Revisão de Estruturas de Dados

Motivação

- Em simulação frequentemente precisamos encontrar o elemento que tem o menor/maior valor de um dado atributo (ex.: tempo do evento)
- A escolha da estrutura depende do número de elementos
  - Estrutura simples tendem a ter melhor desempenho para pouco elementos
  - Estruturas com melhor comportamento assintótico (simples ou não) tendem a ter melhor desempenho para conjuntos grandes

Vetor

- Desvantagem quando temos grande variação no número de elementos: tamanho estático
  - Tamanho inicial grande -> desperdício de memória
  - Redimensionamento -> pode causar cópia de todos os elementos anteriores para uma nova região de memória
- Não ordenado
  - Localizar menor/maior: O(n)
  - Inserir e remover: O(1)

Vetor

- Ordenado
  - Localizar menor/maior: O(1)
  - Inserir e remover: O(n)

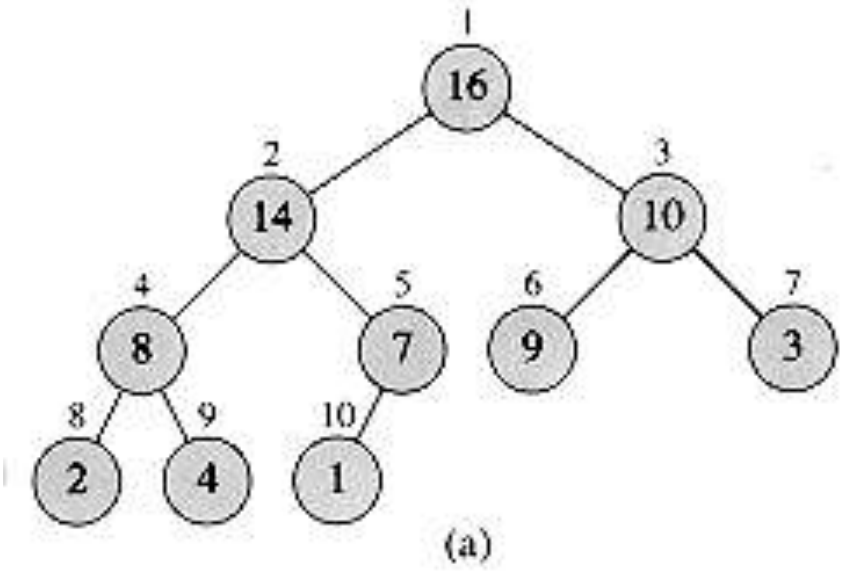
Lista encadeada

- Mais apropriada para quantidade dinâmica
- Não ordenada
  - Menor/maior: O(n)
  - Inserir e remover: O(1)
- Ordenada
  - Menor/maior: O(1)
  - Inserir: O(n)
  - Remover: O(1)

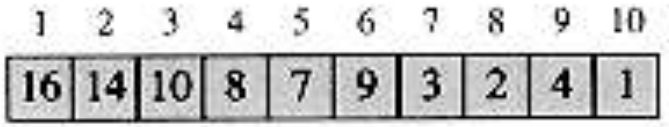
Heap

- Complexidade
  - Menor/maior: O(1)
  - Inserir e remover: O(log(n))
- Árvore com propriedade: pai menor/maior ou igual aos filhos

Heap



(a)



(b)

- Implementação com vetor:
  - pai(i) = piso(i/2)
  - filho_esquerda(i) = 2i
  - filho_direita(i) = 2i + 1

Heap: mantendo a propriedade

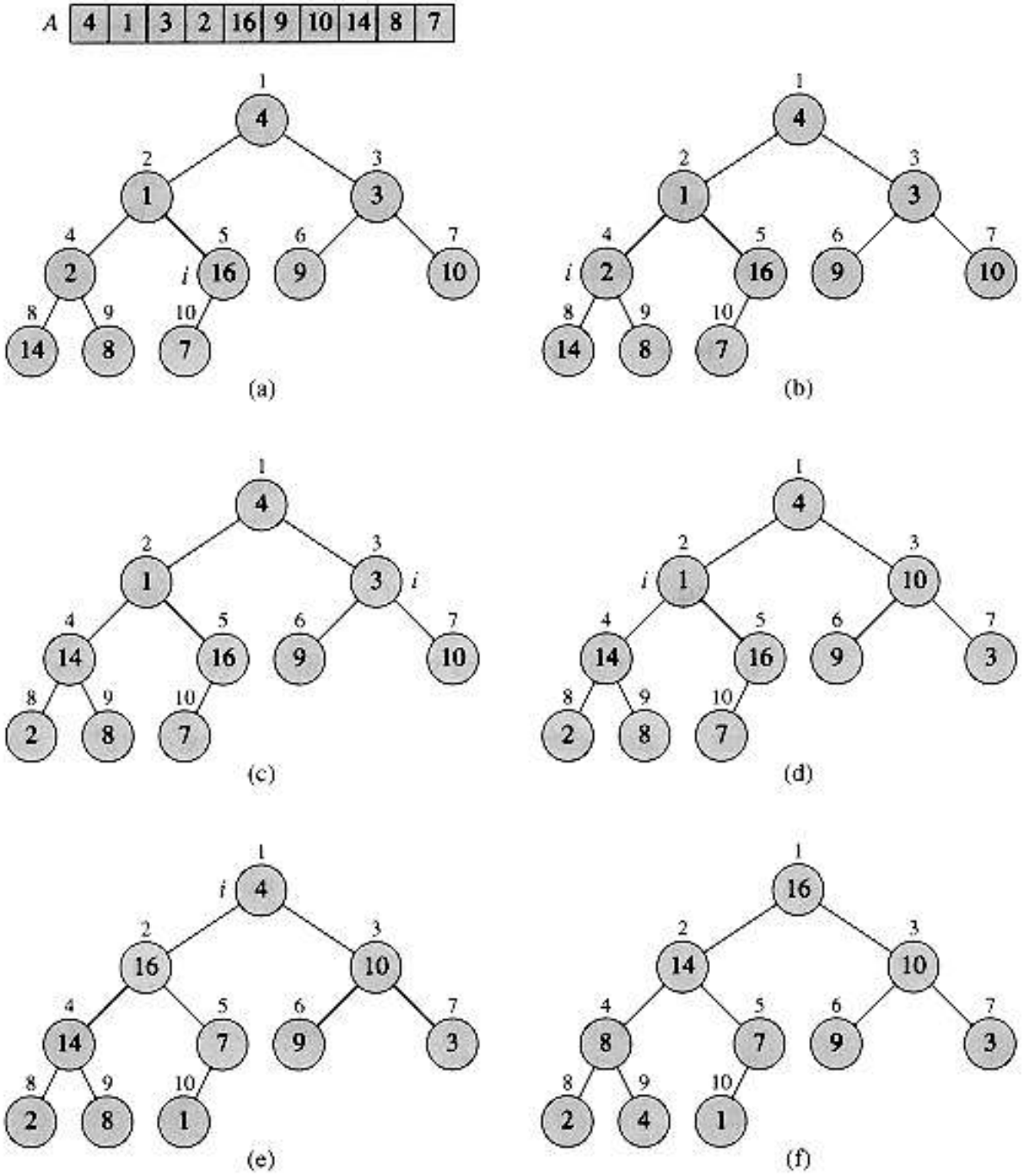
- Heapify(V, i)
  - Sejam ‘e’ e ‘d’ a posição dos filhos esq. e dir. de i
  - Seja m o maior valor entre V[i], V[e], V[d], e k a indice do maior valor
  - Se m é diferente de V[i],
    - Troca V[i] com V[k]
    - Heapify(V, k)

Heap: construção O(n)

- Para i indo do piso(tamanho/2) até 1,
  - Heapfy(V,i)

$$\sum_{h=0}^{\lfloor \lg n \rfloor} \left\lceil \frac{n}{2^{h+1}} \right\rceil O(h) = O\left(n \sum_{h=0}^{\lfloor \lg n \rfloor} \frac{h}{2^h} \right) .$$

$$\begin{aligned} O\left(n \sum_{h=0}^{\lfloor \lg n \rfloor} \frac{h}{2^h} \right) &= O\left(n \sum_{h=0}^{\infty} \frac{h}{2^h} \right) \\ &= O(n) . \end{aligned}$$


A
4 1 3 2 16 9 10 14 8 7
(a)
(b)
(c)
(d)
(e)
(f)

Heap: removendo o topo (menor/maior) – O(log(n))

- Copie o último elemento no vetor para a primeira posição (topo)
- Chame Heapfy(V, 1)

Heap: inserir O(log(n))

- Coloque o novo elemento da última posição do vetor (posição i)
- Enquanto a chave da posição i for menor que a chave do pai de i,
  - Troque i com o pai de i
  - Faça i igual ao pai(i)