Algoritmos e Estruturas de Dados: Filas de Prioridade

Rômulo Silva de Oliveira
Departamento de Automação e Sistemas – DAS – UFSC

romulo@das.ufsc.br
http://www.das.ufsc.br/~romulo
Maio/2011

Referências

- Mastering Algorithms with C
  Kyle Loudon
  O´Reilly, 1999
- Livros de algoritmos e estruturas de dados em geral

Fila de Prioridade – Introdução 1/2

- Filas de prioridade são usadas para definir ordem de prioridade
- Elementos são organizados de tal forma que
  o elemento de prioridade mais alta possa ser acessado facilmente
- Por exemplo:
  - Fila de tarefas no sistema operacional
  - Fila de eventos a serem processador
  - Número enorme de aplicações

Fila de Prioridade – Introdução 2/2

- Muitas formas de implementar
- Mais intuitivo é manter os dados ordenados
- O elemento no início é aquele com prioridade mais alta
- Entretanto, inserir e remover elementos requer reordenar os dados

- Solução melhor:
  mantem os dados parcialmente ordenados usando um heap
- O nodo no topo do heap será sempre aquele com a prioridade mais alta
- Atualização do heap é mais rápida

Fila de Prioridade – Interface 1/6

- ***pqueue_init***
  - void pqueue_init(PQueue **pqueue*,
    int (**compare*)(const void **key1*, const void **key2*),
    void (**destroy*)(void **data*));
- **Retorno**
  - Nenhum.
- **Descrição**
  - Iniciliza a fila de prioridade especificada por *pqueue*. Esta operação deve ser chamada para uma fila de prioridade antes que ela possa ser usada com qualquer outra operação. O argumento *compare* é uma função usada por várias operações para manter a propriedade básica da fila. Esta função deve retornar 1 se *key1*>*key2*, 0 se *key1*=*key2*, e -1 se *key1*<*key2* para uma fila de prioridade na qual chaves maiores tem uma prioridade mais alta. Para uma fila de prioridade na qual chaves menores tem prioridade mais alta, *compare* deverá inverter os casos que retornam 1 e -1. O argumento *destroy* prove uma forma para liberar dados dinamicamente alocados quando *pqueue_destroy* for chamada. Por exemplo, se a fila de prioridade contem dados dinamicamente alocados usando *malloc*, *destroy* deveria ser *free* para liberar os dados quando a fila de prioridade for destruida. Para dados estruturados contendo diversos membros dinamicamente alocados, *destroy* deve ser uma função definida por usuário que chama *free* para cada membro alocado dinamicamente, assim como para a própria estrutura. Para uma fila de prioridade contendo dados que não devem ser liberados, *destroy* deve ser NULL.

Fila de Prioridade – Interface 2/6

- ***pqueue_destroy***
  - void pqueue_destroy(PQueue **pqueue*);
- **Retorno**
  - Nenhum.
- **Descrição**
  - Destrói a fila de prioridade especificada por *pqueue*. Nenhuma outra operação é permitida após a chamada de *pqueue_destroy* a não ser que *pqueue_init* seja chamada novamente. A operação *pqueue_destroy* extrai todos os elementos da fila de prioridade e chama a função passada como *destroy* para *pqueue_init* uma vez para cada elemento quando o mesmo é removido, desde que *destroy* não seja NULL.

Fila de Prioridade – Interface 3/6

- ***pqueue_insert***
  - int pqueue_insert(PQueue *_pqueue_, const void *_data_);
- **Retorno**
  - 0 se a inserção do elemento tiver sucesso, ou -1 caso contrário.
- **Descrição**
  - Insere um elemento na fila de prioridade especificada por *pqueue*. O novo elemento contem um pointer para *data*, logo a memória referenciada por *data* deverá permanecer válida pelo tempo que o elemento permanecer na fila de prioridade. É responsabilidade do chamador gerenciar a memória associada com *data*.

Fila de Prioridade – Interface 4/6

- ***pqueue_extract***
  - int pqueue_extract(PQueue *_pqueue_, void **_data_);
- **Retorno**
  - 0 se a remoção do elemento tiver sucesso, ou -1 caso contrário.
- **Descrição**
  - Extrai o elemento do topo da fila de prioridade especificado por *pqueue*. Após o retorno, *data* aponta para os dados armazenados no elemento que foi extraído. É responsabilidade do chamador gerenciar a memória associada com os dados.

Fila de Prioridade – Interface 5/6

- ***pqueue_peek***
  - void *pqueue_peek(const PQueue *pqueue);
- **Retorno**
  - Elemento de mais alta prioridade na fila de prioridade, ou NULL se a fila de prioridade estiver vazia.
- **Descrição**
  - Macro que avalia o elemento de mais alta prioridade na fila de prioridade especificada por *pqueue*.

Fila de Prioridade – Interface 6/6

- ***pqueue_size***
  - int pqueue_size(const PQueue *_pqueue_);
- **Retorno**
  - Número de elementos na fila de prioridade.
- **Descrição**
  - Macro que avalia o número de elementos na fila de prioridade especificada por *pqueue*.

Fila de Prioridade – Header 1/1

```
#ifndef PQUEUE_H
#define PQUEUE_H

#include "heap.h"

typedef Heap PQueue;

#define pqueue_init heap_init

#define pqueue_destroy heap_destroy

#define pqueue_insert heap_insert

#define pqueue_extract heap_extract

#define pqueue_peek(pqueue) ((pqueue)->tree == NULL ? NULL : (pqueue)->tree[0])

#define pqueue_size heap_size

#endif
```


Fila de Prioridade – Implementação 1/1

Não Precisa !!!!!!!

Fila de Prioridade – Sumário

- Estrutura de dados extremamente útil
- Implementação simples e eficiente através do Heap

- Muito melhor que uma simples lista encadeada ordenada