هدف پروژه

مقایسه‌ی تجربی و تئوری الگوریتم‌های مرتب‌سازی از نظر:

(Time Performance) زمان اجرا

تعداد عملیات (مقایسه و جابجایی)

رفتار الگوریتم‌ها در داده‌های مختلف

داده‌ی **مرتب‌شده**

داده‌ی **معکوس‌شده**

داده‌ی **تصادفی**

ساختار کلی کد :

کد به 6 بخش تقسیم میشود.

1	وارد کردن کتابخانه ها
2	پیاده‌سازی الگوریتم‌های مختلف مرتب‌سازی
3	ثبت زمان و شمارش عملیات
4	تولید داده در حالت های مختلف
5	اجرای تست و ذخیره نتایج
6	رسم نمودار و تحلیل نهایی

بخش اول وارد کردن کتابخانه ها

برای ساخت داده های تصادفی Random

برای اندازه گیری زمان اجرا Time

برای رسم نمودار ها Matplotlib

برای نمایش نتایج در جدول Pandas

بخش دوم

پیاده سازی الگوریتم ها

در این پروژه 9 الگوریتم مرتب سازی پیاده میشود

نوع الگوریتم	دسته بندی	پیچیدگی زمانی
Bubble Sort	ساده	$O(n^2)$
Insertion Sort	ساده	$O(n^2)$
Selection Sort	ساده	$O(n^2)$
Merge Sort	تقسیم و حل	O(n log n)
Quick Sort	تقسیم و حل	در حالت میانگین O(n log n)
Heap Sort	مبتنی بر درخت	O(n log n)
Counting Sort	غیر مقایسه ای	O(n + k)
Radix Sort	غیر مقایسه ای	O(nk)
Shell Sort	insertionبهبود یافته	تقریبی $O(n \log^2 n)$

هر کدام از این الگوریتم ها جوری نوشته میشوند در قالب تابعی نوشته میشوند که دو کار را انجام دهند.

1.مرتب سازی داده ها

2. شمارش تعداد مقایسه ها و جابجایی ها

بخش سوم

ثبت زمان و شمارش عملیات

هر تابعی علاوه بر مرتب سازی، تعداد عملیات رو هم برمیگردونه.

به این ترتیب ما برای هر الگوریتم و هر حالت داده سه معیار داریم:

زمان

تعداد مقایسه ها

تعداد جابجایی ها

بخش چهارم

تولید داده ها

سه نوع داده تولید میکنیم:

داد های تصادفی

داده های از قبل مرتب شده

داد های معکوس

بخش پنجم اجرای تست

در این بخش، برای هر الگوریتم و هر حالت داده، زمان و عملیات‌ها اندازه‌گیری و دردیتا فریم ذخیره میشود.

بخش ششم رسم نمودار ها

سه نمودار اصلی رسم میشود:

زمان اجرا : برای مقایسه هر الگوریتم

تعداد مقایسه ها :نشان دهنده حجم عملیات منطقی

تعداد جابجایی ها :میزان تغییر در داده ها را نشان میدهد.

Bubble Sort	O(n)	$O(n^2)$	$O(n^2)$	مقایسه‌ای
Insertion Sort	O(n)	$O(n^2)$	$O(n^2)$	مقایسه‌ای
Selection Sort	$O(n^2)$	$O(n^2)$	$O(n^2)$	مقایسه‌ای
Merge Sort	O(n log n)	O(n log n)	O(n log n)	تقسیم و حل
Quick Sort	O(n log n)	O(n log n)	$O(n^2)$	تقسیم و حل
Heap Sort	O(n log n)	O(n log n)	O(n log n)	درختی
Counting Sort	O(n + k)	O(n + k)	O(n + k)	غیرمقایسه‌ای
Radix Sort	O(nk)	O(nk)	O(nk)	غیرمقایسه‌ای
Shell Sort	O(n log n)	$O(n \log^2 n)$	$O(n^2)$	بهبودیافته

نتیجه‌گیری

.کوچک مناسب هستند n فقط برای (Bubble, Insertion, Selection) الگوریتم‌های ساده

.در حالت تصادفی بسیار سریع‌ترند Quick و Merge

.در حالت معکوس ممکن است کند شود Quick

.برای داده‌های عددی (محدوده مشخص) فوق‌العاده سریع‌اند Radix و Counting

سرعت بالا و پایداری خوبی دارد HeapSort.