface do programa, preencha todos os dados normalmente, porém coloque o tamanho mínimo da variável da requisição maior que o tamanho máximo.	
	2	Clique em “Adicionar dados”.	Uma mensagem falando “Tamanho mínimo da requisição está maior que o permitido (maior que o

			tamanho máximo definido)” deverá aparecer na tela.

CONSIDERAÇÕES

Todos os testes descritos acima foram validados com sucesso.