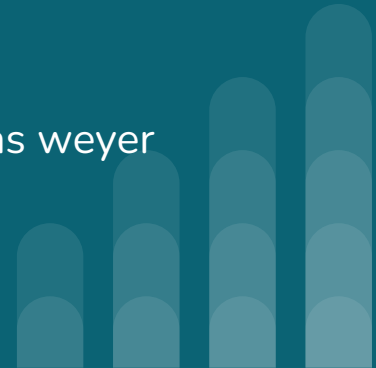


Grupo 1

Final programación III

Integrantes: Rocio Solberg, Joaquina Glerean, Facundo Planas y Nicolas weyer



Buen algoritmo

Un algoritmo es una secuencia de pasos para resolver un problema específico. Un buen algoritmo se caracteriza por ser simple, claro y eficiente.

Características

- Simplicidad: Fácil de entender, mantener y modificar.
- Claridad: Debe ser explícito en cuanto a sus operaciones.
- Eficiencia: Optimizado en tiempo y recursos, evaluado mediante notación Big O.

Tiempo de ejecución y recurrencias

El tiempo de ejecución de un programa, denotado como $T(n)$, describe cómo varía su rendimiento en función del tamaño de la entrada n . Se utiliza la complejidad asintótica para prever el rendimiento a gran escala.

Las recurrencias describen el comportamiento de algoritmos recursivos, expresando su costo en función de subproblemas más pequeños.

Generalmente tiene la forma de:

$$T(n) = a \cdot T(n/b) + f(n):$$

Complejidad asintótica y notación Big(O)

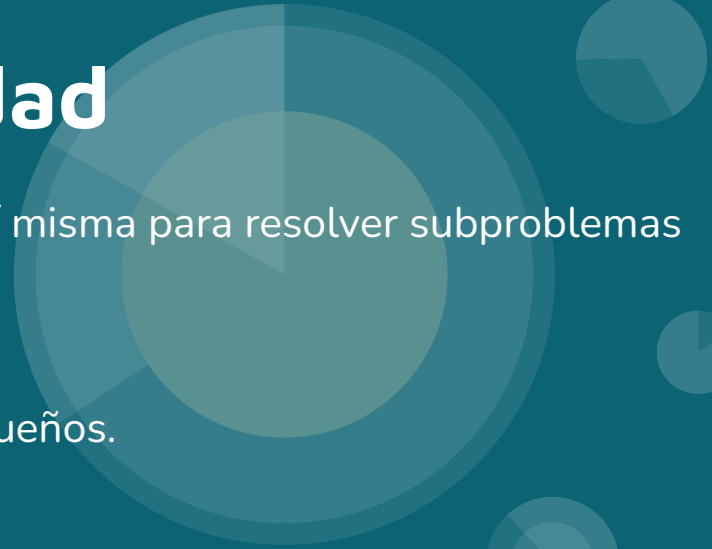
La complejidad asintótica es el estudio de cómo el tiempo de ejecución de un algoritmo varía conforme el tamaño de la entrada crece, especialmente cuando esta es muy grande. La notación Big O (O) se utiliza para expresar la complejidad asintótica, proporcionando una medida estandarizada de la eficiencia de un algoritmo en términos de su peor caso.

Recursividad

La recursividad permite que una función se llame a sí misma para resolver subproblemas más pequeños. Se basa en:

Caso base: Detiene la recursión.

Llamada recursiva: Resuelve subproblemas más pequeños.



Divide y vencerás

Se aplica cuando un problema complejo puede descomponerse en subproblemas más simples del mismo tipo.

Pasos:

- Caso base
- Dividir el problema
- resolver recursivamente los subproblemas
- Combinar las soluciones para resolver el problema original

Quicksort

Algoritmo de ordenamiento.

- Se elige pivote
- menores \rightarrow pivote \rightarrow mayores.
- Repetir el proceso con las sublistas.
- Mejor caso = $n \cdot \log n$
- Peor caso = n^2

MergeSort

Algoritmo de ordenamiento por mezcla

- Dividir la lista en dos sublistas.
- Ordenar cada sublista recursivamente.
- Mezclar las dos sublistas.
- Complejidad = $n \cdot \log n$

Greedy

Decisión óptima local → Solución óptima global

- siempre óptimo: conjunto canónico

$\text{monedas} = \{1, 5, 10, 25\}$

- no siempre óptimo

$\text{monedas} = \{1, 3, 4\}$

```

public static void Monto(int monto) {
    int[] billetes = {1,5,10,25};
    int[] cantidad = new int[billetes.length];
    for(int i = billetes.length - 1; i >= 0 ; i--) {
        cantidad[i]= monto/billetes[i];
        monto=monto-billetes[i]*cantidad[i];
    }
    for(int i = billetes.length - 1; i >= 0 ; i--) {
        if (cantidad[i]!=0) {
            System.out.println(cantidad[i] + " billetes de " + billetes[i]);
        }
    }
}

```

monto = 61

$$61/25 = 2 \rightarrow 61 - 25*2 = 11$$

$$11/10 = 1 \rightarrow 11 - 10*1 = 1$$

$$1 / 5 = 0 \rightarrow 1 - 5*0 = 1$$

$$1 / 1 = 1 \rightarrow 1 - 1*1 = 0$$

2 billetes de 25

1 billete de 10

1 billete de 1

MOCHILA FRACCIONAL

- mochila con peso W .
- cantidad de objetos n .
- cada objeto tiene un valor y un peso.

MOCHILA FRACCIONAL - MÉTODO GREEDY

- ordenar listas \rightarrow valor / peso

	valor	peso	v / p
obj 1	12	4	3
obj 2	10	5	2
obj 3	11	3	3,6

obj3 - obj1 - obj2

- recorrer la lista
- entra \rightarrow agrego
- no entra \rightarrow fracción

Grafos

Existen los dirigidos y los no dirigidos:

- Dirigido: Solo de $V1$ a $V2$
- No dirigido: $V1$ a $V2$ y viceversa

Por otro lado están los ponderados y los no ponderados:

- Ponderados: las aristas tienen peso

(Están los conexos, los ciclos, los árboles)

Representación: lista y matriz de adyacencia

Árbol de Recubrimiento Mínimo (MST)

- Se trabaja con grafos ponderados
- Es un subgrafo conexo (incluye todos los V del grafo original)
- Árbol = no ciclos
- El costo total del árbol debe ser el mínimo posible

Se elige un nodo y se va buscando el camino de menor costo hasta el siguiente vértice. Ahora desde este vértice busco el siguiente camino de menor costo y así hasta recorrer todo el grafo (A menos que en el primer vértice hubiera otra camino menos costoso que todos los del siguiente)

Algoritmo de Prim y algoritmo de Kruskal

Prim: encuentra el MST de un grafo no dirigido

(Se agrega un nodo al MST, se busca la arista de menor peso que lo una con un nodo fuera del MST y se agrega el peso de la arista y el nodo al MST)

Kruskal: encuentra el MST de un grafo no dirigido

(Ordena las aristas por peso y las añade una a una al MST, sin ciclos ($V-1$ aristas). Al final queda todo el grafo cubierto igual que con Prim)

Complejidades

Prim es $O(V^2)$ con matriz de adyacencia, donde V es la cantidad de vértices

(Más eficiente cuando $E \approx V^2$) \rightarrow grafos más densos

Kruskal es $O(E \log E)$, donde E es la cantidad de aristas. $E \log E$ es por el ordenar las aristas

(Más eficiente cuando $E < V^2$) \rightarrow grafos más dispersos

Algoritmo de Dijkstra

Distancia mínima de un vértice origen (específico) a todos los vértices (grafo ponderado y no pesos negativos). Estrategia Greedy

Seleccionamos el origen, buscamos el camino más corto con sus vecinos. Primero se pone que el $d[V \text{ origen}] = \infty$ (como que estuviera aislado) y de ahí se compara la distancia a sus vecinos ($d[V] = 0$ al principio, porque apunta a sí mismo, el resto de V van en infinito al principio)

Complejidad variable según cola de prioridad:

$O((V + E) \log V) \rightarrow$ Heap binario

$O(E + V \log V) \rightarrow$ Heap de Fibonacci

Programación dinámica

Es una técnica algorítmica utilizada para resolver problemas complejos dividiéndolos en subproblemas más pequeños y reutilizando las soluciones de estos.

Principios:

Subproblemas: Los mismos subproblemas aparecen repetidamente, por lo que se almacenan sus soluciones para evitar recalcular.

Estructura óptima: La solución óptima de un problema se construye a partir de las soluciones óptimas de sus subproblemas.

La complejidad suele ser polinómica, típicamente $O(n^2)$ o $O(n \cdot W)$

Algoritmo de Floyd-Warshall

- camino más corto entre pares de vértices.
- matriz
 - existe arista \rightarrow peso
 - no existe \rightarrow infinito
 - $[i][i] \rightarrow 0$
- 3 ciclos \rightarrow recorrer vértices
 - \rightarrow recorrer matriz

Backtracking

- Construcción incremental de soluciones
- Retroceso (Backtracking)
- Descarte

Árbol de búsqueda

- Todas las soluciones en el árbol
- Nodo = elección
- Criptografía, puzzles, algoritmos de ajedrez por ejemplo

Complejidad

Temporal:

- Peor caso: exponencial
- Depende el caso: $O(n!)$ - combinaciones inválidas

Espacial:

- Profundidad del árbol de búsqueda
- Peor caso: longitud del problema

RECORRIDOS SOBRE GRAFOS

DFS → profundidad

- iniciación
- exploración profunda
- retroceso
- repetición

BFS → anchura

- iniciación
- exploración por niveles
- repetición
- finalización

Poda Alfa-Beta

Que es?

- Es una mejora del algoritmo Minimax, utilizado para encontrar movimientos óptimos en juegos de dos jugadores.

Funcionamiento:

- Alfa
- Beta
- Evaluación de nodos

Uniform Cost Search (UCS)

Que es?

- es una variante de búsqueda que encuentra el camino de menor costo entre un nodo inicial y un nodo objetivo en un grafo.

Funcionamiento:

- Inicialización
- Expansión
- Actualización
- Condición de parada

Heurística

Es una técnica de resolución de problemas basada en simplificaciones o reglas empíricas que permiten obtener soluciones aproximadas de manera eficiente, especialmente útil en problemas complejos donde los algoritmos exactos son demasiado costosos.

Características:

- Proporcionan soluciones rápidamente en comparación con los métodos exactos.
- No siempre ofrecen soluciones óptimas, pero suelen ser suficientemente buenas.
- Basadas en criterios locales para guiar la búsqueda.
- No garantizan encontrar la mejor solución posible o una solución factible en todos los casos.

Heurística con Aprendizaje

Este tipo de heurística incorpora aprendizaje, adaptándose y mejorando sus estrategias a medida que resuelve más instancias del problema.

Componentes:

- Heurística base: Algoritmo inicial para generar soluciones aproximadas.
- Mecanismo de aprendizaje: Ajusta la heurística utilizando modelos supervisados o no supervisados.
- Evaluación y retroalimentación: Mejora el rendimiento ajustando parámetros basados en resultados previos.