Lista de Prioridades

HEAP

Prof. Kennedy Lopes

Universidade Federal do Semi-árido

3 de agosto de 2023

Prioridades

Algumas aplicações precisam recuperar rapidamente um dado de maior prioridade.

Exemplo: Lista de tarefas

- **A cada momento, deve-se executar a tarefa com maior prioridade;**

Prioridades

Algumas aplicações precisam recuperar rapidamente um dado de maior prioridade.

Exemplo: Lista de tarefas

- A cada momento, deve-se executar a tarefa com maior prioridade;
- **Selecionar a tarefa mais prioritária de uma lista e retirá-la da lista;**

Prioridades

Algumas aplicações precisam recuperar rapidamente um dado de maior prioridade.

Exemplo: Lista de tarefas

- A cada momento, deve-se executar a tarefa com maior prioridade;
- Selecionar a tarefa mais prioritária de uma lista e retirá-la da lista;
- **Prioridades podem mudar;**

Prioridades

Algumas aplicações precisam recuperar rapidamente um dado de maior prioridade.

Exemplo: Lista de tarefas

- A cada momento, deve-se executar a tarefa com maior prioridade;
- Selecionar a tarefa mais prioritária de uma lista e retirá-la da lista;
- Prioridades podem mudar;
- **Novas tarefas podem chegar e precisam ser acomodadas.**

Prioridades

Algumas aplicações precisam recuperar rapidamente um dado de maior prioridade.

Exemplo: Lista de tarefas

- A cada momento, deve-se executar a tarefa com maior prioridade;
- Selecionar a tarefa mais prioritária de uma lista e retirá-la da lista;
- Prioridades podem mudar;
- Novas tarefas podem chegar e precisam ser acomodadas.

Nova estrutura idealizada:

Prioridades

Algumas aplicações precisam recuperar rapidamente um dado de maior prioridade.

Exemplo: Lista de tarefas

- A cada momento, deve-se executar a tarefa com maior prioridade;
- Selecionar a tarefa mais prioritária de uma lista e retirá-la da lista;
- Prioridades podem mudar;
- Novas tarefas podem chegar e precisam ser acomodadas.

Nova estrutura idealizada:

- **Os dados possuem prioridades de acesso;**

Prioridades

Algumas aplicações precisam recuperar rapidamente um dado de maior prioridade.

Exemplo: Lista de tarefas

- A cada momento, deve-se executar a tarefa com maior prioridade;
- Selecionar a tarefa mais prioritária de uma lista e retirá-la da lista;
- Prioridades podem mudar;
- Novas tarefas podem chegar e precisam ser acomodadas.

Nova estrutura idealizada:

- Os dados possuem prioridades de acesso;
- **Essa prioridade modifica com o decorrer do tempo;**

Prioridades

Algumas aplicações precisam recuperar rapidamente um dado de maior prioridade.

Exemplo: Lista de tarefas

- A cada momento, deve-se executar a tarefa com maior prioridade;
- Selecionar a tarefa mais prioritária de uma lista e retirá-la da lista;
- Prioridades podem mudar;
- Novas tarefas podem chegar e precisam ser acomodadas.

Nova estrutura idealizada:

- Os dados possuem prioridades de acesso;
- Essa prioridade modifica com o decorrer do tempo;
- **Seria interessante manter o dado mais acessado em posição de destaque.**

HEAP - Lista de Prioridade

Definição

- Representa uma tabela, na qual cada um de seus dados está associado a uma prioridade;

HEAP - Lista de Prioridade

Definição

- Representa uma tabela, na qual cada um de seus dados está associado a uma prioridade;
- Objetivo: Descrever uma estrutura de dados que realize as operações abaixo eficientemente:

HEAP - Lista de Prioridade

Definição

- Representa uma tabela, na qual cada um de seus dados está associado a uma prioridade;
- Objetivo: Descrever uma estrutura de dados que realize as operações abaixo eficientemente:
  - Seleção do elemento de maior prioridade;

HEAP - Lista de Prioridade

Definição

- Representa uma tabela, na qual cada um de seus dados está associado a uma prioridade;
- Objetivo: Descrever uma estrutura de dados que realize as operações abaixo eficientemente:
  - Seleção do elemento de maior prioridade;
  - Inserção de um elemento com prioridade específica;

HEAP - Lista de Prioridade

Definição

- Representa uma tabela, na qual cada um de seus dados está associado a uma prioridade;
- Objetivo: Descrever uma estrutura de dados que realize as operações abaixo eficientemente:
  - Seleção do elemento de maior prioridade;
  - Inserção de um elemento com prioridade específica;
  - Remoção do dado de maior prioridade;

HEAP - Lista de Prioridade

Definição

- Representa uma tabela, na qual cada um de seus dados está associado a uma prioridade;
- Objetivo: Descrever uma estrutura de dados que realize as operações abaixo eficientemente:
  - Seleção do elemento de maior prioridade;
  - Inserção de um elemento com prioridade específica;
  - Remoção do dado de maior prioridade;
  - Alteração de prioridade de um dado.

- **Lista Linear (vetor) composta de elementos com chaves** $s_1, s_2, \ldots s_n$**;**

[1] $\lfloor x \rfloor$: Maior inteiro menor ou igual a x (Piso)

Definições

HEAP

- Lista Linear (vetor) composta de elementos com chaves $s_1, s_2, \ldots s_n$;
- **As chaves representam as prioridades;**

[1] $\lfloor x \rfloor$: Maior inteiro menor ou igual a x (Piso)

Definições

HEAP

- Lista Linear (vetor) composta de elementos com chaves $s_1, s_2, \ldots s_n$;
- As chaves representam as prioridades;
- **Não existem dois elementos com a mesma prioridades;**

[1] $\lfloor x \rfloor$: Maior inteiro menor ou igual a x (Piso)

Definições

HEAP

- Lista Linear (vetor) composta de elementos com chaves $s_1, s_2, \ldots s_n$;
- As chaves representam as prioridades;
- Não existem dois elementos com a mesma prioridades;
  - **Heap Máximo: Chaves $s_1, \ldots s_n$, sendo $s_i \leq \lfloor s_{i/2} \rfloor$[1].**

[1] $\lfloor x \rfloor$: Maior inteiro menor ou igual a x (Piso)

Definições

HEAP

- Lista Linear (vetor) composta de elementos com chaves $s_1, s_2, \ldots s_n$;
- As chaves representam as prioridades;
- Não existem dois elementos com a mesma prioridades;
  - Heap Máximo: Chaves $s_1, \ldots s_n$, sendo $s_i \leq \lfloor s_{i/2} \rfloor$[1].
  - **Heap Mínimo: Chaves $s_1, \ldots s_n$, sendo $s_i \geq \lfloor s_{i/2} \rfloor$.**

[1] $\lfloor x \rfloor$: Maior inteiro menor ou igual a x (Piso)

Exemplo

Construir uma HEAP com as entradas abaixo:

1	2	3	4	5	6	7	8
95	60	78	39	28	66	70	33

Exemplo

Verifique qual(is) sequências são HEAP:

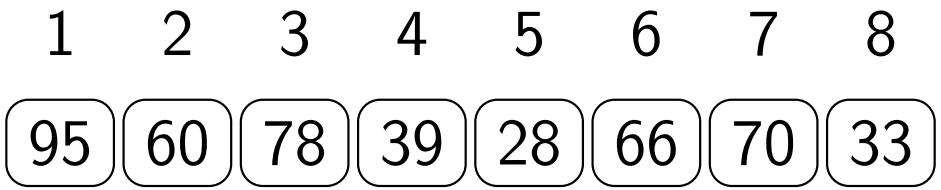
1	2	3	4	5	6	7	8	9
33	32	28	31	29	26	25	30	27

1	2	3	4	5	6	7	8	9
33	32	28	31	26	29	25	30	27

Propriedades

Para um determinado elemento i:

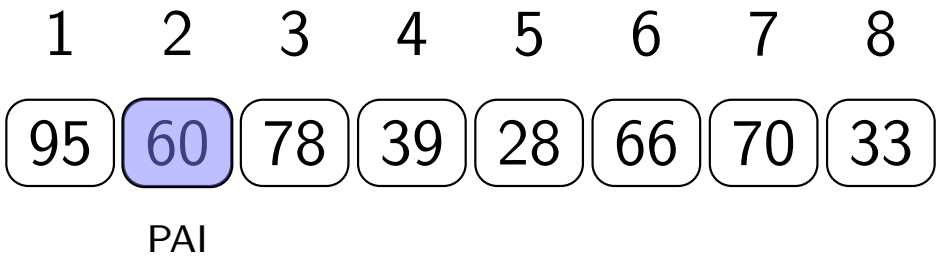
- Pai de i é $\lfloor i/2 \rfloor$;
- Filho esquerdo é $i * 2$;
- Filho direito é $i * 2 + 1$;



Propriedades

Para um determinado elemento i:

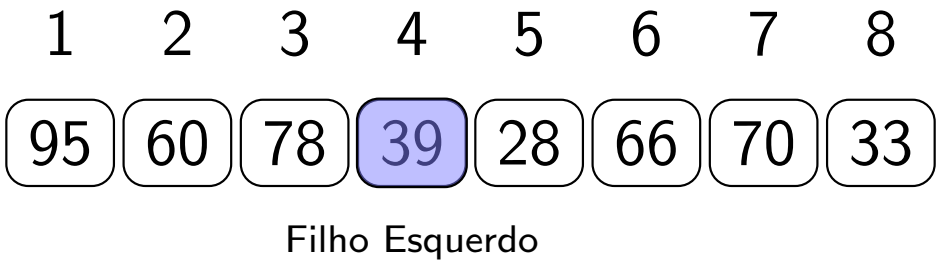
- **Pai de i é $\lfloor i/2 \rfloor$;**
- Filho esquerdo é $i * 2$;
- Filho direito é $i * 2 + 1$;



Propriedades

Para um determinado elemento i:

- Pai de i é $\lfloor i/2 \rfloor$;
- **Filho esquerdo é $i * 2$;**
- Filho direito é $i * 2 + 1$;



Propriedades

Para um determinado elemento i:

- Pai de i é $\lfloor i/2 \rfloor$;
- Filho esquerdo é $i * 2$;
- **Filho direito é $i * 2 + 1$;**

Alteração de Prioridade

Alteração de Prioridade

Ao alterar a prioridade de um nó, é necessário re-arrumar a Heap para que ela respeite as prioridades.

- Um nó que tem a prioridade aumentada precisa *subir* na árvore.

Exemplo: Mudar a prioridade de 66 para 98.

1	2	3	4	5	6	7	8
95	60	78	39	28	66	70	33

Exemplo: Mudar a prioridade de 95 para 37.

1	2	3	4	5	6	7	8
95	60	78	39	28	66	70	33

Alteração de Prioridade

Ao alterar a prioridade de um nó, é necessário re-arrumar a Heap para que ela respeite as prioridades.

- Um nó que tem a prioridade aumentada precisa *subir* na árvore.
- Um nó que tem a prioridade diminuída precisa *descer* na árvore.

Exemplo: Mudar a prioridade de 66 para 98.

1	2	3	4	5	6	7	8
95	60	78	39	28	66	70	33

Exemplo: Mudar a prioridade de 95 para 37.

1	2	3	4	5	6	7	8
95	60	78	39	28	66	70	33

Exemplo: Mudar a prioridade de 66 para 98.

1	2	3	4	5	6	7	8
95	60	78	39	28	66	70	33

Exemplo: Mudar a prioridade de 95 para 37.

1	2	3	4	5	6	7	8
95	60	78	39	28	66	70	33

Inserção

Procedimentos:

- **Tabela com n elementos;**

Inserção

Procedimentos:

- Tabela com n elementos;
- **Inserir novo elemento na posição $(n+1)$;**

Inserção

Procedimentos:

- Tabela com n elementos;
- Inserir novo elemento na posição $(n+1)$;
- **Compare o elemento no final do heap e faça-o subir até sua posição correta:**

Inserção

Procedimentos:

- Tabela com n elementos;
- Inserir novo elemento na posição $(n+1)$;
- Compare o elemento no final do heap e faça-o subir até sua posição correta:
  - **Se estiver em ordem, a inserção terminou;**

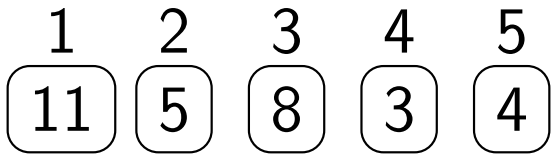
Inserção

Procedimentos:

- Tabela com n elementos;
- Inserir novo elemento na posição $(n+1)$;
- Compare o elemento no final do heap e faça-o subir até sua posição correta:
  - Se estiver em ordem, a inserção terminou;
  - **Se não estiver em ordem, troque com o pai e repita o processo até terminar ou chegar à raiz.**

Exemplo de inserção:

Inserir o elemento 15 na Heap abaixo:



Remoção

Procedimento:

- **Retira-se sempre a raiz (elemento com maior prioridade):**

Remoção

Procedimento:

- Retira-se sempre a **raiz** (elemento com maior prioridade):
- **Coloque na raiz o último elemento da Heap e faça-o** *descer* **até a posição correta.**

Remoção

Procedimento:

- Retira-se sempre a **raiz** (elemento com maior prioridade):
- Coloque na raiz o **último elemento** da Heap e faça-o *descer* até a posição correta.
- **Compare o elemento com seus filhos:**

Remoção

Procedimento:

- Retira-se sempre a **raiz** (elemento com maior prioridade):
- Coloque na raiz o **último elemento** da Heap e faça-o *descer* até a posição correta.
- Compare o elemento com seus filhos:
  - **Se estiver em ordem, a remoção terminou.**

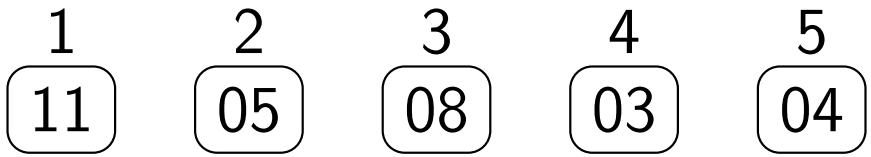
Remoção

Procedimento:

- Retira-se sempre a **raiz** (elemento com maior prioridade):
- Coloque na raiz o **último elemento** da Heap e faça-o *descer* até a posição correta.
- Compare o elemento com seus filhos:
  - Se estiver em ordem, a remoção terminou.
  - **Se não estiver em ordem, troque com o maior filho e repita o processo até terminar ou chegar a um nó folha.**

Exemplo de remoção

Remova o elemento da Heap abaixo:



Construção de Lista de Prioridades

Dado uma lista L de elementos para o qual se deseja construir uma heap H, há três alternativas:

1. **Considerar uma heap vazia e inserir elemento a elemento;**

Construção de Lista de Prioridades

Dado uma lista L de elementos para o qual se deseja construir uma heap H, há três alternativas:

1. Considerar uma heap vazia e inserir elemento a elemento;
2. **Considerar que a lista L já representa uma Heap, e corrigir as prioridades:**

Construção de Lista de Prioridades

Dado uma lista L de elementos para o qual se deseja construir uma heap H, há três alternativas:

1. Considerar uma heap vazia e inserir elemento a elemento;
2. Considerar que a lista L já representa uma Heap, e corrigir as prioridades:

   2.1 **Considerar os nós folhas corretos;**

Construção de Lista de Prioridades

Dado uma lista L de elementos para o qual se deseja construir uma heap H, há três alternativas:

1. Considerar uma heap vazia e inserir elemento a elemento;
2. Considerar que a lista L já representa uma Heap, e corrigir as prioridades:
   - 2.1 Considerar os nós folhas corretos;
   - 2.2 **Corrigir os nós internos realizando descidas.**

Construção de Lista de Prioridades

Dado uma lista L de elementos para o qual se deseja construir uma heap H, há três alternativas:

1. Considerar uma heap vazia e inserir elemento a elemento;
2. Considerar que a lista L já representa uma Heap, e corrigir as prioridades:
   - 2.1 Considerar os nós folhas corretos;
   - 2.2 Corrigir os nós internos realizando descidas.
3. **Corrigir (subir) os nós a partir dos nós folhas. Incluindo processo de ordenação.**

Exemplo: Contrução de uma Heap

Construa uma Heap a partir da lista L abaixo:

$$L = \{28, 33, 39, 60, 66, 70, 78, 95\}$$

Ordenação a partir de uma Heap (HEAPSORT)

A partir de uma heap, é possível ordenar os dados fazendo trocas:

- **O maior elemento(raiz) é trocado com o último elemento;**

Ordenação a partir de uma Heap (HEAPSORT)

A partir de uma heap, é possível ordenar os dados fazendo trocas:

- O maior elemento(raiz) é trocado com o último elemento;
- **Esse elemento já se encontra ordenado!**

Ordenação a partir de uma Heap (HEAPSORT)

A partir de uma heap, é possível ordenar os dados fazendo trocas:

- O maior elemento(raiz) é trocado com o último elemento;
- Esse elemento já se encontra ordenado!
- **Considerar que o vetor possui agora $(n+1)$ posições e descer a *nova* raiz até sua posição correta na Heap;**

Ordenação a partir de uma Heap (HEAPSORT)

A partir de uma heap, é possível ordenar os dados fazendo trocas:

- O maior elemento(raiz) é trocado com o último elemento;
- Esse elemento já se encontra ordenado!
- Considerar que o vetor possui agora $(n+1)$ posições e descer a *nova* raiz até sua posição correta na Heap;
- **Repetir os passos anteriores $(n+1)$ vezes.**

Ordenação a partir de uma Heap (HEAPSORT)

A partir de uma heap, é possível ordenar os dados fazendo trocas:

- O maior elemento(raiz) é trocado com o último elemento;
- Esse elemento já se encontra ordenado!
- Considerar que o vetor possui agora $(n+1)$ posições e descer a *nova* raiz até sua posição correta na Heap;
- Repetir os passos anteriores $(n+1)$ vezes.

Ordenação a partir de uma Heap (HEAPSORT)

A partir de uma heap, é possível ordenar os dados fazendo trocas:

- O maior elemento(raiz) é trocado com o último elemento;
- Esse elemento já se encontra ordenado!
- Considerar que o vetor possui agora $(n+1)$ posições e descer a *nova* raiz até sua posição correta na Heap;
- Repetir os passos anteriores $(n+1)$ vezes.

Complexidade dessa ordenação é de $O(n\ log\ n)$

Resumo das complexidade da Heap

Operação	Lista	Lista Ordenada	Árvore balanceada	Heap
Seleção[2]	$O(n)$	$O(1)$	$O(log(n))$	$O(1)$
Inserção	$O(1)$	$O(n)$	$O(log(n))$	$O(log(n))$
Remoção[3]	$O(n)$	$O(1)$	$O(log(n))$	$O(log(n))$
Construção[4]	$O(n)$	$O(n\ log(n))$	$O(log(n))$	$O(n)$

[2] Sempre se refere a seleção do elemento com maior prioridade
[3] Considerando que se pretende remover um elemento arbitrário
[4] Construção com ordenação

Exercícios

1. Verifique se as sequências correspondem a uma HEAP:

$$33\ 32\ 28\ 31\ 26\ 29\ 25\ 30\ 27$$

$$33\ 32\ 28\ 31\ 29\ 26\ 25\ 30\ 27$$

2. Realize a inserção dos elementos $L = \{63, 55, 59\}$ na Heap abaixo:

$$H = \{66\ 62\ 56\ 60\ 58\ 52\ 50\ 54\}$$

3. Do resultado da Heap anterior, remova os três elementos com maior prioridade.
4. Seja uma lista dada pelas prioridades a seguir:

$$L = \{18\ 25\ 41\ 34\ 14\ 10\ 52\ 50\ 48\}$$

Explique passo a passo o procedimento de contrução para uma Heap-Max e uma Heap-Min.

Exercícios

5. Realize o processo de ordenação da Heap-Min da questão anterior pelo algoritom Heapsort. O resultado será uma ordenação crescente ou decrescente?

6. Seja H um Heap-max, como podemos desenvolver um algoritmo para decidir qual é o segundo maior elemento (sem precisar ordenar).

7. Como generalizar a questão anterior para descobrir o n-ésimo maior elemento?

8. Todo vetor decrescente é uma heap-max?

9. Toda Heap-max é um vetor decrescente?

10. Suponha que $H[1 \ldots m]$ é um heap e p é um índice menor que $m/2$. É verdade que $H[2p] \geq H[2p+1]$? É verdade que $H[2p] \leq H[2p+1]$?