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.

Descripción, dificultades y decisiones:

Bubble Sort

Descripción

Según lo investigado, este algoritmo de ordenamiento compara pares de elementos adyacentes y los intercambia si están en el orden incorrecto. Considera el primer elemento de la izquierda y lo compara con el de la derecha. Si el de la izquierda es mayor, intercambia posiciones. Una vez intercambiado, o que el de la izquierda sea menor, pasa al siguiente. Este proceso se repite varias veces hasta que no se necesiten más intercambios, momento en el cual la lista queda ordenada.

Dificultades

- Al ser el primer algoritmo, dentificar correctamente qué elementos comparar dentro de la lista para evaluar si la posición actual es mayor que la siguiente.
- Lograr que el algoritmo repita el ciclo completo de comparación; en la versión inicial, la reproducción realizaba un solo barrido y no reiteraba el bucle.

Decisiones

- Se introdujeron las variables auxiliares “a” y “b” para representar de manera clara los índices que se comparan en cada paso.
- Se descartó el uso de un for in range, ya que no permitía un control adecuado del estado entre pasos.
- Se utilizó la variable “j” como puntero principal para manejar las posiciones de comparación y para controlar la repetición del ciclo mediante condiciones if.

Insertion Sort

Descripción

Este algoritmo de ordenamiento toma cada elemento y lo compara con los ubicados a la izquierda. Inserta en la posición correcta dentro de la lista, desplazando hacia la derecha los valores mayores.

Dificultades

- La implementación general fue sencilla, pero al forzar que la variable j tomara el valor 0, algunos ciclos del algoritmo dejaban de ejecutarse.

Decisiones

- Para mantener la consistencia del proceso, se decidió permitir que la variable “j” cambie de signo dentro de step, evitando bloqueos y asegurando que todos los ciclos se ejecuten correctamente.

Selection Sort

Descripción

Este algoritmo de ordenamiento recorre la lista entera, para encontrar el elemento mínimo y lo intercambia con la posición inicial del segmento no ordenado. Una vez que identifica el

elemento mínimo, avanza a la siguiente posición en la lista, busca el siguiente elemento mayor inmediato al mínimo y repite el proceso con el resto de la lista.

Dificultades

- El algoritmo devolvía siempre "done": True luego de encontrar el primer mínimo, por lo que el visualizador detenía el proceso antes de completar la ordenación.
- Aunque el mínimo se detectaba correctamente, nunca se realizaba el intercambio porque la variable que indicaba la fase no cambiaba al estado “swap”.
- Manejar adecuadamente los índices i, j y min_idx fue complejo, especialmente para evitar desbordes de rango y cortes inesperados del visualizador.

Decisiones

- Se implementó una variable de control fase con dos estados: "buscar" y "swap", facilitando el seguimiento del ciclo completo.
- Se ajustó el valor de retorno para que "done": True sólo se produzca cuando toda la lista esté ordenada (if i > n - 1:).
- En la fase “buscar” se realiza exclusivamente la comparación para encontrar el mínimo, y sólo al finalizar ese barrido se pasa a la fase “swap”.

Comb Sort

Descripción

Este algoritmo de ordenamiento es similar al Bubble Sort, usando una distancia inicial grande entre elementos a comparar, que se va reduciendo por un factor constante llamado shrink factor. Esto acelera la eliminación de “tortugas” (valores pequeños atrapados al final de la lista).

Dificultades

- El algoritmo entraba en un ciclo infinito una vez finalizado el recorrido principal.
- Cuando debía realizar comparaciones similares a Bubble Sort (en la fase final del algoritmo), no las ejecutaba correctamente.
- Cuando la cantidad de columnas era impar, el último ciclo no quedaba ordenado de forma adecuada.

- Se incorporaron algunos elementos del funcionamiento del algoritmo Bubble Sort para garantizar que, una vez que la distancia se reduce a 1, el proceso de comparación e intercambio se complete 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