Computación y Estructuras Discretas I

Andrés A. Aristizábal P.
aaaristizabal@icesi.edu.co

Departamento de Computación y Sistemas Inteligentes

Universidad ICESI

2024-2

Agenda del día

1. **Heapsort**
   - Montículos
   - Manteniendo la propiedad de montículo
   - Construyendo un montículo
   - Algoritmo HEAPSORT
   - Introducción
   - Operaciones y aplicaciones
   - Implementación de las operaciones
   - Ejercicios

¿Qué es un montículo o heap?

Introducción

¿Qué es un montículo o heap?

- Es un arreglo que puede verse como un árbol binario casi completo.
- Cada nodo del árbol corresponde a un elemento del arreglo.
- El árbol se encuentra lleno en casi todos sus niveles
  - Con la excepción de posiblemente el nivel más bajo
  - Este se encuentra lleno desde la izquierda hasta cierto punto.

Montículos

¿Qué nos dice esta última propiedad sobre la forma de un mónticulo?

Montículos

¿Qué nos dice esta última propiedad sobre la forma de un mónticulo?

- El árbol está lleno en casi todos sus niveles con excepción de posiblemente el último.
  - La longitud de toda rama es h o $h - 1$, donde h es la altura del árbol.
- El árbol está lleno en casi todos sus niveles con excepción de posiblemente el último, que se encuentra lleno desde la izquierda hasta cierto punto.
  - No puede existir una rama de longitud h a la derecha de una rama de longitud $h - 1$.

Montículos

¿Cuál es la altura de un nodo del montículo?

Montículos

¿Cuál es la altura de un nodo del montículo?

El número de aristas en el camino simple más largo desde el nodo hasta la hoja.

Montículos

¿Cuál es la altura de un nodo del montículo?

El número de aristas en el camino simple más largo desde el nodo hasta la hoja.

¿Cuál es la altura del montículo?

Montículos

¿Cuál es la altura de un nodo del montículo?

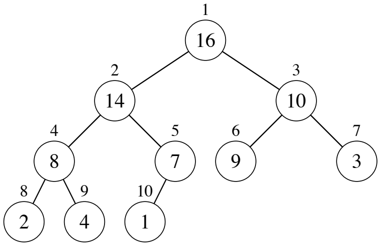
El número de aristas en el camino simple más largo desde el nodo hasta la hoja.

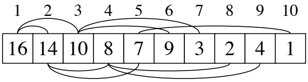
¿Cuál es la altura del montículo?

Al estar basado en un árbol binario completo, es $\Theta(\mathit{lg}\ n)$

¿Cuál sería un ejemplo de montículo?

¿Cuál sería un ejemplo de montículo?





Montículos

¿Cómo es ese arreglo A que representa al montículo?

Montículos

¿Cómo es ese arreglo A que representa al montículo?

- Es un arreglo con dos atributos
  - $A.length$, que es el número de elementos del arreglo
  - $A.heap\_size$, que es el número de elementos del montículo que se encuentran en el arreglo.
  - $A[1..A.length]$ puede contener números
  - Solamente los elementos $A[1..A.heap\_size]$, donde $0 \leq A.heap\_size \leq A.length$, son válidos.

Montículos

¿Qué más se puede decir de ese arreglo A?

Montículos

¿Qué más se puede decir de ese arreglo A?

- La raíz del árbol es $A[1]$
- El padre de $A[i] = A[\lfloor i/2 \rfloor]$
- El hijo izquierdo de $A[i]$ es $A[2i]$
- El hijo derecho de $A[i]$ es $A[2i+1]$
- El cómputo de las dos operaciones anteriores es rápido con una implementación de su representación binaria.

Montículos

¿Cuáles son los dos tipos de montículos?

Montículos

¿Cuáles son los dos tipos de montículos?

- max-heap
  - Para todos los nodos i, excluyendo la raíz, $A[Padre(i)] \geq A[i]$.

Montículos

¿Cuáles son los dos tipos de montículos?

- max-heap
  - Para todos los nodos i, excluyendo la raíz, $A[Padre(i)] \geq A[i]$.
- min-heap

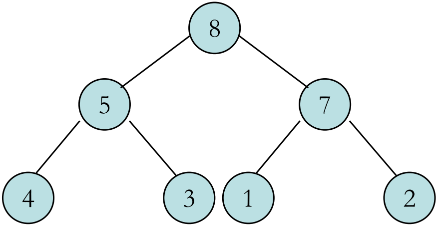
Montículos

¿Cuáles son los dos tipos de montículos?

- max-heap
  - Para todos los nodos i, excluyendo la raíz, $A[Padre(i)] \geq A[i]$.
- min-heap
  - Para todos los nodos i, excluyendo la raíz, $A[Padre(i)] \leq A[i]$.

Ejemplo

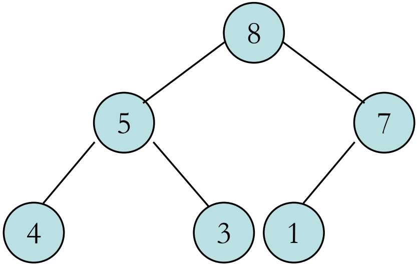
Analicemos las propiedades de orden y forma de este montículo



Montículos

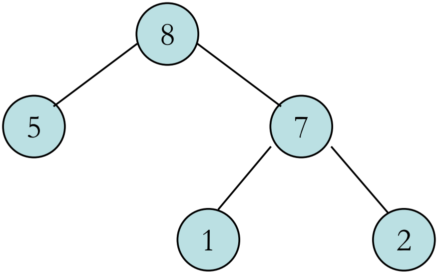
Ejemplo

Analicemos las propiedades de orden y forma de este montículo



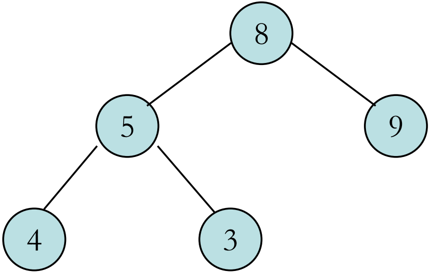
Ejemplo

Analicemos las propiedades de orden y forma de este montículo



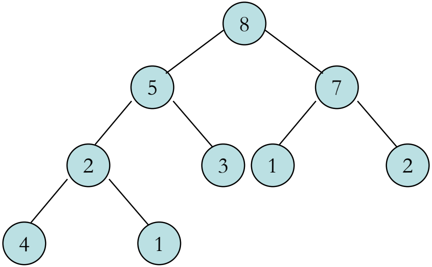
Ejemplo

Analicemos las propiedades de orden y forma de este montículo



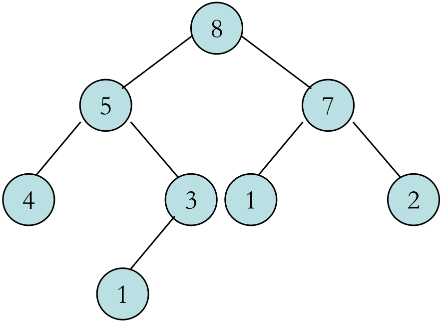
Ejemplo

Analicemos las propiedades de orden y forma de este montículo



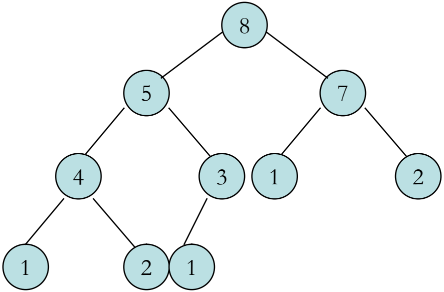
Ejercicio

Indique si se cumplen las propiedades de orden y de forma para este montículo



Ejercicio

Indique si se cumplen las propiedades de orden y de forma para este montículo



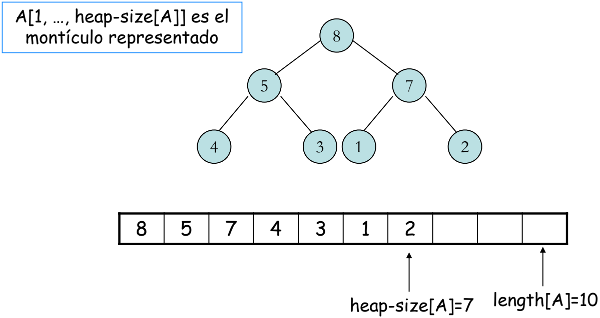
Montículos

¿Cómo se almacenan los elementos de un montículo en el arreglo?

Montículos

¿Cómo se almacenan los elementos de un montículo en el arreglo?

Los datos se almacenan en el arreglo recorriendo, por niveles, de izquierda a derecha.



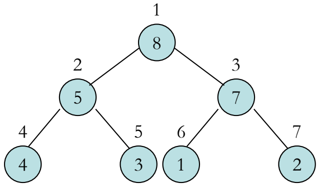
Montículos

Ejercicio

Indique si se cumplen las propiedades de orden y de forma para

1. $A = \{20, 10, 5, 4, 3, 1\}$ *donde heap-size[A]=6 y length[A]=6*
2. $A = \{8, 4, 2, 1, 7, 9\}$ *donde heap-size[A]=4 y length[A]=6*

Ejercicio



1	2	3	4	5	6	7	8	9	10
8	5	7	4	3	1	2			

Evalue $\lfloor i/2 \rfloor$ para i=2 y 3

Evalue $\lfloor i/2 \rfloor$ para i=4 y 5

Evalue $\lfloor i/2 \rfloor$ para i=6 y 7

Padre(i): $\lfloor i/2 \rfloor$

Montículos

¿Cuáles son las operaciones más importantes que se realizan con montículos?

Montículos

¿Cuáles son las operaciones más importantes que se realizan con montículos?

- HEAPIFY: $O(lg\ n)$
- BUILD-HEAP: $O(n)$
- HEAPSORT: $O(n\ lg\ n)$

Agenda del día

1. **Heapsort**
   - Montículos
   - Manteniendo la propiedad de montículo
   - Construyendo un montículo
   - Algoritmo HEAPSORT
   - Introducción
   - Operaciones y aplicaciones
   - Implementación de las operaciones
   - Ejercicios

Manteniendo la propiedad de montículo

¿Qué hace la operación HEAPIFY?

Manteniendo la propiedad de montículo

¿Qué hace la operación HEAPIFY?

- Esta es importante para manipular montículos
- Se usa para garantizar la propiedad de orden del montículo.

Manteniendo la propiedad de montículo

¿Qué hace la operación MAX-HEAPIFY?

Manteniendo la propiedad de montículo

¿Qué hace la operación MAX-HEAPIFY?

- Se usa para garantizar la propiedad de orden del max-heap.
- Antes de aplicar el MAX-HEAPIFY, A[i] puede ser menor que sus hijos.
- Se asume que los subárboles izquierdos y derechos son max-heaps.
- Luego de MAX-HEAPIFY, el subárbol con raíz i es un max-heap.

Manteniendo la propiedad de montículo

¿Cuál es el algoritmo para el MAX-HEAPIFY?

Manteniendo la propiedad de montículo

¿Cuál es el algoritmo para el MAX-HEAPIFY?

```
MAX-HEAPIFY(A, i)
l = LEFT(i)
r = RIGHT(i)
if l ≤ A.heap-size and A[l] > A[i]
    largest = l
else largest = i
if r ≤ A.heap-size and A[r] > A[largest]
    largest = r
if largest ≠ i
    exchange A[i] with A[largest]
    MAX-HEAPIFY(A, largest)
```

Manteniendo la propiedad de montículo

¿Cómo funciona el MAX-HEAPIFY?

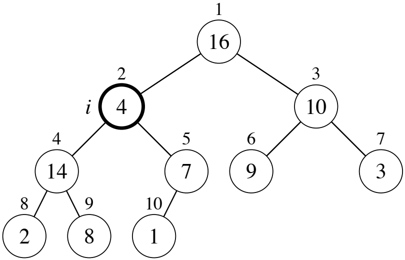
Manteniendo la propiedad de montículo

¿Cómo funciona el MAX-HEAPIFY?

- Compara $A[i]$, $A[LEFT(i)]$ y $A[RIGHT(i)]$
- Si es necesario intercambia $A[i]$ con el mayor de sus hijos para preservar la propiedad del max-heap.
- Continua con el proceso comparando y cambiando, bajando sobre el montículo, hasta que el subárbol con la raíz i sea un max-heap.
  - Si se llega a una hoja el subárbol con la hoja como raíz es un max-heap de manera trivial.

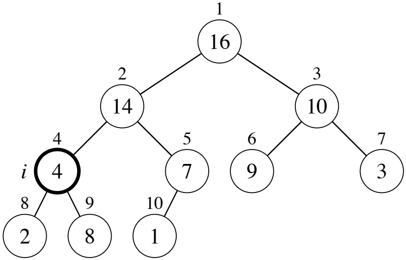
Manteniendo la propiedad de montículo

¿Cómo funciona el MAX-HEAPIFY sobre este ejemplo?



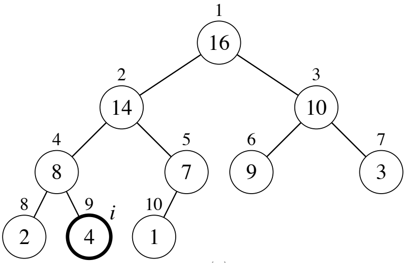
Manteniendo la propiedad de montículo

¿Cómo funciona el MAX-HEAPIFY sobre este ejemplo?



Manteniendo la propiedad de montículo

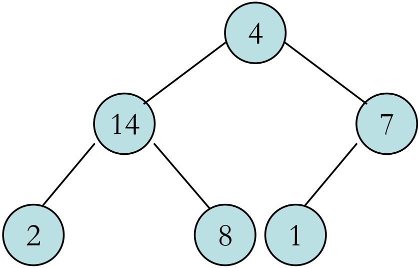
¿Cómo funciona el MAX-HEAPIFY sobre este ejemplo?



Manteniendo la propiedad de montículo

Ejercicio

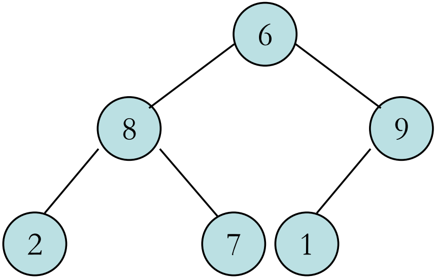
Aplique el algoritmo MAX-HEAPIFY(A, 1)



Manteniendo la propiedad de montículo

Ejercicio

Aplique el algoritmo MAX-HEAPIFY(A, 1)



Agenda del día

1. **Heapsort**
   - Montículos
   - Manteniendo la propiedad de montículo
   - Construyendo un montículo
   - Algoritmo HEAPSORT
   - Introducción
   - Operaciones y aplicaciones
   - Implementación de las operaciones
   - Ejercicios

Construyendo un montículo

¿Qué hace la operación BUILD-MAX-HEAP?

¿Qué hace la operación BUILD-MAX-HEAP?

- Haciendo uso del MAX-HEAPIFY convierte un arreglo $A[1..n]$ donde $n = A.length$ en un max-heap.

Construyendo un montículo

¿Cuál es el algoritmo para el BUILD-MAX-HEAP?

Construyendo un montículo

¿Cuál es el algoritmo para el BUILD-MAX-HEAP?

BUILD-MAX-HEAP(A)

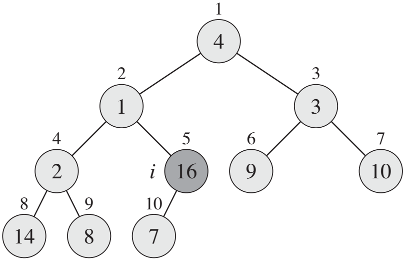
$A.\mathit{heap\text{-}size} = A.\mathit{length}$

for $i = \lfloor A.\mathit{length}/2 \rfloor$ **downto** 1

    MAX-HEAPIFY(A, i)

Construyendo un montículo

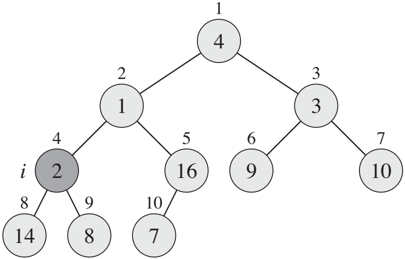
¿Cómo funciona el BUILD-MAX-HEAP sobre este ejemplo?



(a)

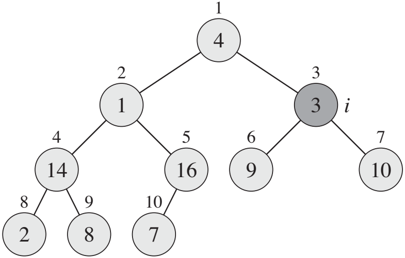
Construyendo un montículo

¿Cómo funciona el BUILD-MAX-HEAP sobre este ejemplo?



(b)

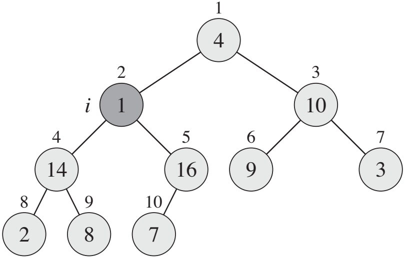
¿Cómo funciona el BUILD-MAX-HEAP sobre este ejemplo?



(c)

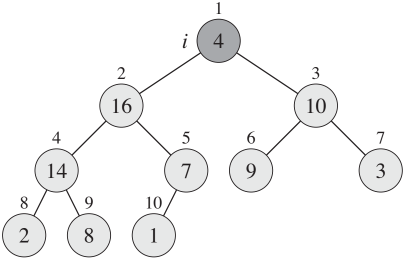
Construyendo un montículo

¿Cómo funciona el BUILD-MAX-HEAP sobre este ejemplo?



(d)

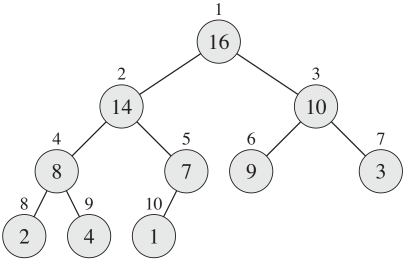
¿Cómo funciona el BUILD-MAX-HEAP sobre este ejemplo?



(e)

Construyendo un montículo

¿Cómo funciona el BUILD-MAX-HEAP sobre este ejemplo?



(f)

Construyendo un montículo

Ejercicio

Aplique el algoritmo BUILD-HEAP(A), para $A = \{5, 7, 10, 1, 4, 6, 8, 2, 9, 12\}$ *y heap-size(A)=10*

Agenda del día

1. **Heapsort**
   - Montículos
   - Manteniendo la propiedad de montículo
   - Construyendo un montículo
   - Algoritmo HEAPSORT
   - Introducción
   - Operaciones y aplicaciones
   - Implementación de las operaciones
   - Ejercicios

Algoritmo HEAPSORT

¿Qué hace el HEAPSORT?

Algoritmo HEAPSORT

¿Qué hace el HEAPSORT?

- Comienza utilizando el BUILD-MAX-HEAP para construir un max-heap a partir de un arreglo de entrada A, donde $n = A.length$
- Como el máximo elemento del montículo se guarda en la raíz $A[1]$, lo ponemos en su posición adecuada $A[n]$ intercambiando valores.
- Descartamos el nodo A[n] del montículo decrementando A.heap-size.
- Como la nueva raíz puede incumplir la propiedad del max-heap llamamos a MAX-HEAPIFY(A,1).
- Este deja el arreglo A[1..n-1] como un max-heap.
- El HEAPSORT repite este algoritmo desde el max-heap de $n - 1$ hasta 2.

Algoritmo HEAPSORT

¿Cuál es el algoritmo para el HEAPSORT?

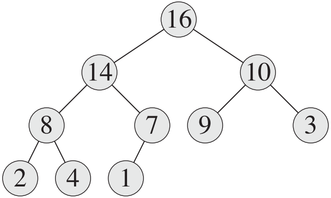
Algoritmo HEAPSORT

¿Cuál es el algoritmo para el HEAPSORT?

HEAPSORT(A)

BUILD-MAX-HEAP(A)
for $i = A.length$ **downto** 2
    exchange $A[1]$ with $A[i]$
    $A.\mathit{heap\text{-}size} = A.\mathit{heap\text{-}size} - 1$
    MAX-HEAPIFY$(A, 1)$

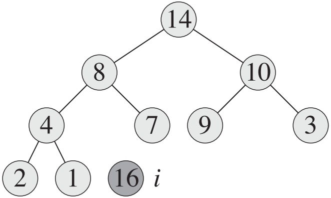
¿Cómo funciona el HEAPSORT sobre este ejemplo?



(a)

Algoritmo HEAPSORT

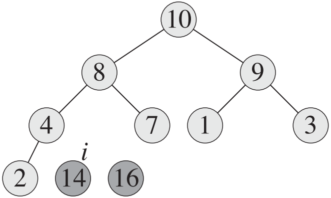
¿Cómo funciona el HEAPSORT sobre este ejemplo?



(b)

Algoritmo HEAPSORT

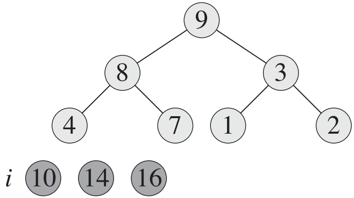
¿Cómo funciona el HEAPSORT sobre este ejemplo?



(c)

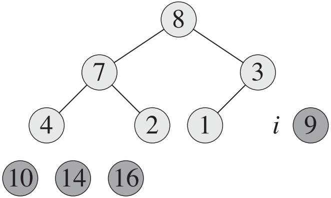
Algoritmo HEAPSORT

¿Cómo funciona el HEAPSORT sobre este ejemplo?



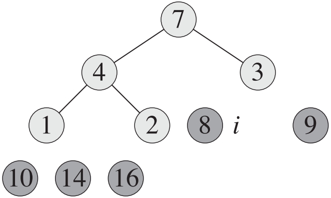
(d)

¿Cómo funciona el HEAPSORT sobre este ejemplo?



(e)

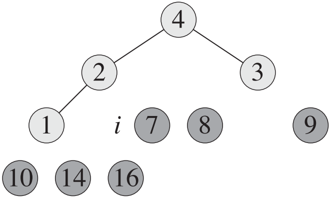
¿Cómo funciona el HEAPSORT sobre este ejemplo?



(f)

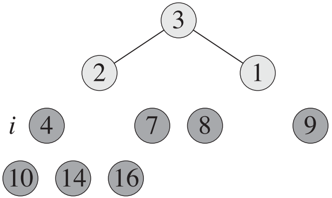
Algoritmo HEAPSORT

¿Cómo funciona el HEAPSORT sobre este ejemplo?



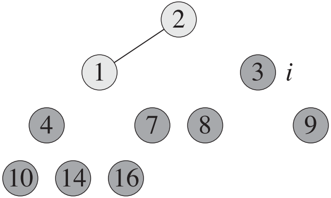
(g)

¿Cómo funciona el HEAPSORT sobre este ejemplo?



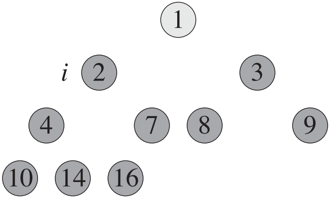
(h)

¿Cómo funciona el HEAPSORT sobre este ejemplo?



(i)

¿Cómo funciona el HEAPSORT sobre este ejemplo?



(j)

¿Cómo funciona el HEAPSORT sobre este ejemplo?

A	1	2	3	4	7	8	9	10	14	16

(k)

Algoritmo HEAPSORT

Ejercicio

Aplique el HEAPSORT(A), para $A = \{12, 9, 10, 7, 8, 1\}$ *y heap-size(A)=6*

Ejercicio

Aplique el HEAPSORT(A), para $A = \{5, 7, 10, 1, 4, 6, 8, 2, 9, 12\}$ *y heap-size(A)=10*

Ejercicio

Construya la estructura montículo genérica y posteriormente utilice dicha estructura para implementar el heapsort.

Agenda del día

1. **Heapsort**
   - Montículos
   - Manteniendo la propiedad de montículo
   - Construyendo un montículo
   - Algoritmo HEAPSORT
   - Introducción
   - Operaciones y aplicaciones
   - Implementación de las operaciones
   - Ejercicios

Introducción

¿Qué es una cola de prioridad?

Introducción

¿Qué es una cola de prioridad?

- Es una estructura de datos utilizada para mantener un conjunto de elementos S, cada uno con una clave como valor asociado.

Introducción

¿Qué es una cola de prioridad?

- Es una estructura de datos utilizada para mantener un conjunto de elementos S, cada uno con una clave como valor asociado.
- Similar a una cola, pero los elementos tienen adicionalmente, una prioridad asignada (clave).

Introducción

¿Qué es una cola de prioridad?

- Es una estructura de datos utilizada para mantener un conjunto de elementos S, cada uno con una clave como valor asociado.
- Similar a una cola, pero los elementos tienen adicionalmente, una prioridad asignada (clave).
- En una cola de prioridades un elemento con mayor prioridad será desencolado antes que un elemento de menor prioridad.

Introducción

¿Qué es una cola de prioridad?

- Es una estructura de datos utilizada para mantener un conjunto de elementos S, cada uno con una clave como valor asociado.
- Similar a una cola, pero los elementos tienen adicionalmente, una prioridad asignada (clave).
- En una cola de prioridades un elemento con mayor prioridad será desencolado antes que un elemento de menor prioridad.
- Si dos elementos tienen la misma prioridad, se desencolarán siguiendo el orden de cola.

Introducción

¿Qué es una cola de prioridad?

- Es una estructura de datos utilizada para mantener un conjunto de elementos S, cada uno con una clave como valor asociado.
- Similar a una cola, pero los elementos tienen adicionalmente, una prioridad asignada (clave).
- En una cola de prioridades un elemento con mayor prioridad será desencolado antes que un elemento de menor prioridad.
- Si dos elementos tienen la misma prioridad, se desencolarán siguiendo el orden de cola.
- Al igual que con los montículos existen dos tipos de colas de prioridad:

Introducción

¿Qué es una cola de prioridad?

- Es una estructura de datos utilizada para mantener un conjunto de elementos S, cada uno con una clave como valor asociado.
- Similar a una cola, pero los elementos tienen adicionalmente, una prioridad asignada (clave).
- En una cola de prioridades un elemento con mayor prioridad será desencolado antes que un elemento de menor prioridad.
- Si dos elementos tienen la misma prioridad, se desencolarán siguiendo el orden de cola.
- Al igual que con los montículos existen dos tipos de colas de prioridad:
    - Las colas de prioridad máximas (max-priority queues) basadas en max-heaps

Introducción

¿Qué es una cola de prioridad?

- Es una estructura de datos utilizada para mantener un conjunto de elementos S, cada uno con una clave como valor asociado.
- Similar a una cola, pero los elementos tienen adicionalmente, una prioridad asignada (clave).
- En una cola de prioridades un elemento con mayor prioridad será desencolado antes que un elemento de menor prioridad.
- Si dos elementos tienen la misma prioridad, se desencolarán siguiendo el orden de cola.
- Al igual que con los montículos existen dos tipos de colas de prioridad:
  - Las colas de prioridad máximas (max-priority queues) basadas en max-heaps
  - Las colas de prioridad mínimas (min-priority queues) basadas en min-heaps

Agenda del día

1. **Heapsort**
   - Montículos
   - Manteniendo la propiedad de montículo
   - Construyendo un montículo
   - Algoritmo HEAPSORT
   - Introducción
   - Operaciones y aplicaciones
   - Implementación de las operaciones
   - Ejercicios

Operaciones y aplicaciones

¿Qué operaciones soportan las max-priority queues?

Operaciones y aplicaciones

¿Qué operaciones soportan las max-priority queues?

- `Insert(S,x)` inserta el elemento en el conjunto S, lo cual es equivalente a la operación $S \cup \{x\}$.

Operaciones y aplicaciones

¿Qué operaciones soportan las max-priority queues?

- `Insert(S,x)` inserta el elemento en el conjunto S, lo cual es equivalente a la operación $S \cup \{x\}$.
- `Maximum(S)` retorna el elemento de S con la mayor clave.

Operaciones y aplicaciones

¿Qué operaciones soportan las max-priority queues?

- `Insert(S,x)` inserta el elemento en el conjunto S, lo cual es equivalente a la operación $S \cup \{x\}$.
- `Maximum(S)` retorna el elemento de S con la mayor clave.
- `Extract-Max(S)` elimina y retorna el elemento de S con la mayor clave.

Operaciones y aplicaciones

¿Qué operaciones soportan las max-priority queues?

- `Insert(S,x)` inserta el elemento en el conjunto S, lo cual es equivalente a la operación $S \cup \{x\}$.
- `Maximum(S)` retorna el elemento de S con la mayor clave.
- `Extract-Max(S)` elimina y retorna el elemento de S con la mayor clave.
- `Increase-Key(S,x,K)` incrementa el valor de la clave del elemento x a un nuevo valor K, que se asume debe ser tan o más grande que el valor actual de la clave de x.

¿Cuáles serían algunas de las aplicaciones de estas colas de prioridad máximas?

Operaciones y aplicaciones

¿Cuáles serían algunas de las aplicaciones de estas colas de prioridad máximas?

- Programar tareas en un computador compartido.

Operaciones y aplicaciones

¿Cuáles serían algunas de las aplicaciones de estas colas de prioridad máximas?

- Programar tareas en un computador compartido.
  - La cola de prioridad mantendrá un registro de dichas tareas y sus prioridades

Operaciones y aplicaciones

¿Cuáles serían algunas de las aplicaciones de estas colas de prioridad máximas?

- Programar tareas en un computador compartido.
  - La cola de prioridad mantendrá un registro de dichas tareas y sus prioridades
  - Cuando una tarea finaliza o se interrumpe, el scheduler selecciona a través del `Extract-Max` la tarea con mayor prioridad de aquellas que se encuentran pendientes por ejecutarse.

Operaciones y aplicaciones

¿Cuáles serían algunas de las aplicaciones de estas colas de prioridad máximas?

- Programar tareas en un computador compartido.
  - La cola de prioridad mantendrá un registro de dichas tareas y sus prioridades
  - Cuando una tarea finaliza o se interrumpe, el scheduler selecciona a través del `Extract-Max` la tarea con mayor prioridad de aquellas que se encuentran pendientes por ejecutarse.
  - El scheduler puede añadir otra tarea mediante el `Insert`

Operaciones y aplicaciones

¿Qué operaciones soportan las min-priority queues?

Operaciones y aplicaciones

¿Qué operaciones soportan las min-priority queues?

- `Insert(S,x)` inserta el elemento en el conjunto S

Operaciones y aplicaciones

¿Qué operaciones soportan las min-priority queues?

- `Insert(S,x)` inserta el elemento en el conjunto S
- `Minimum(S)` retorna el elemento de S con la menor clave.

Operaciones y aplicaciones

¿Qué operaciones soportan las min-priority queues?

- `Insert(S,x)` inserta el elemento en el conjunto S
- `Minimum(S)` retorna el elemento de S con la menor clave.
- `Extract-Min(S)` elimina y retorna el elemento de S con la menor clave.

Operaciones y aplicaciones

¿Qué operaciones soportan las min-priority queues?

- `Insert(S,x)` inserta el elemento en el conjunto S
- `Minimum(S)` retorna el elemento de S con la menor clave.
- `Extract-Min(S)` elimina y retorna el elemento de S con la menor clave.
- `Decrease-Key(S,x,K)` decrementa el valor de la clave del elemento x a un nuevo valor K, que se asume debe ser igual o menor que el valor actual de la clave de x.

Operaciones y aplicaciones

¿Cuáles serían algunas de las aplicaciones de estas colas de prioridad mínimas?

Operaciones y aplicaciones

¿Cuáles serían algunas de las aplicaciones de estas colas de prioridad mínimas?

- Simulador de eventos.

Operaciones y aplicaciones

¿Cuáles serían algunas de las aplicaciones de estas colas de prioridad mínimas?

- Simulador de eventos.
  - Los elementos en la cola son eventos a ser simulados, cada uno con una clave que representa el tiempo en el que deben simularse u ocurrir

Operaciones y aplicaciones

¿Cuáles serían algunas de las aplicaciones de estas colas de prioridad mínimas?

- Simulador de eventos.
  - Los elementos en la cola son eventos a ser simulados, cada uno con una clave que representa el tiempo en el que deben simularse u ocurrir
  - Los eventos deben simularse en el orden de su tiempo de ocurrencia, ya que la simulación de un evento puede causar que otros eventos sean simulados en el futuro.

Operaciones y aplicaciones

¿Cuáles serían algunas de las aplicaciones de estas colas de prioridad mínimas?

- Simulador de eventos.
  - Los elementos en la cola son eventos a ser simulados, cada uno con una clave que representa el tiempo en el que deben simularse u ocurrir
  - Los eventos deben simularse en el orden de su tiempo de ocurrencia, ya que la simulación de un evento puede causar que otros eventos sean simulados en el futuro.
  - El programa de simulación llama a `Extract-Min` a cada paso, de tal manera que se elija el siguiente evento a simularse.

Operaciones y aplicaciones

¿Cuáles serían algunas de las aplicaciones de estas colas de prioridad mínimas?

- Simulador de eventos.
  - Los elementos en la cola son eventos a ser simulados, cada uno con una clave que representa el tiempo en el que deben simularse u ocurrir
  - Los eventos deben simularse en el orden de su tiempo de ocurrencia, ya que la simulación de un evento puede causar que otros eventos sean simulados en el futuro.
  - El programa de simulación llama a `Extract-Min` a cada paso, de tal manera que se elija el siguiente evento a simularse.
  - A medida que se producen nuevos eventos se van insertando en la cola de prioridad a través del `Insert`.

Operaciones y aplicaciones

¿Cómo asociar elementos de aplicaciones como programadores de tareas o simuladores de eventos con colas de prioridad?

Operaciones y aplicaciones

¿Cómo asociar elementos de aplicaciones como programadores de tareas o simuladores de eventos con colas de prioridad?

- Debemos determinar qué elementos de la cola de prioridad corresponden a los objetos de la aplicación.

Operaciones y aplicaciones

¿Cómo asociar elementos de aplicaciones como programadores de tareas o simuladores de eventos con colas de prioridad?

- Debemos determinar qué elementos de la cola de prioridad corresponden a los objetos de la aplicación.
- Cuando se utilizan montículos para implementar las colas de prioridad necesitamos guardar un handle al objeto correspondiente en la aplicación, en cada elemento del montículo.

Operaciones y aplicaciones

¿Cómo asociar elementos de aplicaciones como programadores de tareas o simuladores de eventos con colas de prioridad?

- Debemos determinar qué elementos de la cola de prioridad corresponden a los objetos de la aplicación.
- Cuando se utilizan montículos para implementar las colas de prioridad necesitamos guardar un handle al objeto correspondiente en la aplicación, en cada elemento del montículo.
- Este depende de la aplicación, ya sea un puntero o un entero.

Operaciones y aplicaciones

¿Cómo asociar elementos de aplicaciones como programadores de tareas o simuladores de eventos con colas de prioridad?

- Debemos determinar qué elementos de la cola de prioridad corresponden a los objetos de la aplicación.
- Cuando se utilizan montículos para implementar las colas de prioridad necesitamos guardar un handle al objeto correspondiente en la aplicación, en cada elemento del montículo.
- Este depende de la aplicación, ya sea un puntero o un entero.
- Asimismo debemos guardar en el objeto de la aplicación un handle al elemento del montículo. En este caso, típicamente sería un índice del arreglo.

Operaciones y aplicaciones

¿Cómo asociar elementos de aplicaciones como programadores de tareas o simuladores de eventos con colas de prioridad?

- Debemos determinar qué elementos de la cola de prioridad corresponden a los objetos de la aplicación.
- Cuando se utilizan montículos para implementar las colas de prioridad necesitamos guardar un handle al objeto correspondiente en la aplicación, en cada elemento del montículo.
- Este depende de la aplicación, ya sea un puntero o un entero.
- Asimismo debemos guardar en el objeto de la aplicación un handle al elemento del montículo. En este caso, típicamente sería un índice del arreglo.
- Como los elementos de los montículos cambian de posiciones durante las operaciones, en una implementación si se reubica el elemento del montículo, se debe actualizar el índice del arreglo en el objeto de la aplicación.

Implementación de las operaciones

¿Cómo es el `HEAP-MAXIMUM(A)`?

¿Cómo es el `HEAP-MAXIMUM(A)`?

```
HEAP-MAXIMUM(A)
1  return A[1]
```

Implementación de las operaciones

¿Cómo es el `HEAP-MAXIMUM(A)`?

HEAP-MAXIMUM(A)
1 **return** $A[1]$

¿Y la complejidad?

Implementación de las operaciones

¿Cómo es el `HEAP-MAXIMUM(A)`?

HEAP-MAXIMUM(A)

1 **return** $A[1]$

¿Y la complejidad?

$\Theta(1)$

¿Cómo es el `HEAP-EXTRACT-MAX(A)`?

Implementación de las operaciones

¿Cómo es el `HEAP-EXTRACT-MAX(A)`?

HEAP-EXTRACT-MAX(A)

if $A.\mathit{heap\text{-}size} < 1$
    error "heap underflow"
$\mathit{max} = A[1]$
$A[1] = A[A.\mathit{heap\text{-}size}]$
$A.\mathit{heap\text{-}size} = A.\mathit{heap\text{-}size} - 1$
MAX-HEAPIFY$(A, 1)$
return *max*

Implementación de las operaciones

¿Cómo es el `HEAP-EXTRACT-MAX(A)`?

```
HEAP-EXTRACT-MAX(A)
if A.heap-size < 1
    error "heap underflow"
max = A[1]
A[1] = A[A.heap-size]
A.heap-size = A.heap-size - 1
MAX-HEAPIFY(A, 1)
return max
```

¿Y la complejidad?

Implementación de las operaciones

¿Cómo es el `HEAP-EXTRACT-MAX(A)`?

```
HEAP-EXTRACT-MAX(A)
if A.heap-size < 1
    error "heap underflow"
max = A[1]
A[1] = A[A.heap-size]
A.heap-size = A.heap-size - 1
MAX-HEAPIFY(A, 1)
return max
```

¿Y la complejidad?

$O(lg\ n)$

¿Cómo es el `HEAP-INCREASE-KEY(A,i,key)`?

Implementación de las operaciones

¿Cómo es el `HEAP-INCREASE-KEY(A,i,key)`?

```
HEAP-INCREASE-KEY(A, i, key)
if key < A[i]
    error "new key is smaller than current key"
A[i] = key
while i > 1 and A[PARENT(i)] < A[i]
    exchange A[i] with A[PARENT(i)]
    i = PARENT(i)
```

Implementación de las operaciones

¿Cómo es el `HEAP-INCREASE-KEY(A,i,key)`?

```
HEAP-INCREASE-KEY(A, i, key)
if key < A[i]
    error "new key is smaller than current key"
A[i] = key
while i > 1 and A[PARENT(i)] < A[i]
    exchange A[i] with A[PARENT(i)]
    i = PARENT(i)
```

¿Y la complejidad?

Implementación de las operaciones

¿Cómo es el `HEAP-INCREASE-KEY(A,i,key)`?

```
HEAP-INCREASE-KEY(A, i, key)
if key < A[i]
    error "new key is smaller than current key"
A[i] = key
while i > 1 and A[PARENT(i)] < A[i]
    exchange A[i] with A[PARENT(i)]
    i = PARENT(i)
```

¿Y la complejidad?

O(*lg n*)

Implementación de las operaciones

¿Cómo es el `MAX-HEAP-INSERT(A,key)`?

¿Cómo es el `MAX-HEAP-INSERT(A,key)`?

MAX-HEAP-INSERT(A, key)

$A.heap\text{-}size = A.heap\text{-}size + 1$
$A[A.heap\text{-}size] = -\infty$
HEAP-INCREASE-KEY$(A, A.heap\text{-}size, key)$

Implementación de las operaciones

¿Cómo es el `MAX-HEAP-INSERT(A, key)`?

MAX-HEAP-INSERT(A, key)

$A.heap\text{-}size = A.heap\text{-}size + 1$

$A[A.heap\text{-}size] = -\infty$

HEAP-INCREASE-KEY$(A, A.heap\text{-}size, key)$

¿Y la complejidad?

Implementación de las operaciones

¿Cómo es el `MAX-HEAP-INSERT(A,key)`?

MAX-HEAP-INSERT(A, key)

1 $A.heap\text{-}size = A.heap\text{-}size + 1$
2 $A[A.heap\text{-}size] = -\infty$
3 HEAP-INCREASE-KEY$(A, A.heap\text{-}size, key)$

¿Y la complejidad?

O(*lg n*)

Agenda del día

1. **Heapsort**
   - Montículos
   - Manteniendo la propiedad de montículo
   - Construyendo un montículo
   - Algoritmo HEAPSORT
   - Introducción
   - Operaciones y aplicaciones
   - Implementación de las operaciones
   - Ejercicios

Ejercicios

Ejercicio

Ilustre la operación HEAP-EXTRACT-MAX en el montículo $A = \langle 15, 13, 9, 5, 12, 8, 7, 4, 0, 6, 2, 1 \rangle$.

Ejercicio

Ilustre la operación MAX-HEAP-INSERT(A,10) en el montículo $A = \langle 15, 13, 9, 5, 12, 8, 7, 4, 0, 6, 2, 1 \rangle$.

Ejercicios

Ejercicio

Ilustre la operación HEAP-INCREASE-KEY(A,8,15) en el montículo $A = \langle 16, 14, 10, 8, 7, 9, 3, 2, 4, 1 \rangle$.