Repaso I2

Clase 21

IIC 2133 - Sección 1

Prof. Sebastián Bugedo

Sumario

¿Cómo están?

Los grandes temas

En esta interrogación abarcamos los siguientes macrotemas

1. Diccionarios
   - Árboles de búsqueda (binarios, AVL, 2-3, rojo-negro)
   - Tablas de hash
2. Orden en tiempo lineal
   - Orden de strings y números por dígitos (counting sort, radix sort)
3. Estrategias algorítmicas
   - Backtracking
   - Algoritmos codiciosos (greedy)
   - Programación dinámica
4. Intro a algoritmos en grafos
   - DFS (detección de ciclos, orden topológico, Kosaraju)

Planteados aquí por tema... no necesariamente
se condicen uno a uno con las 4 preguntas de la I2!

Interrogación 2

Objetivos a evaluar en la I2

- ☐ Comprender implementaciones de diccionarios y sus diferencias
- ☐ Aplicar algoritmos de ordenación lineal
- ☐ Diseñar algoritmos usando técnicas e ideas estudiadas
- ☐ Aplicar algoritmos sobre grafos dirigidos

Varios objetivos pueden incluirse en cada pregunta

Interrogación 2

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 I2: promedio de las 3 preguntas entregadas

Interrogación 2

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

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

Diccionarios

Definición

Un **diccionario** es una estructura de datos con las siguientes operaciones

- **Asociar** un valor a una llave
- **Actualizar** el valor asociado a una llave
- **Obtener** el valor asociado a una llave
- En ciertos casos, **eliminar** de la estructura una asociación llave-valor

Objetivo central: búsqueda eficiente

Diccionarios: dos enfoques

Vimos dos instancias de diccionarios

1. Árboles de búsqueda
   - Binarios AVL
   - 2-3
   - Binarios rojo-negro
2. Tablas de Hash

Árboles de búsqueda

Tres tipos estudiados: binarios AVL y rojo-negro, y árboles 2-3

Aspectos esenciales

- Los tres tipos tienen la **propiedad de búsqueda**: valores ordenados en

  $$\text{hijo izquierdo} < \text{padre} < \text{hijo derecho}$$

- Cada tipo tiene una noción de **balance** dada por sus operaciones
  - AVL cuida la altura de sus hijos recursivamente
  - 2-3 mantiene balance a través de la mantención de hojas en un mismo nivel
  - Rojo-negro cuida la cantidad de nodos negros hacia las hojas

Importancia del balance: mantener el árbol con profundidad logarítmica

Árboles de búsqueda

Operaciones

- Búsqueda gracias a la propiedad de búsqueda
- Inserción
  - AVL: involucra posibles rotaciones
  - 2-3: involucra posibles splits
  - Rojo-negro: involucra posibles rotaciones y cambios de color
- Eliminación: en general compleja y recursiva

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

Las operaciones se benefician del balance

Árboles de búsqueda

Orientaciones para el estudio

- ☐ Comprender la estrategia de balance de cada tipo
- ☐ Construir árboles pequeños con inserción de llaves suscesivas
- ☐ Definir pequeñas secuencias de inserción que generan árboles más/menos desbalanceados
- ☐ Comparar el desempeño de los árboles en posibles situaciones prácticas. ¿Cuál se puede portar mejor?

AVISO: No evaluaremos eliminación de nodos (especialmente en rojo-negro)

Tablas de Hash

Aspectos esenciales de las tablas de hash

- La búsqueda se basa en le resultado de una **función de hash**
- El valor de hash se usa como índice en un arreglo (la tabla)
- Cada colisión se almacena mediante algún procedimiento
  - Direccionamiento abierto: dónde se pueda
  - Encadenamiento: usar listas ligadas de colisiones

También se podrían usar árboles para manejar colisiones.
Beneficio: la búsqueda es más rápida que en las listas

Tablas de Hash

Operaciones

- Búsqueda en dos fases
  1. Identificación de celda: $\mathcal{O}(1)$ si la función de hash es adecuada
  2. Búsqueda en colisiones
- Inserción también en dos fases

Note que la complejidad depende de cómo se almacenan colisiones

- Listas: $\mathcal{O}(n)$ en el peor caso
- Árboles: $\mathcal{O}(\log(n))$ en el peor caso, manteniendo balance

Se pueden usar otras formas de almacenamiento de colisiones: tablas de hash por ej

Tablas de hash

Orientaciones para el estudio

- ☐ Comprender funciones de hash y propiedades deseables (impacto en las colisiones)
- ☐ Comparar desempeño de técnicas de resolución de colisiones
- ☐ Uso de tablas para almacenar y consultar valores

Sumario

Orden lineal

Aspectos esenciales de los algoritmos de orden lineal

- Operan sobre naturales (índices para un arreglo)
- Counting Sort permite ordenar naturales de un conjunto acotado
- La idea se generaliza a Radix para ordenar palabras cualesquiera
  - Cada símbolo se asocia con un natural
  - Requiere un algoritmo estable como Counting Sort

Estos algoritmos ordenan n datos en tiempo $\mathcal{O}(n)$
si la cantidad de símbolos diferentes es $\mathcal{O}(n)$

Orden lineal

Orientaciones para el estudio

- ☐ Comprender cuándo se pueden ocupar los algoritmos lineales
- ☐ Comprender pseudocódigo de Counting Sort y Radix para potenciales modificaciones

Sumario

Estrategias algorítmicas

Estudiamos tres de ellas: Backtracking, greedy y programación dinámica

Aspectos esenciales

- Cada una se centra en una forma de tomar decisiones y avanzar
  - Backtracking: se puede arrepentir y cambiar su decisión
  - Codiciosos: toman la mejor decisión aparente en el momento y no se arrepienten
  - Dinámica: comparan soluciones de subproblemas, recordando
- Las tres se pueden usar para optimizar, pero greedy y dinámica son las más indicadas para esto

Estrategias algorítmicas

Backtracking

- Requiere considerar **satisfacción de restricciones**
- Al evaluar una decisión posible, se verifica si es factible *en este momento*
- Se hace un llamado recursivo para determinar si dicha decisión puede ser exitosa en última instancia
- Si no lo es, entonces se revierte y se toma otra
- Cuando no quedan opciones, se da por fracasado el llamado actual

Estrategias algorítmicas

Algoritmos codiciosos

- Exclusivos para problemas de optimización
- Se basan en una **estrategia codiciosa**
- Esta define qué elemento/decisión tomar a continuación
- Existen problemas sin una estrategia codiciosa exitosa

Estrategias algorítmicas

Programación dinámica

- Muy usados para problemas de optimización, pero no de forma exclusiva
- Se basan en una **ecuación de recurrencia**
- Esta compara decisiones posibles (recursivas)
- La recurrencia sugiere la forma del algoritmo diseñado
- **Se deben guardan** los resultados de subproblemas para reciclarlos

Orden lineal

Orientaciones para el estudio

- ☐ Comprender casos de uso de las tres estrategias
- ☐ Conocer la idea general del funcionamiento de cada estrategia
- ☐ Diseñar algoritmos usando estas estrategias

Sumario

Algoritmos basados en DFS

- **Búsqueda en profundidad** que recorre aristas hasta agotarlas
- Permite resolver variados problemas
  - Detección de ciclos: aristas hacia atrás en grafos dirigidos
  - Orden topológico
  - Kosaraju: componentes fuertemente conectadas

Orden lineal

Orientaciones para el estudio

- ☐ Comprender el funcionamiento del recorrido DFS de un grafo y cómo su modificación origina nuevos algoritmos
- ☐ Comprender requisitos de cada algoritmo
- ☐ Comprender qué problema resuelve cada algoritmo

Sumario

Ejemplo: Backtracking

Ejercicio (I2 P4 - 2022-2)

Para asegurar la conectividad del trasporte en el extremo sur del país existen tramos en los cuales se utilizan barcazas para llevar vehículos (autos particulares y camiones) entre dos puntos que no tienen conectividad por tierra. La capacidad de la barcaza se define en función de los metros lineales de vehículos que puede acomodar (4 filas de vehículos de máximo 15 metros cada fila son 60 metros lineales de capacidad máxima) y el peso máximo total que puede transportar (por ejemplo 240.000 kilos de carga). Así una barcaza B se define como

(`B.n_filas`, `B.m_por_fila`, `B.max_carga`).

Los vehículos V que están a la espera de transporte están en una fila y tienen determinado su largo y peso (`V.largo`, `V.peso`) expresados en metros y kilogramos.

Ejemplo: Backtracking

Ejercicio (I2 P4 - 2022-2)

(a) [1 pto.] Identifique las Variables, Dominios y Restricciones del problema.

Denotaremos por w_i y ℓ_i el peso y largo del auto i-ésimo.

- Variables $X = \{x_1, \ldots, x_n\}$, una para cada auto indicando si se sube o no a la barcaza
- Dominios idénticos para cada variable: $\{0, 1\}$, donde 1 indica que sí se sube
- Restricciones
  - Peso máximo: $W =$ `B.max_carga` tal que
    $$\sum_{i=1}^{n} x_i \, w_i \leq W$$
  - Largo máximo: $L =$ (`B.n_filas`) $\cdot$ (`B.m_por_fila`) tal que
    $$\sum_{i=1}^{n} x_i \, \ell_i \leq L$$

Ejemplo: Backtracking

Ejercicio (I2 P4 - 2022-2)

(b) [3 ptos.] Diseñe un algoritmo para definir qué vehículos de la fila transportar de modo de maximizar la cantidad de vehículos sin superar la capacidad de la barcaza (en metros lineales totales y la carga máxima de la misma). **No considere** la capacidad de cada fila de la barcaza, sino la **capacidad total**.

Ejemplo: Backtracking

Ejercicio (I2 P4 - 2022-2)

Supondremos que

- $X[0 \ldots n-1]$ es el arreglo para guardar los valores binarios
- $Y[0 \ldots n-1]$ es el arreglo para guardar la asignación óptimo, inicializado con ceros
- $\#(X)$ entrega el número de autos asignados en X
- $\ell(i)$ entrega el largo del auto i
- $w(i)$ entrega el peso del auto i

Ejemplo: Backtracking

Ejercicio (I2 P4 - 2022-2)

input : $X[0 \ldots n-1]$ arreglo, L largo permitido,
W peso permitido, i índice a asignar en X

`Backtrack` (X, L, W, i)**:**

if $i = n$ **:**

if $\#(X) > \#(Y)$ **:**

$Y \leftarrow$ copia de X

else:

for $j \in \{0, 1\}$ **:**

if *asignar* $X[i] \leftarrow j$ *no supera restricciones* **:**

$X[i] \leftarrow j$

`Backtrack`$(X,\ L - j \cdot \ell(i),\ W - j \cdot w(i),\ i + 1)$

El algoritmo se llama con `Backtrack`$(X, L, W, 0)$ y una vez que termina, Y contiene la asignación óptima.

Ejemplo: Backtracking

Ejercicio (I2 P4 - 2022-2)

(c) [2 ptos.] Modifique su algoritmo anterior para que entregue en qué fila de la barcaza va cada vehículo a transportar, al maximizar la cantidad de vehículos sin superar la capacidad de la barcaza.

Primero, extendemos el conjunto de variables para poder asignar valores en $\{0, 1, 2, 3, 4\}$. El valor 0 denota no incluir el auto, y los demás son los números de fila asignados en caso de incluirlo.

Con este cambio, solo hay que incluir los largos permitidos de cada fila y verificarlo en la restricción. Para esto, se puede usar un arreglo $L[0 \ldots 3]$ donde se mantienen los largos disponibles por fila. Al inicio es $L = [\texttt{B.m\_por\_fila}, \texttt{B.m\_por\_fila}, \texttt{B.m\_por\_fila}, \texttt{B.m\_por\_fila}]$

Ejemplo: Backtracking

Ejercicio (I2 P4 - 2022-2)

input : $X[0 \ldots n-1]$ arreglo, $L[0 \ldots 3]$ arreglo de largos permitidos, W peso total permitido, i índice a asignar en X

```
Backtrack2 (X, L, W, i):
    if i = n :
        if #(X) > #(Y) :
            Y ← copia de X
    else:
        for j ∈ {0, 1, 2, 3, 4} :
            if asignar X[i] ← j no supera restricciones :
                X[i] ← j
                if j > 0 :
                    L[j − 1] ← L[j − 1] − ℓ(i)
                Backtrack2(X, L, W − j · w(i), i + 1)
                if j > 0 :
                    L[j − 1] ← L[j − 1] + ℓ(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

La clave del éxito:

Playlist: DatiWawos Segundo Acto

Además sigan en instagram:
@orquesta_tamen

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

Epílogo

Ve a

www.menti.com

Introduce el código

1710 2511

O usa el código QR