Repaso I3

Clase 25

IIC 2133 - Sección 2

Prof. Mario Droguett

Sumario

Interrogación 3

Objetivos a evaluar en la I3

- ☐ Comprender heaps y su aplicación a colas de prioridad
- ☐ Identificar dominios de aplicación de algoritmos en grafos
- ☐ Aplicar algoritmos destacados de grafos
- ☐ Modificar algoritmos de grafos

Lo esencial de esta interrogación es decidir qué algoritmo usar y cómo modificarlo

Interrogación 3

Formato de la prueba

- 2 horas de tiempo
- Pool de 4 preguntas para elegir 3
- Cada pregunta incluye un título que describe sus temas
- ¡**SOLO** se entregan 3 preguntas respondidas!

Nota de la I3: promedio de las 3 preguntas entregadas

Interrogación 3

Material adicional

- Pueden usar un formulario/apuntes durante la prueba
- Debe estar escrito a mano (puede ser impreso de tablet)
- Una hoja (por ambos lados)
- Sugerencia: incluyan los pseudocódigos vistos

No se aceptarán diapositivas impresas

les recuperativas

Formato de la prueba

- 1 hora de tiempo
- 2 preguntas
- Cada pregunta incluye un título que describe sus temas
- Nota: promedio de las dos preguntas

Esta nota reemplaza la interrogación a la que se faltó

Objetivos de la clase

- ☐ Recordar elementos esenciales de los contenidos a evaluar
- ☐ Identificar diferencias de contenidos que aplican a un mismo escenario
- ☐ Aplicar algunos de los contenidos a ejemplos concretos
- ☐ Conocer enfoque esperado en las respuestas de algunos ejemplos

Sumario

Heaps binarios

Definición

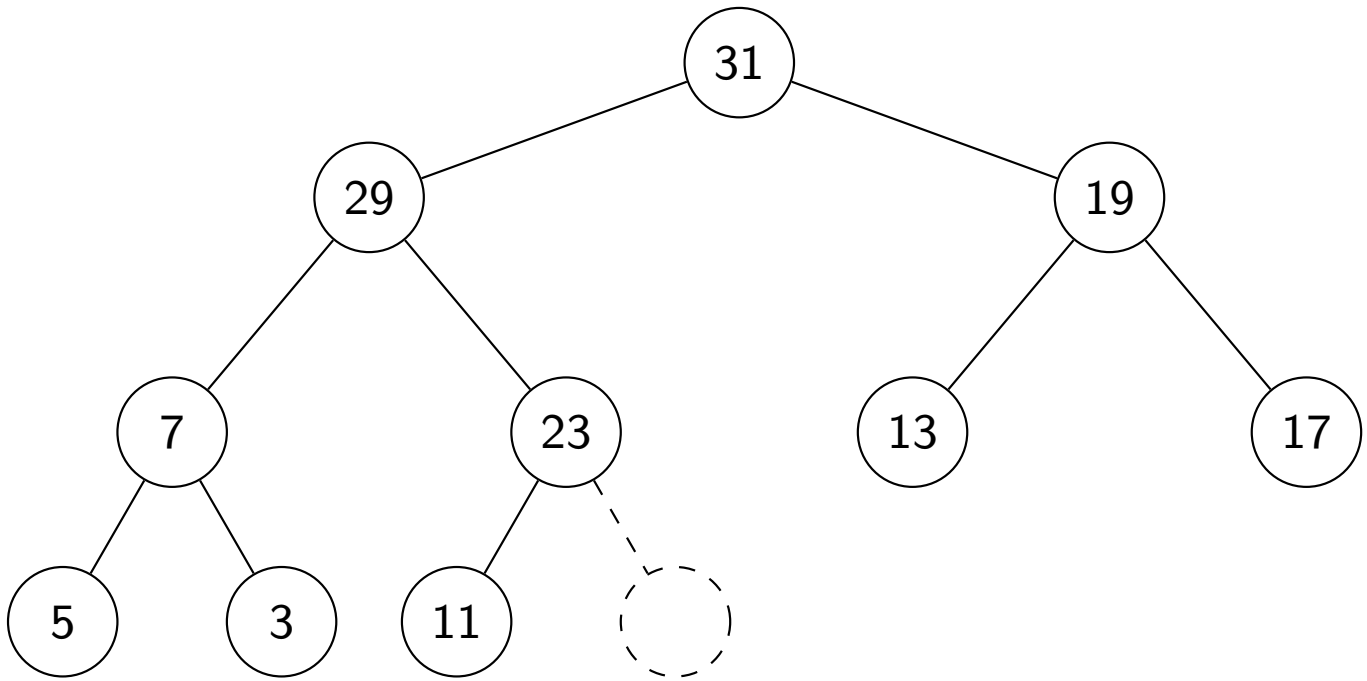
Un **heap binario** H es un árbol binario tal que

- $H.left$ y $H.right$ son heaps binarios
- $H.value > H.left.value$
- $H.value > H.right.value$

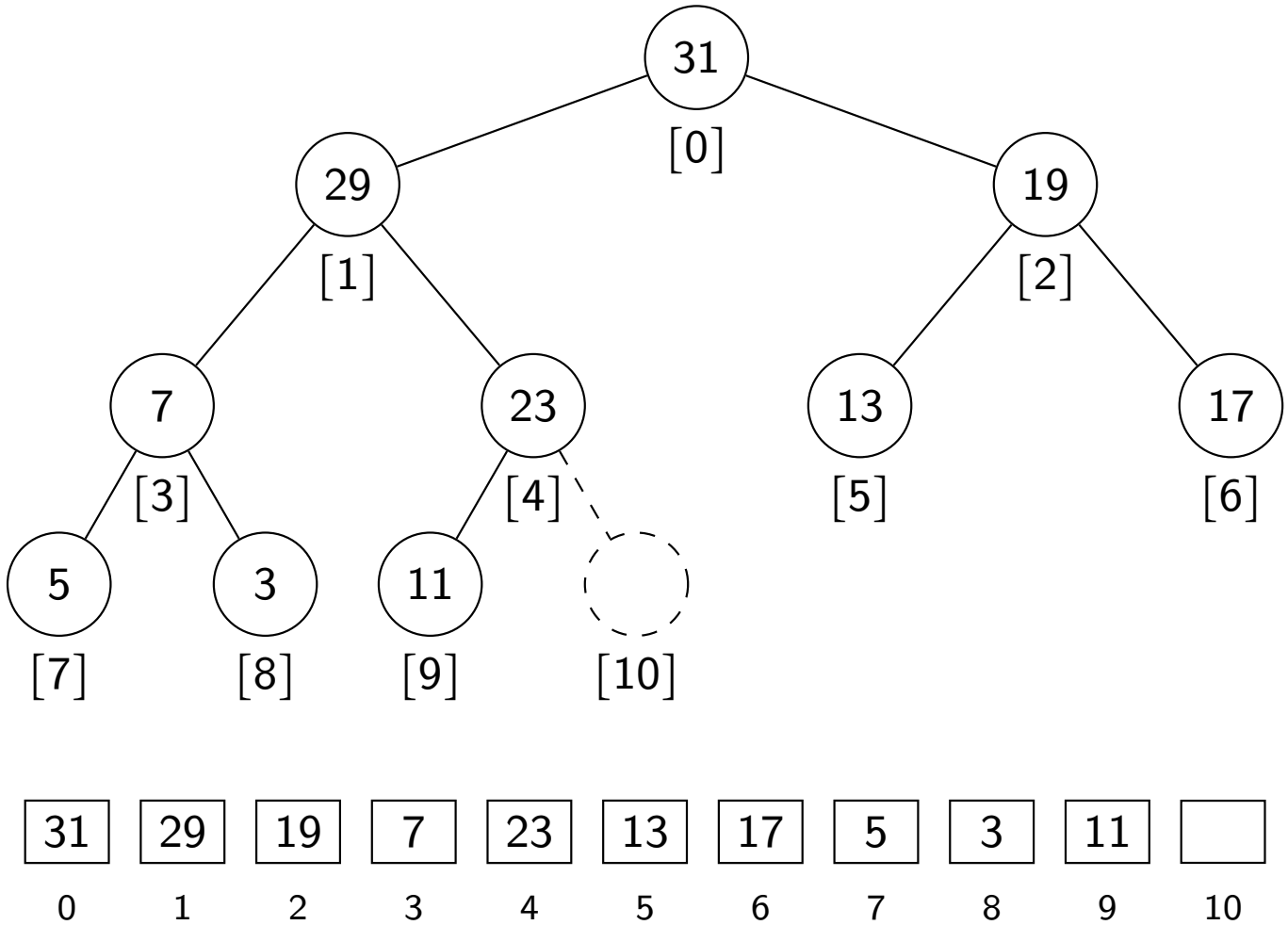
A estas condiciones les llamamos **propiedad de heap**

Esta es la definición de un **MAX heap**

Heaps binarios



La gracia: representación compacta



Heaps binarios

Aspectos esenciales

- Propiedad de heap

$$\text{descendientes} < \text{nodo} < \text{ancestros}$$

- Se asegura **balance** mediante sus operaciones
  - Inserción al final del arreglo, reubicando (`SiftUp`)
  - Extracción de la raíz, reemplazando con el último y reubicando (`SiftDown`)
  - Actualización de prioridad (`SiftUp`)

Operaciones logarítmicas en la cantidad de nodos

Heaps binarios

Operaciones

- Inserción: agregar un nuevo elemento
  - Se inserta al final del arreglo
  - Se reubica con `SiftUp`
- Extracción: sacar y retornar el más prioritario
  - Se saca el primer elemento y se reemplaza con el último
  - Se reubica la nueva raíz con `SiftDown`
- Actualización: aumentarle la prioridad a un nodo
  - Se reubica el nodo con `SiftUp`

Todas estas son operaciones $\mathcal{O}(\log(n))$

Los heaps no se usan para buscar,
sino para informar el elemento más prioritario

Heaps binarios

Orientaciones para el estudio

- ☐ Comprender la representación compacta de heaps
- ☐ Comprender operaciones y su uso para mantener una cola de prioridad
- ☐ Considerar casos borde: ¿qué pasa si la prioridad es el orden de llegada? ¿Qué se puede hacer más rápido y cómo?

Incluyan en el formulario el pseudocódigo de los métodos de heaps

Sumario

Algoritmos en grafos

Cada algoritmo tiene un contexto de aplicación

- ¿Qué problema resuelve?
- ¿Qué queda guardado en sus variables/EDD's?

Además, el grafo estudiado debe tener ciertas características

- Dirección de aristas: dirigido o no dirigido
- Pesos en aristas: sin pesos, pesos no negativos, pesos libres

Es vital conocer los requisitos y contextos de aplicación de cada uno

Algoritmos en grafos

Familias de problemas que estudiamos

1. Recorrido en amplitud
   - 1.1 BFS: Grafos dirigidos (y también no dirigidos)
2. Rutas más baratas
   - 2.1 Dijkstra: costos no negativos
   - 2.2 Bellman-Ford: costos libres
3. Árboles de cobertura mínimos (MST)
   - 3.1 Kruskal y Prim: grafo no dirigido con costos libres

Recordar que en "este mundo", las complejidades dependen de E y V

Algoritmos en grafos

Aspectos de implementación

- BFS es un caso particular de Dijkstra
  - Costos iguales para toda arista
  - No requiere estructuras más poderosas que listas
- Dijkstra y Prim son algoritmos codiciosos
  - Su decisión codiciosa se basa en elegir menor costo actual
  - Esto se mantiene en una **cola de prioridad**
- Kruskal basa su funcionamiento en determinar comunidades
  - Implementación de **conjuntos disjuntos**
  - Permiten saber de forma eficiente si dos elementos están en el mismo subárbol

No olvidar las estructuras que se usan por debajo en cada algoritmo

Algoritmos en grafos

Orientaciones para el estudio

- ☐ Comprender qué problema resuelve cada algoritmo
- ☐ Comprender su funcionamiento/pseudocódigo
- ☐ Aplicarlos a situaciones diferentes a los problemas originales: modificarlos!

Incluyan en el formulario los pseudocódigos!

Sumario

Ejemplo: BFS

Ejercicio (I3 P4(b) - 2022-2)

El diámetro de un árbol no dirigido conexo T se define como el largo del camino más largo en T. Suponga que existe un único camino de largo máximo en T con extremos u y v. Si x es un nodo cualquiera, se puede demostrar que el nodo más lejano a x es u o v. Usando este resultado, proponga un algoritmo que determine el diámetro de un árbol T.

Ejemplo: BFS

La propiedad descrita permite plantear el siguiente algoritmo

`Diameter`(V, E)**:**

$x \leftarrow$ cualquier nodo de V

$d \leftarrow \texttt{BFS}(x)$

$u \leftarrow$ nodo que tiene máximo $d[\cdot]$

$d \leftarrow \texttt{BFS}(u)$

$D \leftarrow \max\{d[v] \mid v \in V\}$

return D

donde se usa una versión modificada de `BFS` para que retorne el arreglo con distancias desde la fuente, así como el nodo asociado a cada distancia.

Ejemplo: Dijkstra

Ejercicio (I3 P4(a) - 2022-2)

Considere un grafo dirigido $G = (V, E)$ que en lugar de aristas con pesos tiene nodos con pesos. El problema de rutas más cortas desde una fuente se define análogamente, donde el costo de un camino es la suma de los costos de sus nodos. Proponga cómo modificar el grafo para poder utilizar el algoritmo de Dijkstra para resolver dicho problema con la misma complejidad vista en clase.

Ejemplo: Dijkstra

Ejercicio (I3 P4(a) - 2022-2)

Al iniciar el recorrido Dijkstra desde una fuente s, el costo de este nodo no se considera pues está presente en todos los posibles caminos que salen de él. Nos interesamos en los costos para llegar a los vecinos en un paso.

De acuerdo con esto, dado un grafo con pesos en sus nodos, podemos modificar el grafo incluyendo pesos en sus aristas de acuerdo al nodo de llegada:

```
CreateWeights(V, E):
    for v ∈ V :
        for (x, y) ∈ E :
            if v = y :
                cost(x, y) ← cost(v)
```

Al ejecutar Dijkstra, el costo resultando de cada camino debe ser aumentado en $cost(s)$ para incorporar el costo de la fuente.

Ejemplo: Heaps

Ejercicio (I1 P2 - 2020-2)

Escriba un algoritmo $\mathcal{O}(n)$ que transforme un ABB (árbol binario de búsqueda) con n nodos e un min-Heap. Demuestre la correctitud de su algoritmo.

Aprovecharemos que en un ABB T podemos obtener sus elementos ordenados:

- Recorrer sub-árbol izquierdo primero
- Guardar nodo padre en un arreglo A
- Recorrer sub-árbol derecho

Este procedimiento permite obtener un arreglo A ordenado a partir del árbol inicial T. Observemos que cada nodo es visitado una vez, y el algoritmo toma $\mathcal{O}(n)$.

Notemos que $A[0 \dots n-1]$ está ordenado (de menor a mayor).

Ejemplo: Heaps

Ejercicio (I1 P2 - 2020-2)

Notemos que $A[0 \ldots n-1]$ está ordenado (de menor a mayor).

Demostraremos que este arreglo ya es un min-heap. Sea k un índice cualquiera tal que tiene hijos en el heap.

- Los índices de sus posibles hijos son $2k+1$ y $2k+2$
- Como los índices cumplen $k < 2k+1 < 2k+2$...
- ... y el arreglo está ordenado,
- concluímos que $A[k] < A[2k+1]$ y $A[k] < A[2k+2]$

Esto se cumple para un k arbitrario y por lo tanto, A es un min-heap.

Ejemplo: Heaps

Ejercicio

Entregue el pseudocódigo de la operación `HeapDelete`(A, i) que recibe un heap representado como arreglo A y una posición i del arreglo, y elimina el elemento $A[i]$ reestableciendo la propiedad de heap balanceado.

input : heap representado como arreglo $H[0 \ldots n-1]$, índice i

`HeapDelete`(H, i)**:**

$j \leftarrow$ última celda no vacía de H
$H[i] \leftarrow H[j]$
$H[j] \leftarrow \varnothing$
`SiftDown`(H, i)

Sumario

Recomendaciones finales

Para los ejemplos vistos en clase

- Replicarlos comprendiendo los pasos de su resolución
- Asegurarse de poder motivar las decisiones

Pautas anteriores

- Hay harto material resuelto en el repo!
- No se aprendan pautas... seleccionen y aprovéchenlas
- Planifiquen su solución antes de verla, y luego consulten la pauta

Objetivos de la clase

- ☐ Recordar elementos esenciales de los contenidos a evaluar
- ☐ Identificar diferencias de contenidos que aplican a un mismo escenario
- ☐ Aplicar algunos de los contenidos a ejemplos concretos
- ☐ Conocer enfoque esperado en las respuestas de algunos ejemplos