

Repaso I2

Clase 19

IIC 2133 - Sección 2

Prof. Mario Droguett

Sumario

Introducción

Tablas de hash

Orden lineal

Estrategias algorítmicas

Un ejemplo de prueba

Cierre

Los grandes temas

En esta interrogación abarcamos los siguientes macrotemas

1. Diccionarios
 - 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

Más de un tema podría corresponderse con una misma pregunta en la I2

Interrogación 2

Objetivos a evaluar en la I2

- ☐ Comprender implementación y uso de tablas de hash
- ☐ Aplicar algoritmos de ordenación lineal
- ☐ Diseñar algoritmos usando técnicas e ideas estudiadas

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

Introducción

Tablas de hash

Orden lineal

Estrategias algorítmicas

Un ejemplo de prueba

Cierre

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

Tablas de Hash

Aspectos esenciales de las tablas de hash

- La búsqueda se basa en el 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

Introducción

Tablas de hash

Orden lineal

Estrategias algorítmicas

Un ejemplo de prueba

Cierre

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

Introducción

Tablas de hash

Orden lineal

Estrategias algorítmicas

Un ejemplo de prueba

Cierre

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 guardar** 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

Introducción

Tablas de hash

Orden lineal

Estrategias algorítmicas

Un ejemplo de prueba

Cierre

Ejemplo: Greedy + dinámica

Ejercicio (I2 P3 - 2023-1)

Sea $A = \{a_1, \dots, a_n\}$ un conjunto de naturales (distintos) y K una cota natural. El problema de *subset sum* responde si acaso existe un subconjunto $S \subseteq A$ de forma que la suma de los elementos de S sea exactamente igual a K .

Para resolver este problema, consideremos su versión como problema de optimización: determinar un $S \subseteq A$ tal que la suma de sus elementos es máxima y además no supera la cota K .

Observe que el problema original es de decisión, pero lo resolvemos con su versión de optimización

Ejemplo: Greedy + dinámica

Ejercicio (I2 P3 - 2023-1)

- (a) [1 pto.] Considere un algoritmo “codicioso” que revisa los elementos a_1, \dots, a_n en ese orden y determina si se debe incluir a_i con la siguiente estrategia codiciosa: *se le incluye si, y solo si, hacerlo no supera la suma permitida por la cota*. Demuestre que esta estrategia no es correcta.

Propuesta de solución.

Consideramos el conjunto $A = \{2, 3, 6\}$ y la cota $K = 6$. En este caso, el algoritmo codicioso selecciona el subconjunto $S_{\text{greedy}} = \{2, 3\}$ con suma 5. Pero el óptimo es $S_{\text{opt}} = \{6\}$. Vemos que la estrategia codiciosa no es correcta.

Ejemplo: Greedy + dinámica

Ejercicio (I2 P3 - 2023-1)

- (b) [2 ptos.] Dado un conjunto $A = \{a_1, \dots, a_n\}$ y cota K , sea $f(k, T)$ la suma máxima lograble cuando se considera el subproblema de optimización para $\{a_1, \dots, a_k\}$ con $k \leq n$ y tomando la cota $T \leq K$. Determine una ecuación de recurrencia para $f(k, T)$, especificando casos base y borde de ser necesario.

Propuesta de solución.

Proponemos la siguiente función recursiva

$$f(k, T) = \max\{f(k-1, T), f(k-1, T - a_k) + a_k\}$$

Las condiciones de borde cuando se acaban los elementos ($k = 0$) o cuando la suma disponible se agotó

$$f(k, T) = \begin{cases} 0 & \text{si } T \geq 0 \wedge k = 0 \\ -\infty & \text{si } T < 0 \end{cases}$$

Ejemplo: Greedy + dinámica

Ejercicio (I2 P3 - 2023-1)

- (c) [2 ptos.] Plantee el pseudocódigo de un algoritmo de programación dinámica que permita obtener la suma óptima para un conjunto $A = \{a_1, \dots, a_n\}$ y cota K , e indicando en qué estructura de datos almacenará los óptimos de los subproblemas.

Ejemplo: Greedy + dinámica

Propuesta de solución.

input : Índice $k \in \{1, \dots, n\}$ y cota T

output: Suma óptima

SubsetSum(k, T):

```
1   if  $k = 0$  AND  $T \geq 0$  :  
2       return 0  
3   elif  $T < 0$  :  
4       return  $-\infty$   
5   else:  
6       if  $M[k][T] \neq \emptyset$  :  
7           return  $M[k][T]$   
8       else:  
9            $M[k][T] \leftarrow$   
10           $\max\{\text{SubsetSum}(k-1, T), \text{SubsetSum}(k-1, T - a_k) + a_k\}$   
11          return  $M[k][T]$ 
```

donde M es un arreglo de arreglos global, donde se guardan los subóptimos.

Ejemplo: Greedy + dinámica

Ejercicio (I2 P3 - 2023-1)

- (d) [1 pto.] Describa cómo responder a la pregunta de si existe o no un subconjunto con suma exactamente K y explique cómo determinar el subconjunto a partir de la estructura usada en su solución en (c). No necesita entregar un pseudocódigo, sino que basta con una descripción a alto nivel.

Propuesta de solución.

A partir del arreglo M , se comienza revisando $M[n][K]$. Este valor se compara con todos los valores $M[n-1][T]$ para los T almacenados. Si coincide con alguno, significa que el dato a_n no se incluye. Si coincide con alguno añadiendo a_n , significa que sí se incluye. Este proceso se repite revisando hacia atrás, deduciendo si cada elemento se incluye o no.

Sumario

Introducción

Tablas de hash

Orden lineal

Estrategias algorítmicas

Un ejemplo de prueba

Cierre

Recomendaciones finales

Para los ejemplos vistos en clase

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

Pautas anteriores

- Hay hart 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