CI1056: Algoritmos e Estruturas de Dados II

Prof. Dr. Marcos Castilho

Departamento de Informática/UFPR

4 de dezembro de 2020

Resumo

Implementação de *Heaps* binários

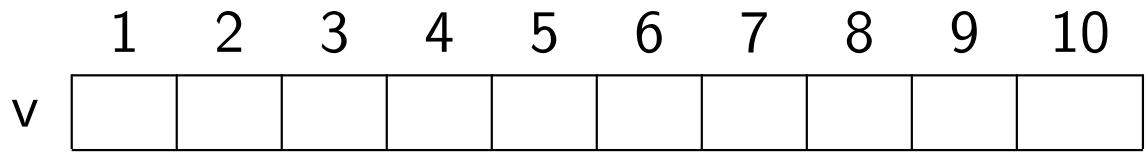
Objetivos da aula

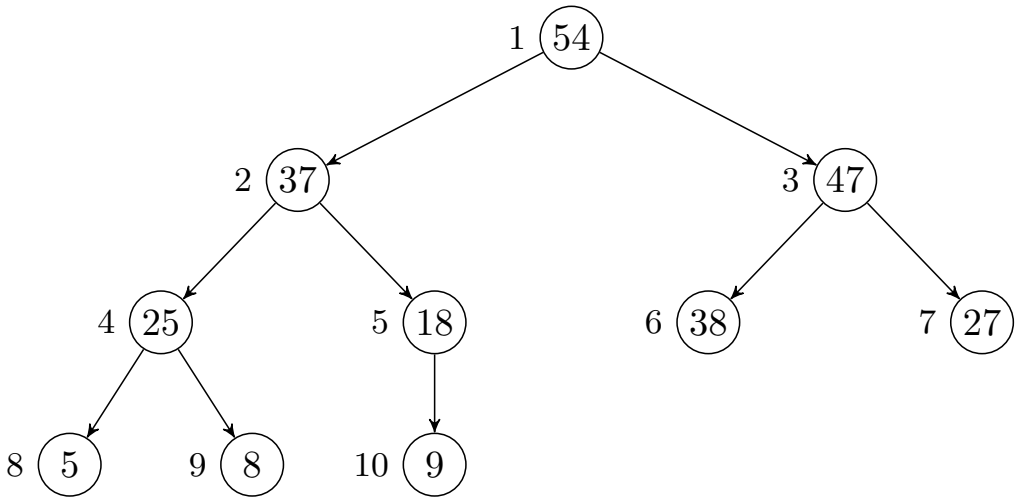
- Detalhar o conceito de *heap* binário e uma representação usando vetores

Implementação em vetores

- É possível representar um *heap* usando vetores
- O primeiro elemento (índice 1) contém a raiz
- Todos os outros elementos têm a seguinte propriedade
  - O pai do nodo i está no índice $i/2$ (divisão inteira)
  - O filho da esquerda do nodo i está no índice $2i$
  - O filho da direita do nodo i está no índice $2i + 1$
- Estas operações são extremamente rápidas nos computadores

Exemplo para um *heap* de máximo





Propriedade de *heap* de máximo

- Seja $pai(i)$ o índice do pai do nodo i em *heap* de máximo
- Então:

$$v[pai(i)] \geq v[i], \textbf{ para todo nodo } i\textbf{, exceto a raiz.}$$

- Assim, o maior elememento do vetor está na raiz ($v[1]$)

Propriedade de *heap* de mínimo

- Seja $pai(i)$ o índice do pai do nodo i em *heap* de máximo
- Então:

$$v[pai(i)] \leq v[i], \textbf{ para todo nodo } i\textbf{, exceto a raiz.}$$

- Assim, o menor elememento do vetor está na raiz ($v[1]$)

Aplicações

- *Heaps* de máximo podem ser usadas em algoritmos de ordenação, veremos o *heapsort*, em breve
- *Heaps* de mínimo podem ser usadas em sistemas operacionais, ou em qualquer situação que dependa de escolher o menor elemento de um conjunto eficientemente

Altura de um *heap*

- Não entraremos em detalhes, mas a altura máxima de um *heap* é em função de $log_2(n)$
- Como os algoritmos que veremos logo a seguir dependem da altura da árvore, veremos que eles terão complexidade $log_2 n$
- Claro que para encontrar o máximo em um *heap* de máximo terá tempo constante, idem para o mínimo em um *heap* de mínimo

O tipo abstrato de dados

- A seguir veremos os algoritmos que manipulam o *heap* de máximo
- Os equivalentes para *heap* de mínimo são trivialmente obtidos a partir destes
- Um tipo abstrato de dados deve prover funções que manipulam os dados de modo a **manter a propriedade** do tipo, no caso a propriedade de *heap* de máximo

O tipo abstrato de dados

- São duas as funções:
  - Inserir
  - Remover o máximo
- Evidentemente teremos funções internas (restritas) para ajudar e também podemos ter algumas outras de interesse, tais como testar se um *heap* está vazio, . . .
- Nós começaremos com algoritmos internos que garantes que a **propriedade de heap de máximo é mantida**

Max-heapify: manutenção da propriedade de *heap*

- Instância: (v, i, n), onde v é um vetor e $i \in [1..n]$, onde n é o tamanho do vetor $v[1..n]$, de modo que:
  - Seja $esquerda(i)$ a subárvore que tem como raiz o nodo filho da esquerda do nodo i
  - Seja $direita(i)$ a subárvore que tem como raiz o nodo filho da direita do nodo i
  - $esquerda(i)$ e $direita(i)$ são *heaps* de máximo
  - Eventualmente, $v[i]$ pode ser menor que seus filhos, e para este nodo, é aceitável que ele viole a propriedade de *heap* de máximo
- Resposta: retorna um *heap* com a propriedade de máximo, conforme definido acima

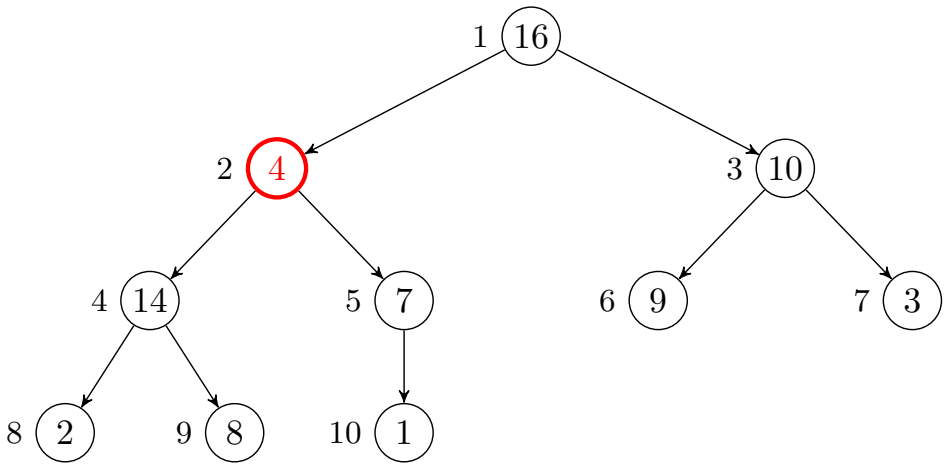
O algoritmo *Max-heapify*

```
max-heapify (v, i)
    esq = esquerda (i)
    dir = direita (i)
    se esq <= n e v[esq] > v[i]
        maior = esq
    senao
        maior = i
    se dir <= n e v[dir] > v[maior]
        maior = dir
    se maior != i
        troca v[i] com v[maior]
        max-heapify (v, maior)
```

A ideia é que o elemento v[i] vá para sua posição, recursivamente, e o *heap*, ao final, tenha a propriedade de *heap* de máximo

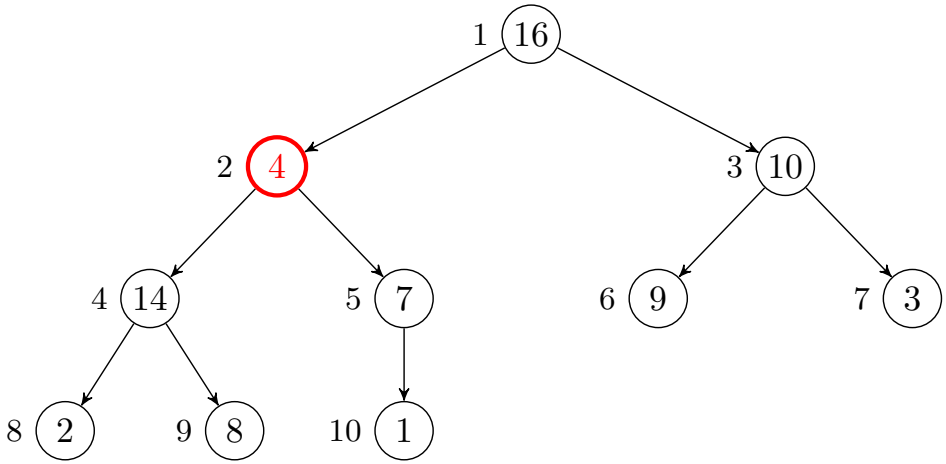
Exemplo

Observe que o nodo em vermelho não satisfaz a propriedade de *heap* de mínimo, mas as duas subárvores de esquerda e da direita sim



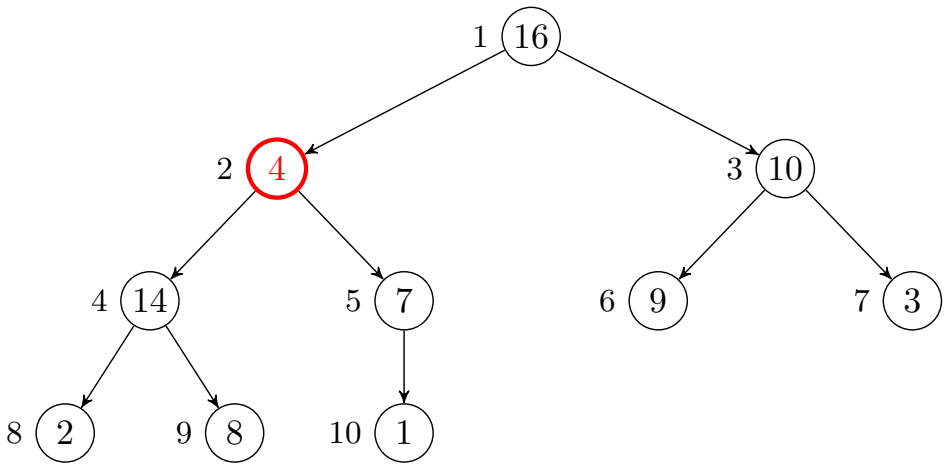
Exemplo

Entre o 14 e o 7, quem tem que subir? Evidentemente o 14!
Explique



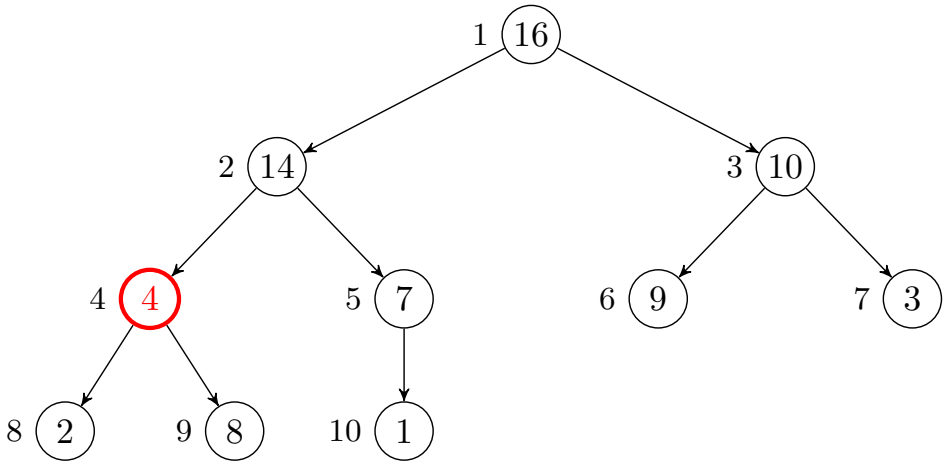
Exemplo

O algoritmo escolhe o maior dos três envolvidos e o faz subir!



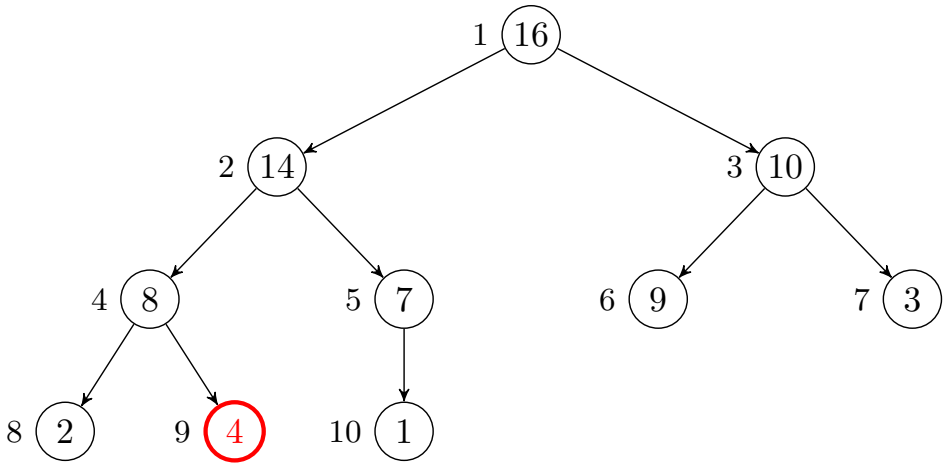
Exemplo

Resultado de $max - heapify(v, 2)$. Agora o nodo em vermelho está errado. Aplicar $max - heapify(v, 4)$



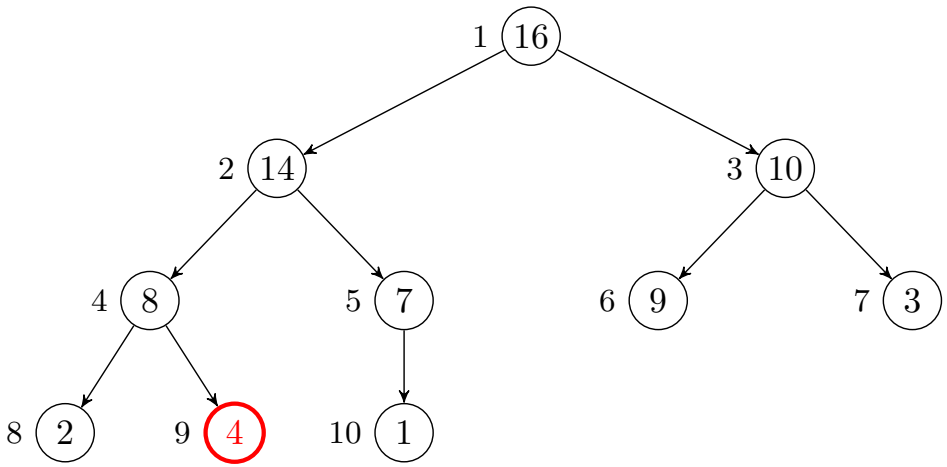
Exemplo

Resultado de $max - heapify(v, 4)$.



Exemplo

A função ainda chama recursivamente $max - heapify(v, 9)$, mas este é o caso base e não ocorre nada. O algoritmo termina.



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/`