Pilhas e Filas

Prof. Salles Magalhaes

Pilhas e filas

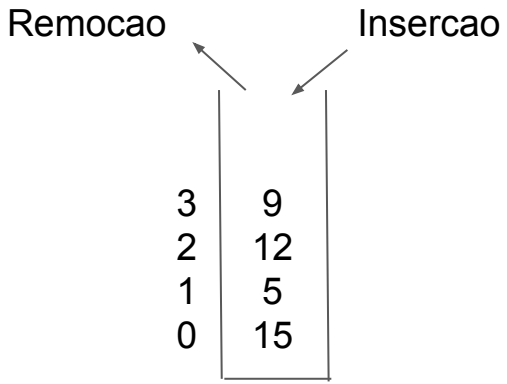
Pilhas e filas sao estruturas de dados similares a listas.

- A diferenca principal e’ que essas estruturas apresentam um conjunto bem limitado de operacoes

Pilhas

Pilhas sao estruturas de dados lineares que oferecem acesso do tipo LIFO (“last in, first out”).

Exemplo da vida real: uma pilha de papel



Pilhas

Operacoes suportadas:

- Construtor: constrói uma pilha vazia
- push(el): insere o elemento “el” no topo da pilha
- top(): retorna o elemento que está no topo da pilha (sem remove-lo!)
- pop(): remove o elemento que está no topo da pilha (sem retorna-lo!)
- empty(): retorna true se a pilha estiver vazia, falso caso contrario
- size(): retorna o tamanho da pilha (quantos elementos ha na pilha)

Considerando essas operacoes, qual estrutura de dados seria adequada para implementar uma pilha?

Pilhas

Precisamos basicamente de uma estrutura de dados linear que permita a insercao e remocao eficiente no final.

Listas encadeadas e por contiguidade: ok!

Listas por contiguidade: melhor localidade de referencia (cache)

Exercicio: como poderiamos implementar uma pilha usando uma lista por contiguidade?

Pilhas

Como implementar uma pilha utilizando uma lista por contiguidade? (ex: MyVec)

Veja o exemplo em MyStack.h (https://drive.google.com/open?id=1_olKoMobDzXtSNpA1Zv_MH2sJH_9GWC7) e TestaMyStack.cpp (https://drive.google.com/open?id=1_AlnQqlEt7UFSEYO_ucBT9uMUJsf4w5P)

Pilhas - exemplo de aplicacao

Exercicio (solucao: https://drive.google.com/open?id=1dZDu5cnsxz0QWJnKHroBubBpIu6MBIT0): dada uma expressao (representada por uma string) contendo, entre outros caracteres, os delimitadores: [,],(,),{,} , verifique se os delimitadores estao consistentes.

Exemplos de entrada consistente:	Exemplos de entrada inconsistente:
8*(a+b+[c+2]) (9+{[][][]() (()) }) (5=1*5) 1+2=100000	))(((9+2+(1+1)

Filas

Fila e’ uma estrutura de dados similar a pilhas. Porem, ela oferece acesso do tipo FIFO (“first in, first out”).

Basicamente uma fila e’ uma lista na qual as insercoes so podem ser feitas no final e as remocoes so podem ser realizadas no inicio.

Essa estrutura não prove operacoes que acessam elementos no meio da lista.

Filas

Operacoes suportadas:

- Construtor: constrói uma fila vazia
- push(el): insere o elemento “el” no final da fila
- front(): retorna o elemento que está no inicio da fila (sem remove-lo!)
- pop(): remove o elemento que está no inicio da fila (sem retorna-lo!)
- empty(): retorna true se a fila estiver vazia, falso caso contrario
- size(): retorna o tamanho da fila

Filas

```
MyQueue<int> f;
for(int i=0;i<5;i++) f.push(i);
while(!f.empty()) {
   cout << f.front() << endl;
   f.pop();
}
```

O que sera impresso?

Filas

Considere as operacoes principais:

- push(el): insere o elemento “el” no final da fila
- front(): retorna o elemento que está no inicio da fila (sem remove-lo!)
- pop(): remove o elemento que está no inicio da fila (sem retorna-lo!)

Exercicio: como uma fila poderia ser implementada de forma eficiente?

- Usando uma lista encadeada?
- Duplamente encadeada?
- Por contiguidade?

Filas

- Uma fila pode ser trivialmente implementada utilizando uma lista encadeada (remocao no inicio e insercao no final sao operacoes eficientes, O(1)).
- O uso de uma lista duplamente encadeada não ajudaria em nada (e aumentaria o uso de memoria).
- Porem, listas por contiguidade sao mais eficientes em termos de localidade dos dados (o que usa melhor a cache dos processadores), e possuem um overhead de memoria um pouco menor dependendo do tipo armazenado.
    - Qual seria a desvantagem de utilizar uma lista por contiguidade?

Desafio:

- Listas por contiguidade não suportam remocao eficiente no inicio.
    - Ideia: e se armazenarmos os dados em ordem reversa? (removendo os elementos do final)

Desafio:

- Listas por contiguidade não suportam remocao eficiente no inicio.
  - Ideia: e se armazenarmos os dados em ordem reversa? (removendo os elementos do final)
    - Nesse caso a insercao teria que ser feita no inicio! (tambem ineficiente)
- Solucao: lista por contiguidade circular
  - Estrategia para permitir insercao (no final) e remocao (no inicio) eficientes em uma lista por contiguidade.
  - Utiliza uma lista por contiguidade internamente.

Exemplo - lista circular

Veja uma implementacao de fila com lista circular em MyQueue.h:

Aplicacoes de fila

Exemplo simples: dada uma palavra representada por uma string (sem caracteres especiais), verifique se ela e' uma palindrome (leia um caractere por vez da entrada e armazene dados apenas em estruturas de dados como listas/pilhas)

Veja o exemplo: verificaPalindrome.cpp

Outros exemplos: botoes de “avancar” e “voltar” em navegadores de internet.

Exercicio: inverta a ordem dos elementos de uma pilha usando duas pilhas adicionais (sem usar operacoes de atribuicao de pilhas)