CI1056: Algoritmos e Estruturas de Dados II

Prof. Dr. Marcos Castilho

Departamento de Informática/UFPR

4 de dezembro de 2020

Resumo

Filas de prioridade

Objetivos da aula

- Apresentar o conceito fila de prioridades
- Apresentar os principais algoritmos
- Caracterizar a fila de prioridades como um Tipo Abstrato de Dados

Motivação

- Em algoritmos I vimos duas estruturas de dados importantes:
  - Filas (First In First Out)
  - Pilhas (Last In First Out)
- Ambas têm aplicações importantes e estão presente em muitos sistemas computacionais
- As filas de prioridade também têm inúmeras aplicações:
  - Quando queremos atribuir prioridades para os elementos
  - Exemplos: controle de processos
  - O celular tem que atender a chamada com maior prioridade do que o seu jogo

Filas de prioridade

- Esta estrutura de dados tem duas operações de interesse:
  - Remover o máximo
  - Inserir
- Evidentemente queremos que os algoritmos que façam estas operações sejam eficientes
- Veremos uma forma de implementar que faz as operações em tempo logarítmico
- Depois usaremos esta estrutura de dados para apresentar um dos mais eficientes algoritmos de ordenação, o *heapsort*

Estruturas de dados

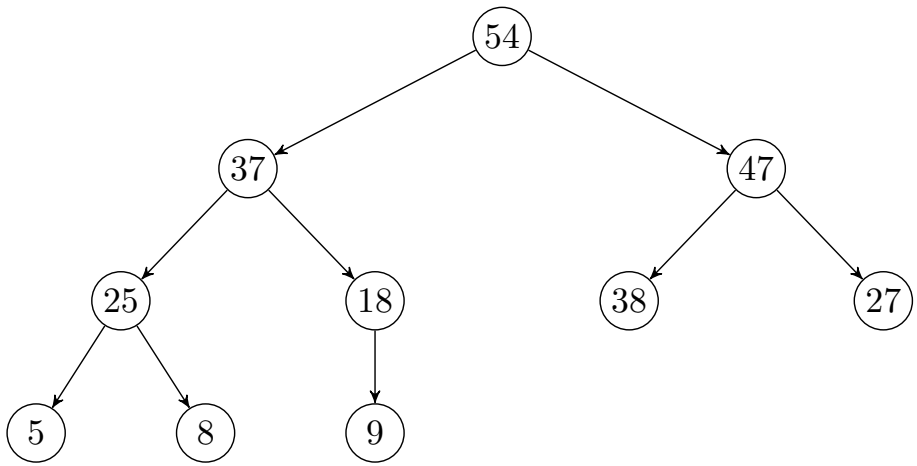
- Vimos em Algoritmos I que um vetor, por exemplo não é por si só uma estrutura de dados
- A forma como os elementos estão distribuídos no vetor importa
- Isto é: a propriedade associada aos elementos do vetor
  - Vetor ordenado ou não
  - Elementos do início ao fim ou do fim ao início, ...
- Da mesma maneira, vimos que vetores podem ser utilizados para representar pilhas e filas
- Vimos que a estrutura de dados influencia muito na eficiência dos algoritmos

A estrutura de dados Fila de Prioridade

- Implementaremos Filas de prioridades usando vetores
- Utilizaremos a versão conhecida como *heap* binário
- Nela, como só temos duas operações, inserção e remoção do máximo, não usaremos nem um vetor qualquer, nem um vetor completamente ordenado
- A propriedade sobre os elmentos do vetor é algo diferente, conforme veremos
- Inicialmente, apresentaremos a estrutura do ponto de vista abstrato, depois veremos como implementar usando vetores

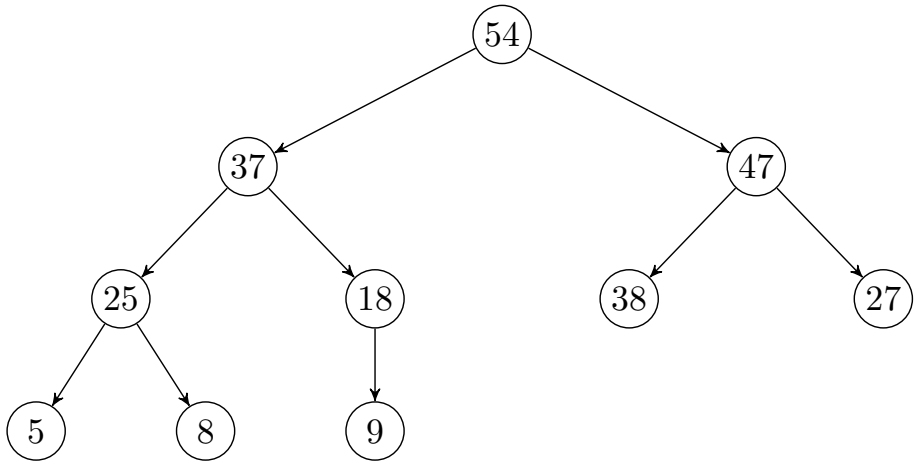
Heap binário

Um *heap* binário é uma *árvore binária quase completa*

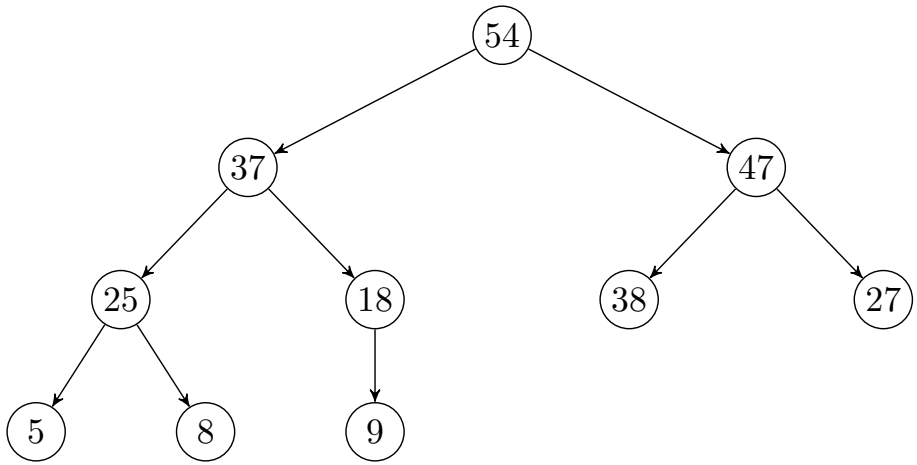


Definições

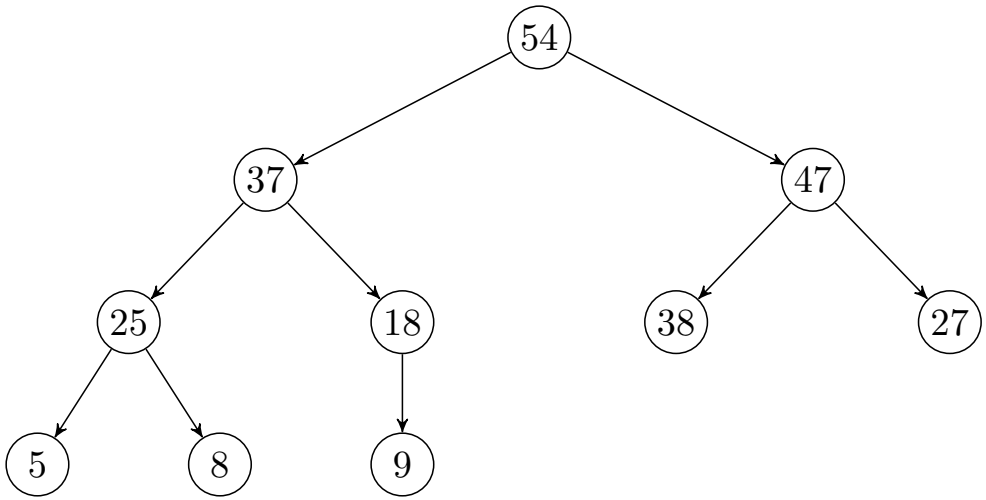
Árvore, nodo, aresta, nó raiz, filhos (da esquerda e da direita) e folhas



Subárvore, descendentes



Altura de um nó e altura da raiz



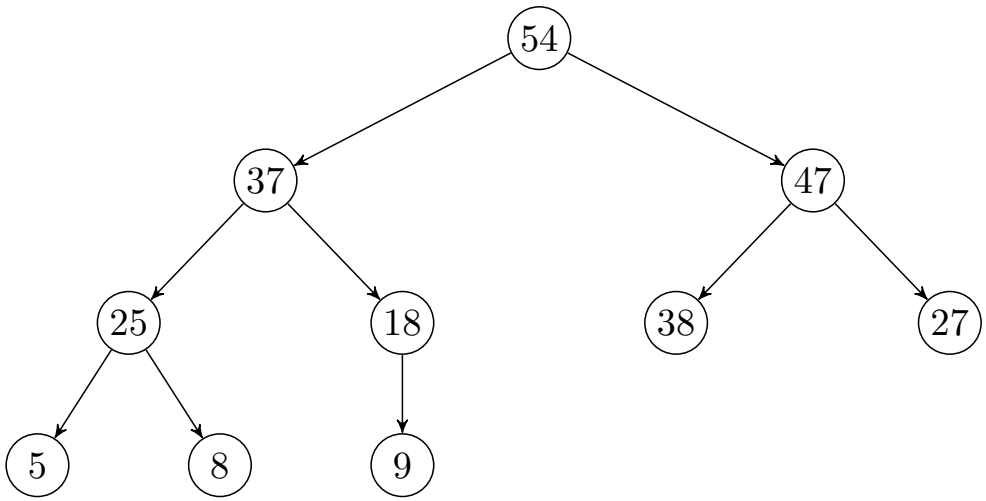
Tipos de *heaps*

- *heap* de máximo
- *heap* de mínimo

Heap em inglês é sinônimo de pilha, amontoado, bateria (no sentido de bateria de testes), ... Nós utilizaremos o termo em inglês, pois no Brasil não há uma tradução de consenso.

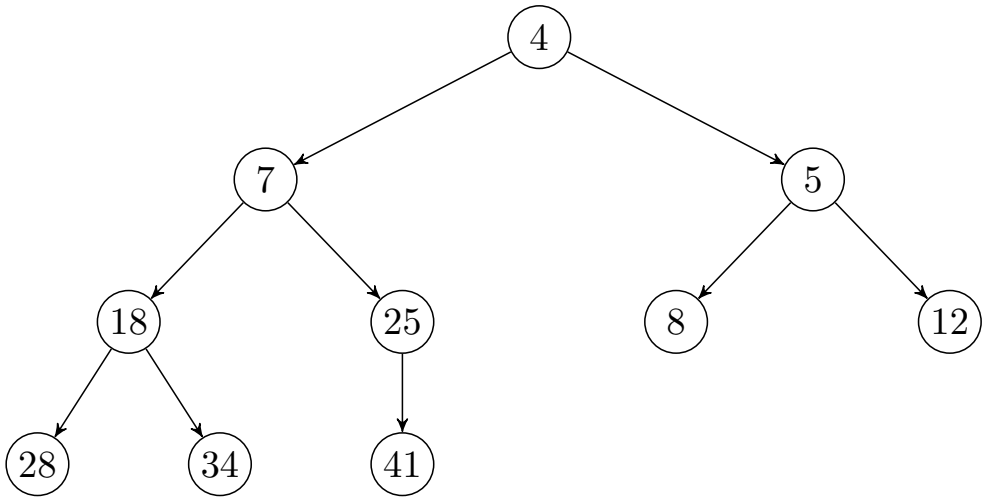
Heap de máximo

Exemplo:



Heap de mínimo

Exemplo



Observações

- Não existe ordem relativa entre elementos de mesmo nível
- Mas o elemento da raiz de uma subárvore é maior (*heap* de máximo) ou menor (*heap* de mínimo) do que todos os seus descendentes

Fim do tópico

- O conteúdo desta aula está no livro Cormen, Leiserson, Rivest e Stein, no capítulo 6, seções 6.1 e 6.2. Também está no livro Sedgewick e Wayne, seção 2.4, conforme referência já citadas

Licença

- Slides feitos em LaTeX usando beamer
- Licença

  Creative Commons Atribuição-Uso Não-Comercial-Vedada a Criação de Obras Derivadas 2.5 Brasil License.`http://creativecommons.org/licenses/by-nc-nd/2.5/br/`

  Creative Commons Atribuição-Uso Não-Comercial-Vedada a Criação de Obras Derivadas 2.5 Brasil License.`http://creativecommons.org/licenses/by-nc-nd/2.5/br/`