

**Instituto Tecnológico de Mérida**

*Campus Poniente*

***“Estructura de datos”***

Alumno: Luis Alberto Salazar Canché

Matricula: **E23080244**

Carrera: Ingeniería en sistemas computacionales

Fecha: 13/05/24

Materia: Estructura de datos

Docente: Ing. Armando López Valadez

***ADA2 -Metodos de ordenamiento***

El alumno investigará 3 ejercicios en donde utilice los siguientes métodos de ordenamiento, desarrollar en Python.

1. ShellSort
2. Quicksort
3. Heapsort
4. Radix

***Ejercicio 1 (ShellSort)***

Es un algoritmo de ordenamiento basado en la inserción directa, pero mejorado para ser más eficiente en listas grandes. Utiliza la técnica de "reducción de intervalos" para ordenar elementos que están lejos uno del otro y luego reduce gradualmente el intervalo hasta que es 1, en cuyo punto se convierte en una ordenación por inserción directa.

*Funcionamiento*

* Divide la lista en sublistas con un cierto intervalo (gap).
* Ordena cada sublista usando una ordenación de inserción.
* Reduce el gap y repite el proceso hasta que gap es 1.

*Por qué se usa*

* Es más eficiente que la ordenación por inserción simple para listas grandes.
* Texto

  Descripción generada automáticamenteSu rendimiento depende del intervalo (gap) elegido.

***Ejercicio 2 (QuickSort)***

Es un algoritmo de ordenación eficiente que usa la estrategia de divide y vencerás. Selecciona un 'pivote' y particiona los elementos en dos sublistas, una con elementos menores que el pivote y otra con elementos mayores, y luego ordena las sublistas recursivamente.

*Funcionamiento*

* Selecciona un pivote.
* Particiona los elementos en sublistas basadas en comparaciones con el pivote.
* Ordena recursivamente las sublistas.

*Para qué se usa*

* Tiene un rendimiento promedio de O(n log n), aunque en el peor caso es O(n^2) (mitigable con una buena elección de pivote).
* Texto

  Descripción generada automáticamenteEs muy eficiente en la práctica y ampliamente utilizado.

***Ejercicio 3 (HeapSort)***

Es un algoritmo de ordenación basado en la estructura de datos llamada 'heap'. Convierte la lista en un max-heap y luego intercambia el primer elemento (máximo) con el último elemento del heap y reduce el tamaño del heap, repitiendo el proceso hasta que la lista esté ordenada.

*Funcionamiento*

* Convierte la lista en un max-heap.
* Intercambia el primer elemento (máximo) con el último y reduce el tamaño del heap.
* Repite hasta que el heap esté vacío.

*Para qué se usa*

* Tiene un rendimiento garantizado de O(n log n).
* Texto

  Descripción generada automáticamenteEs eficiente en términos de espacio, ya que no requiere memoria adicional.

*Conclusiones*

ShellSort es útil para listas de tamaño mediano y cuando la simplicidad de implementación es importante. No es el más rápido en términos de complejidad.

Quicksort es generalmente el más rápido en la práctica para una amplia gama de datos, especialmente con una buena estrategia de pivote, pero tiene un peor caso de O(n^2).

Heapsort es robusto con un rendimiento garantizado de O(n log n) y no requiere espacio adicional, pero puede ser más lento en la práctica que Quicksort debido a constantes más altas.

No hay como tal un mejor algoritmo, depende mucho del contexto y de las características que tengan los datos