MC202 - Estruturas de Dados

Lab 05

Data da Primeira Chance: 11 de Setembro de 2023
Link da atividade: https://classroom.github.com/a/aOO6lMhE
Peso: 3

A Lei de Moore deixou de ser uma verdade há alguns anos. Durante um tempo, a indústria de processadores se apegou ao Escalonamento de Dennard para aumentar o desempenho de cada core, aumentando a velocidade de clock. Em paralelo, incentivados pela regra de Amdahl's, a quantidade de cores em um chip cresceu consideravelmente na última década (o primeiro processador comercial com mais de um core foi lançado em 2005). Recentemente, o futuro desta área aparenta ser em Arquiteturas Específicas de Domínio para extrair ainda mais desempenho. O exemplo mais notável são as placas aceleradoras de aprendizado de máquina lançadas pela Google. *Mas isso é spoiler de MC404.*

Neste contexto, o Homem-Morcego, Cavaleiro das Trevas, Cruzado Encapuzado, Maior Detetive do Mundo, primeiro de seu nome, vulgarmente conhecido como Batman, precisa de sua ajuda para construir um novo bat-computador para combater o crime em Gothan City. Você, como grande fã do trabalho do vigilante noturno, se propôs a implementar as funções de gerenciamento da bat-memória.

Você implementará algumas operações a serem feitas sob a bat-memória e utilizará um vetor **dinâmico** de inteiros para guardar essas informações. Tome cuidado com as restrições do nosso herói, ele pode ser meio exigente às vezes.

Entrada

Em todos os testes, a bat-memória deve começar com 8 inteiros e crescer baseado na necessidade. A primeira linha contém um inteiro, a quantidade de instruções a serem analisadas. Cada uma das linhas seguintes correspondem a uma instrução de memória:

Bat-alloc

A operação de bat-alloc guarda uma lista de inteiros na bat-memória virtual do seu bat-computador e imprime a posição de início dele. Da forma como Batman pediu para você implementar, ele deseja que todo vetor de inteiros seja armazenado de forma contínua, ou seja, um vetor de 4 inteiros não pode ter seus inteiros colocados espalhados pela bat-memória.

Além disso, Batman pediu para você sempre colocar o vetor no primeiro espaço de memória disponível em que ele cabe, seguindo a ordem dos índices. Assim, se um vetor couber começando na posição 5 ou na posição 40, você sempre irá escolher a posição 5.

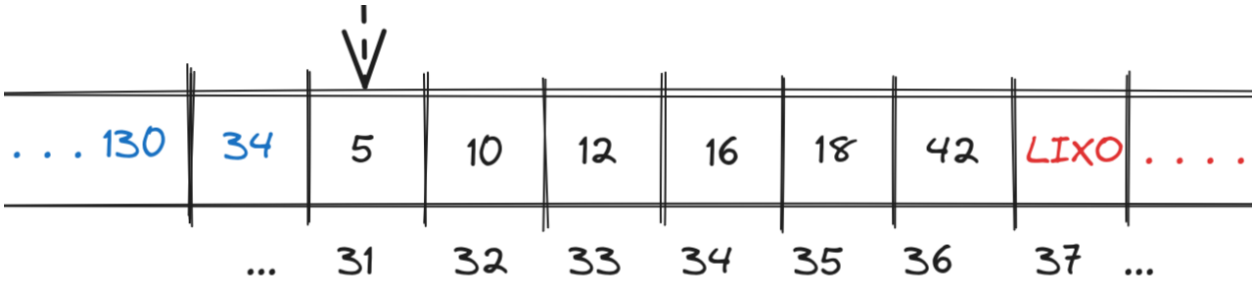
Operações como esta serão lidas na forma:

`bat-alloc N LISTA`

Guarda a lista de N inteiros na memória virtual. Uma instrução pode ser parecer como a seguinte:

`bat-alloc 5 10 12 16 18 42`

Assim, Batman espera que você imprima o local da memória onde o vetor foi armazenado. Há um detalhe: **a primeira posição do espaço armazenado deve conter o tamanho do vetor**. Assim, na bat-memória, teríamos:



Nesse caso, deve ser impresso 31.

Caso não haja nenhum espaço contínuo para colocar o vetor, você deve duplicar a capacidade do vetor de bat-memória virtual. Você pode fazer isso criando um novo vetor e copiando (nos mesmos locais de memória) o vetor antigo. Assim, o vetor acima, depois de uma expansão, sempre permanecerá na posição 31, até que seja bat-liberado. Sempre será possível alocar após a expansão.

Não é necessário tratar, de forma alguma, a fragmentação de memória. Atentem-se apenas às regras esperadas pelo Batman, as necessidades deles são realmente específicas.

Exemplo de saída:

```
31
```

Bat-free

A operação de bat-free libera a memória correspondente a um vetor. Ela aparecerá no formato

`bat-free ENDEREÇO`

Assim, seguindo o exemplo anterior, se o Batman te pedir para dar bat-free no endereço 32, você poderá usar as posições 32 à 36 (inclusive) livremente para alocar outras coisas depois. Perceba que isso irá deixar lacunas vazias na sua memória, mas no bat-computador isso não é um problema.

Caso, após uma liberação, não haja bat-memória alocada depois do **primeiro quarto** do vetor, ele deve ser **reduzido pela metade**. Assim, suponha que estamos com uma memória de 128 inteiros. Se não houver nenhuma memória alocada entre as posições 32 a 127 (incluso), o vetor deve ser reduzido a 64 inteiros.

Atenção: a memória nunca deve ser reduzida a menos que 8 inteiros. Além disso, Batman, como grande programador que é, nunca erra os endereços passados para você. Assim, todo bat-free possui um alvo válido (alocado e ainda não liberado).

Nessa instrução, não é necessário imprimir nada.

Bat-print

A operação de bat-print imprime a lista de inteiros indicada pelo seu início. Ela aparecerá no formato:

```
bat-print ENDEREÇO
```

Os inteiros são impressos separados por espaço. Apenas os inteiros correspondentes ao vetor alocado naquela posição. Perceba que a posição apontada pelo endereço é, na verdade, o tamanho do vetor.

Exemplo de saída:

```
10 12 16 18 42
```

Bat-uso

A operação de bat-uso imprime uma linha com dois inteiros: a quantidade de inteiros da memória que estão sendo usados (seja pela alocação ou por representarem o tamanho do vetor) e o tamanho total da bat-memória.

Exemplo de saída:

```
321 de 512
```

Exemplos

Exemplo 1:

Entrada

```
6
bat-alloc 5 1 2 3 4 5
bat-print 0
bat-alloc 3 9 8 7
bat-print 6
bat-free 0
bat-uso
```

Saída

```
0
1 2 3 4 5
6
9 8 7
4 de 16
```

Exemplo 2:

Entrada

```
8
bat-alloc 7 52 57 63 25 33 40 37
bat-alloc 6 69 35 62 49 44 44
bat-free 0
bat-alloc 4 36 45 59 66
bat-uso
bat-alloc 3 24 34 51
bat-print 0
bat-uso
```

Saída

```
0
8
0
12 de 16
15
36 45 59 66
16 de 32
```

Dicas

Em todos os labs desta disciplina, é importante que você separe as tarefas que você deve implementar. Neste lab, é ainda mais imprescindível.

Por fim, é recomendável que você crie TAD's e atribua funções e responsabilidades a eles. Uma sugestão é ter um vetor de registros de início e tamanho de espaços alocados (além do vetor de inteiros correspondentes à memória).

Regras e Avaliação

Seu código será avaliado não apenas pelos testes do GitHub, mas também pela qualidade. Dentre os critérios subjetivos de qualidade de código analisaremos neste laboratório:

- A escolha de bons nomes de funções e variáveis;
- A ausência de trechos de código repetidos desnecessariamente;
- O uso apropriado de funções;
- A ausência de vazamentos de memória;
- A eficiência dos algoritmos propostos;
- A correta utilização das Estruturas de Dados;
- Documentação adequada;
- **Não é permitido o uso da função realloc.**
- **Você deve representar a memória virtual pelo uso de vetores dinâmicos de inteiros.** Você pode usar mais estruturas e TADs, mas soluções que não armazenam os inteiros num vetor que precisa ter seu tamanho alterado com o passar do tempo terão a nota zerada.

Note, porém, que essa não é uma lista exaustiva, pois outros critérios podem ser analisados dependendo do código apresentado visando mostrar ao aluno como o código poderia ser melhor.

Submissão

Você deverá submeter no repositório criado no aceite da tarefa. Você pode enviar arquivos adicionais caso deseje para serem incluídos por `batmemory.c`. Não se esqueça de dar `git push` !

Lembre-se que sua atividade será corrigida automaticamente na aba "Actions" do repositório. Confirme a correção e o resultado, já que o que vale é o que está lá e não na sua máquina.

Atenção: O repositório da sua atividade conterá alguns arquivos iniciais. Fica **estritamente proibido** ao aluno alterar os arquivos já existentes, tais como o testador existente ou demais arquivos de configuração do laboratório.

Após a correção da primeira entrega, será aberta uma segunda chance, com prazo de entrega apropriado.