Filas de Prioridade

Heaps

Filas de Prioridade

Motivação

- Triagem de Manchester, é um sistema de gestão de urgências de um hospital, que permite classificar o grau de gravidade da situação de cada paciente, atribuindo-lhe uma das seguintes cores:

  AZUL: Não urgente

  VERDE: Pouco urgente

  AMARELO: Urgente

  LARANJA: Muito urgente

  VERMELHO: Emergente

- A ordem pela qual os utentes são atendidos depende da sua prioridade

  Exemplo: um paciente classificado com **vermelho** é atendido antes de qualquer utente classificado com **verde** ou **azul**, mesmo que estes estejam à espera há muito tempo

Fila

- Capta a noção elementar de ordem de chegada no processamento de tarefas
- No processamento de tarefas, para além da ordem de chegada é normal ter em conta a prioridade das tarefas
  - as tarefas mais prioritárias devem ser realizadas primeiro que as menos prioritárias

Fila de prioridade

- Objetos na fila têm um número indicativo da sua prioridade
- Procurar o próximo objeto a tratar: encontrar o mais prioritário da fila
- Retirar o objeto a tratar da fila: remover o mais prioritário da fila

Definição

- É uma EAD que permite que o elemento com maior ou menor valor da chave (associado à maior prioridade) seja facilmente acedido e removido da estrutura
- As EAD estudadas dependem apenas da **ordem de chegada** e não se ajustam (automaticamente) a um processo do tipo da Triagem de Manchester
  - uma **pilha** segue o lema "o último a entrar é o primeiro a sair"
  - uma **fila** segue sempre o lema "o primeiro a entrar é o primeiro a sair"
- Para simular este tipo de processos, é necessário uma EAD que tenha em consideração as **prioridades**
  - se fossem sempre só as 5 prioridades, podia-se usar 1 fila para cada prioridade
  - há casos mais gerais, onde a quantidade de diferentes prioridades não é limitada
    - a prioridade pode ser medida por um número qualquer (inteiro ou real)

Definição

- Uma **fila de prioridade** é uma EAD para guardar uma coleção de elementos e suportar três operações principais:
  - **consultar(FP)**

    devolve (sem retirar) o elemento com **maior prioridade** da Fila de Prioridade **FP**
  - **remover(FP)**

    extrai e **remove** o elemento com **maior prioridade** da Fila de Prioridade FP
  - **inserir(x, FP)**

    insere um elemento **x** na Fila de Prioridade **FP**

  e duas secundárias:
  - **atualizarChave(x, k, FP)**

    atualiza (melhora) com o valor **k** a **chave** de um elemento **x** da Fila de Prioridade **FP**
  - **removerElemento(x, FP)**

    remove o **elemento x** da Fila de Prioridade **FP** (por abandonar a fila)

Definição

- Para além das cinco operações anteriores, a gestão de uma **fila de prioridade** necessita de duas operações auxiliares:
  - **criar()**

    cria uma Fila de Prioridade vazia
  - **vazia(FP)**

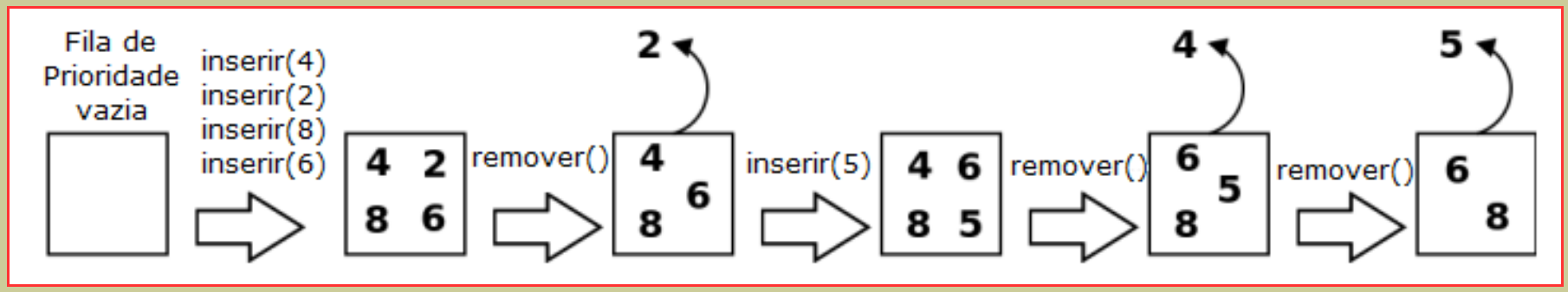
    verifica se a Fila de Prioridade **FP** está vazia

Aplicações

- Exemplos de sistemas que usam filas de prioridade
  - uma **fila** (ex: correios) com atendimento prioritário (ex: grávidas e idosos)
  - um **router** com tráfego prioritário (ex: chamadas VoIP)
  - processamento de tarefas nos sistemas operativos
  - **simulação baseada em eventos**:
    - sistema em que vários eventos começam em momentos diferentes
    - pode-se usar filas de prioridade para saber qual o próximo evento a acontecer (a hora de início de um evento mede a prioridade desse evento)
- Existem muitos algoritmos que envolvem grafos que usam filas de prioridade
  - **Algoritmo de Dijkstra** (caminhos mais curtos):
    - para determinar qual o nó mais próximo que ainda não foi processado
  - **Algoritmo de Prim** (árvore abrangente de custo mínimo):
    - para determinar qual o nó mais próximo da árvore que ainda não foi adicionada.

Implementação

- Começar por definir o que significa ser "**mais prioritário**"
- Sem perda de generalidade, assumir que os elementos que definem as prioridades são comparáveis e que o **mais prioritário** é o com **menor valor**
  - se forem valores inteiros em { 8, 4, 5 }, o menor é 4
  - no caso da triagem, associar às cores mais prioritárias números menores
    (vermelho = 1, laranja = 2, amarelo = 3, verde = 4 e azul = 5)
- Se fosse mais útil associar ao mais prioritário o maior valor, e não o menor, bastaria mudar as prioridades de forma correspondente
- Exemplo:



Implementação

- Pode-se usar representações estáticas, como as EAD que se seguem
  - Lista não ordenada, usando um array com n elementos
    - inserir um elemento é fácil (colocar na última posição): O(1)
    - remover o elemento mais prioritário implica deslocar vários elementos: O(n)
    - consultar o elemento mais prioritário implica pesquisa linear: O(n)
  - Lista ordenada (o mais prioritário na última posição), usando um array com n elementos
    - inserir um elemento implica manter a ordem: O(n)
    - remover o elemento mais prioritário implica remover o último elemento: O(1)
    - consultar o elemento mais prioritário é fácil (devolve o da última posição): O(1)

Implementação

- Pode-se usar representações dinâmicas, como as EAD que se seguem
  - Lista não ordenada, usando uma lista ligada com n elementos
    - inserir um elemento é fácil (colocar no início): O(1)
    - remover o elemento mais prioritário implica percorrer a lista: O(n)
    - consultar o elemento mais prioritário implica pesquisa linear: O(n)
  - Lista ordenada (o mais prioritário no início), usando uma lista ligada ordenada com n elementos
    - inserir um elemento implica manter a ordem: O(n)
    - remover o elemento mais prioritário implica remover o primeiro nodo: O(1)
    - consultar o elemento mais prioritário é fácil (devolve o primeiro): O(1)

Implementação

- Pode-se usar representações dinâmicas, como as EAD que se seguem
  - Árvore binária de pesquisa não balanceada com n nós
    - inserir um elemento tem um custo que depende da altura da ABP: O(n)
    - remover o elemento mais prioritário (nodo mais à esquerda da ABP): O(n)
    - consultar o elemento mais prioritário (menor valor), consiste em procurar o nodo mais à esquerda da ABP: O(n)
  - Árvore binária de pesquisa balanceada com n nós
    - inserir um elemento tem um custo que depende da altura da ABP: O(log n)
    - remover o elemento mais prioritário (nodo mais à esquerda da ABP): O(log n)
    - consultar o elemento mais prioritário (menor valor) consiste em procurar o nodo mais à esquerda da ABP: O(log n)
  - Árvores de outros tipos (**Heaps**)
    - **heap Binário**
    - **heap Binomial**

Implementação

- Eficiência (ordem de complexidade)

Estrutura Abstrata de Dados	inserir	remover	consultar
Lista não ordenada (array)	O(1)	O(n)	O(n)
Lista ordenada (array)	O(n)	O(1)	O(1)
Lista não ordenada (listas ligadas)	O(1)	O(1)	O(n)
Lista ordenada (listas ligadas)	O(n)	O(1)	O(1)
Árvore binária pesquisa	O(n)	O(n)	O(n)
Árvore binária pesquisa balanceada	O(log n)	O(log n)	O(log n)
Heap Binário			
Heap Binomial			

Heaps

Definição

- Heaps são tipos abstratos de dados usados para gerir filas de prioridades, utilizando árvores (EAD Árvore)
- Um heap pode ser de dois tipos: minHeap e maxHeap
  - um **minHeap** é uma árvore na qual, para todos os nós, o valor da chave do pai deve ser menor que o valor da chave dos seus filhos
  - um **maxHeap** é uma árvore na qual, para todos os nós, o valor da chave do pai deve ser maior que o valor da chave dos seus filhos

Operações sobre um Heap

- Operações auxiliares sobre um heap:
  - **criar** um heap vazio
  - **verificar** se um heap está vazio
- Operações principais (mais importantes) sobre um heap:
  - **consultar** um heap, que consiste em **devolver** o elemento com **maior prioridade**
  - **extrair** e **remover** o elemento com **maior prioridade** de um heap
  - **inserir** um elemento num heap
- Operações secundárias (menos importantes) sobre um heap:
  - **atualizar** (melhorar) o valor da chave de um elemento de um heap
  - **remover** um **elemento** de um heap