Heaps

Rafael Alves da Costa

ANÁLISE E DESENVOLVIMENTO DE SISTEMAS
FATEC Carapicuíba

Aula 11 - Estrutura de Dados

11/2025

Sumário

Fatec Carapicuíba

Heaps

O que é um Heap?

Heap é uma estrutura de dados baseada em árvore binária completa que mantém uma **propriedade de ordem** entre os elementos.

- É comumente utilizada para implementar **filas de prioridade**.
- Pode ser de dois tipos:
  - **Min-Heap**: o menor valor está na raiz.
  - **Max-Heap**: o maior valor está na raiz.
- Elementos são armazenados como pares `(chave, valor)`.
- Pode ser representado eficientemente usando um **array**.

Aplicações:

- Algoritmos de ordenação (Heapsort)
- Algoritmos de grafos (Dijkstra, Prim)
- Gerenciamento de tarefas com prioridades

Montes (Heaps) - Parte 1

- **Para implementar uma fila de prioridade, vimos duas abordagens:**
  - **Lista não ordenada**: inserção rápida $O(1)$, mas remoção do menor elemento é lenta $O(n)$.
  - **Lista ordenada**: remoção do menor é rápida $O(1)$, mas a inserção pode exigir $O(n)$ para manter a ordenação.
- Ambas têm limitações em termos de desempenho para grandes volumes de dados.

Montes (Heaps) - Parte 2

A estrutura de dados **heap binário** oferece uma forma eficiente de implementar filas de prioridade.

Usando a estrutura de uma **árvore binária completa** com a **propriedade de ordem de heap**, podemos garantir:

- Inserções: $O(\log n)$
- Remoções: $O(\log n)$

Isso representa uma **melhoria significativa** em relação às implementações baseadas em listas.

- **Relação de dominância de funções tipicamente utilizadas:**

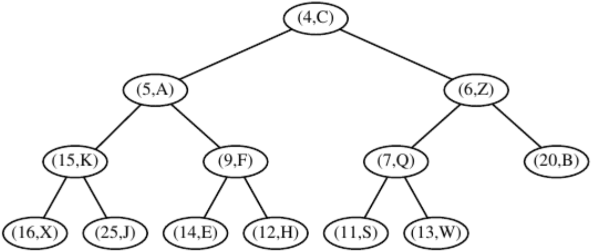
$$n! \gg 2^n \gg n^3 \gg n^2 \gg n \log n \gg n \gg \log n \gg 1$$

Fatec
Carapicuíba

Heap: Propriedade de Ordem

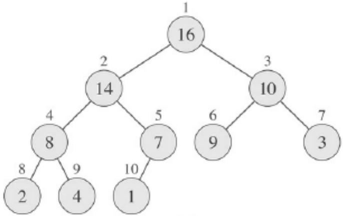
Uma **Heap** é uma árvore binária que satisfaz:

- **Propriedade estrutural:** árvore binária completa.
- **Propriedade de ordem:**
  - **Min-heap:** a chave de cada nó é maior ou igual à de seu pai $\rightarrow$ raiz contém o menor valor.
  - **Max-heap:** a chave de cada nó é menor ou igual à de seu pai $\rightarrow$ raiz contém o maior valor.

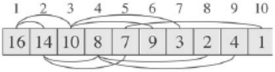


Heap: Propriedade de Árvore Binária Completa

- **Propriedade de Árvore Binária Completa:**
  - Uma heap T com altura h é uma árvore binária completa se os níveis $0, 1, 2, \ldots, h-1$ tiverem o número máximo de nós possível (i.e., 2^i nós no nível i, para $0 \leq i \leq h-1$).
  - Os nós restantes no nível h estão nas posições mais à esquerda possíveis nesse nível.
  - A representação em array de uma árvore binária completa com n elementos ocupa as posições $A[0]$ até $A[n-1]$.
- **Exemplo:** No array, uma heap com 10 elementos seria armazenada de $A[0]$ a $A[9]$, com o último nível preenchido da esquerda para a direita.



(a)



(b)

A Altura de um Heap

A altura h de uma heap com n elementos é dada por:

$$h = \lfloor \log_2 n \rfloor$$

Isso garante que as operações principais, como **inserção** e **remoção**, sejam executadas em tempo proporcional à altura da árvore, ou seja, $O(\log n)$.

Justificativa da Altura de uma Heap Completa

Considere uma heap completa com n elementos e altura h:

- Cada nível i da árvore (de 0 até $h-1$) tem 2^i nós.
- Total de nós até o penúltimo nível:

$$1 + 2 + 4 + \cdots + 2^{h-1} = \sum_{i=0}^{h-1} 2^i = 2^h - 1$$

- No nível h, podem existir de 1 até 2^h nós.
- Assim, temos:

$$2^h \leq n \leq 2^{h+1} - 1 \Rightarrow h = \lfloor \log_2 n \rfloor$$

Consequência: A altura da heap cresce lentamente com n, o que torna as operações principais eficientes: $O(\log n)$.

Fatec Carapicuíba

Operações em Fila de Prioridade com Heap

Em uma **fila de prioridade implementada com heap**:

- Cada elemento é armazenado como um par `(chave, valor)`.
- `min()` $\rightarrow$ retorna o item da raiz (mínima chave).
- `add(k, v)` $\rightarrow$ insere um novo item e ajusta a posição (*heapify-up*).
- `remove_min()` $\rightarrow$ remove a raiz e reestrutura o heap (*heapify-down*).

Como a altura da árvore é $O(\log n)$, essas operações são **eficientes** mesmo para grandes volumes de dados.

Fatec Carapicuíba

Inserção em um Heap (add)

Ao adicionar um novo elemento ao **heap**:

- O novo nó é adicionado na **próxima posição livre** para manter a **completude da árvore**.
- Após a inserção, o heap pode ser reorganizado com o procedimento **heapify-up** (subida da chave).
- Esse processo garante que a **propriedade de ordem do heap** seja preservada.

Fatec Carapicuíba

Questão?

- No slide sobre a "Propriedade de Ordem do Heap", o exemplo representa um min-heap ou um max-heap?
  Justifique sua resposta com base na relação entre pais e filhos.

Min-Heap vs Max-Heap: Semelhanças Estruturais

Semelhanças estruturais:

- Ambos são **árvores binárias completas** com a mesma estrutura interna.
- Ambos mantêm a **propriedade de heap**:
  - **Min-heap**: cada pai é **menor ou igual** aos filhos.
  - **Max-heap**: cada pai é **maior ou igual** aos filhos.
- Ambos armazenam os elementos em um **array** com regras idênticas:
  - Pai de índice i: $A[\lfloor (i-1)/2 \rfloor]$
  - Filhos de índice i: $A[2i+1]$ e $A[2i+2]$

Fatec Carapicuíba

Min-Heap vs Max-Heap: Diferenças de Acesso

Diferenças principais: acesso ao mínimo ou ao máximo

Operação	Min-Heap	Max-Heap
`min()`	$O(1)$ – está na raiz	$O(n)$ – percorre tudo
`max()`	$O(n)$ – percorre tudo	$O(1)$ – está na raiz
`inserção`	$O(\log n)$ – heapify-up	$O(\log n)$ – heapify-up
`remove_min()`	$O(\log n)$ – remove raiz	$O(n)$ – encontra menor
`remove_max()`	$O(n)$ – encontra maior	$O(\log n)$ – remove raiz

Min-Heap vs Max-Heap: Conclusão

Conclusão:

- Ambas as estruturas realizam **inserção** e **remoção da raiz** em $O(\log n)$, devido à necessidade de reorganização (*heapify-up/down*).
- A principal diferença está em **qual valor** (mínimo ou máximo) está disponível diretamente na raiz:
  - Use um **min-heap** para acessar ou remover rapidamente o **menor valor**.
  - Use um **max-heap** para acessar ou remover rapidamente o **maior valor**.

Comparação de Estruturas de Fila de Prioridade

Estrutura	Inserção	Remoção Mínimo	Busca Mínimo
Lista não ordenada	$O(1)$	$O(n)$	$O(n)$
Lista ordenada	$O(n)$	$O(1)$	$O(1)$
Heap (min-heap)	$O(\log n)$	$O(\log n)$	$O(1)$
Heap (max-heap)	$O(\log n)$	$O(n)$	$O(n)$

Comparação de complexidade entre diferentes implementações de filas de prioridade.

Fatec Carapicuíba

Considerações

Considerações

Considerações

- **Recordando!!!**
  - Heaps.
  - Exercícios.

Fatec Carapicuíba

Considerações

- **Recordando!!!**
  - Heaps.
  - Exercícios.
- **Referências utilizadas!!!**

  - Cormen, T. H. et al. Algoritmos: Teoria e prática, 3a edição. Elsevier, 2012.
  - Levitin, A. Introduction to the Design and Analysis of Algorithms, 2007.
  - Goodrich, M. T., Tamassia, R., Goldwasser, M. H. (2013). Data structures and algorithms in Python.
  - Goodrich, M. T., Tamassia, R., Goldwasser, M. H. (2014). Data Structures and Algorithms in Java.