Universidad Nacional de General Sarmiento

Visualización de algoritmos de ordenamiento

Número de materia:
A0444

Comisión:
07

Profesores:
Sergio Santa Cruz
Nahuel Sauma
Gonzalo Godoy

#Grupo 6

Integrantes:
Benitez Leandro
Iglesias Mariano

Introducción

El objetivo de este trabajo es comprender y aplicar los principios básicos de la programación a través del diseño e implementación de algoritmos de ordenamiento. Se busca analizar cómo estos algoritmos organizan conjuntos de datos, comparando su eficiencia y funcionamiento, así como describir cómo generarlos.
A partir de la creación y prueba de distintos métodos de ordenamiento (como bubble, selection o insertion), se pretende fortalecer el pensamiento lógico y la capacidad de resolver problemas mediante la programación estructurada, así como relevar la capacidad de generar esquemas de pseudo programa y programa en Python.

Desarrollo

Trabajamos en la implementación de distintos algoritmos de ordenamiento (bubble, Selection, Insertion y Comb), entendidos como procedimientos que toman una lista y reorganizan sus elementos según un criterio definido, comparando posiciones y realizando los intercambios necesarios hasta obtener una secuencia ordenada.
Para comprender su funcionamiento y adaptarlos al formato del contrato init(vals), step(), analizamos videos explicativos y estudiamos paso a paso cómo traducir cada lógica al modelo por ciclos del visualizador, identificando qué punteros utilizar, cómo controlar las comparaciones y en qué momentos debe producirse cada intercambio; a partir de ese análisis se desarrollaron las decisiones y ajustes específicos para cada algoritmo.

Dificultades y decisiones:

Algoritmo Bubble

Dificultades:

- Determinar qué elementos se considerarán para tomar una lista “x” y ordenarlos según bubble (considerando si la posición de la lista que se analiza es mayor a la siguiente y reposicionando luego).
- Al reproducir, ordena, pero no repite el bucle

Decisiones:

- Utilizaremos nuevos elementos (“a” y “b”)
- Intentamos con “for in range” y fallamos miserablemente.
- Para corregir el ordenamiento, “j” será un modelo para identificar el problema. “j” también se utilizará para controlar los valores (y posiciones) de “a” y “b” y definir si se repite o no a partir de un “IF”

Algoritmo Selection

Dificultades:

- Al principio el algoritmo encontraba el mínimo pero no seguía ordenando, porque siempre devolvía “done”:True y el visualizador cortaba el proceso
- El algoritmo buscaba el mínimo pero nunca llegaba a intercambiarlo, porque no cambiamos la variable fase a “swap”
- También fue complicado manejar bien los índices “i”, “j” y “min_idx” para que no se salgan del rango de la lista y el visualizador no se corte

Decisiones:

- Decidimos usar una variable fase con dos estados: “buscar” y “swap” para que el algoritmo sea más fácil de controlar
- Ajustamos el “return” para que sólo devuelva “done”:True cuando realmente se haya terminado de ordenar toda la lista (if i>n-1:)

- En la fase “buscar” hacemos únicamente la recorrida para encontrar el mínimo y recién cuando termina el barrido pasamos a la fase “swap”

Algoritmo Insertion

Dificultades:

- El algoritmo insertion era relativamente fácil. Al realizar el checklist, si forzábamos a que el valor “j” sea igual a “0”, el código no se ejecutaba en algunos ciclos

Decisiones:

- Para evitar que el algoritmo no refleje valores definimos que la variable “j” cambie de signo en “Step”

Algoritmo Comb

Dificultades:

- Al finalizar el bucle se repite infinitamente
- Al llegar al momento en que debería realizar una validación similar al “bubble”, no lo ejecuta
- Si la cantidad de columnas es impar, el último ciclo no queda correctamente ordenado.

Decisiones:

- Tomamos algunos conceptos del bubble, para que el código se ejecute correctamente.

Conclusiones

El trabajo con los algoritmos de ordenamiento nos permitió ver un “pantallazo”, de cómo se estructuran, qué decisiones requiere su implementación y cómo se articulan sus pasos dentro de un contrato.
A partir de pruebas, errores y ajustes (apoyados en material teórico y videos) logramos que cada algoritmo cumpliera su recorrido completo y ordenara correctamente, fortaleciendo tanto la lógica de programación como la capacidad de traducir pseudocódigo en código.

Universidad Nacional de General Sarmiento

Bibliografía y enlaces:

Sorting - Visual go
https://visualgo.net/en/sorting?slide=1

Comparison among Bubble Sort, Selection Sort and Insertion Sort
https://www-geeksforgeeks-org.translate.goog/dsa/comparison-among-bubble-sort-selection-sort-and-insertion-sort/?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es&_x_tr_pto=tc

Aprende Divide y Vencerás | Subarreglo Máximo | Diseño de Algoritmos
https://www.youtube.com/watch?v=UxtAqHOb8aw