پروژه درس طراحی الگوریتم

استاد درس: دکتر زرین بال ماسوله استادیار: مهندس رضا امیر یعقوبی

اعضا: شهریار خلوتی- ۹۹۲۵۰۱۵ امیرحسین قاسمی-۹۹۲۵۰۷۱

پروژه اول: Github repo: https://github.com/ShahriarKh/sort-algorithms-visualizer

sort visualizer - در این پروژه چهارتا از الگوریتم های sorting را در بستر وب و با زبان برنامه نویسی javaScript ویژوالایز کردیم. این الگوریتم ها شامل: - BubbleSort - BucketSort - CountingSort - InsertionSort - SelectionSort هستند.
برای ران کردن این وب اپ در سیستم ابتدا npm را نصب کنید. سپس cmd را باز کرده و به دایرکتوری پروژه بروید. سپس npm i را بزنید و صبر کنید. سپس npm run dev را بزنید تا اپ در لوکال هاست لانچ شود. سپس مرورگر را باز کرده و به آدرس http://localhost:3000/ بروید.
یک ویدیو از دمو این پروژه هم به صورت screen record ارسال میشود که ببینید پروژه چطور کار میکند.

الگوریتم ها:

بابل سورت (Bubble Sort) : این الگوریتم دارای O(n^2) است که به دلیل وجود دو حلقه تو در تو دو هم است.
این الگوریتم در بدترین حالت همه عناصر را با هم مقایسه میکند که یعنی تعداد کل محاسبات میشود:

$$n*(n-1)/2 = (n-1) + (n-2) + ... + 1$$

باکت سورت (Bucket Sort) : در این الگوریتم کار های متفاوتی انجام میشود:
۱-محاسبه‌ی ماکزیمم عنصر در آرایه‌ی ورودی و ایجاد یک آرایه‌ی با طول برابر با این مقدار که پیچیدگی این مرحله O(n) است.
۲- قرار دادن هر عنصر از آرایه ورودی در کانتینر متناظر با آن که پیچیدگی زمانی این مرحله نیز O(n) است.
۳- استفاده ار یک الگوریتم برای مرتب سازی هر کانتینر که این هم(O(n است.
۴- ادغام کانتینر ها به ترتیب که این مرحله نیز پیچیدگی(O(n دارد.
حال اگر تعداد کانتینر ها را ثابت فرض کنیم، پیچیدگی زمانی همه این مراحل نیز O(n) میشود.

کانتینگ سورت(Counting Sort): پیچیدگی زمتنی این الگوریتم نیز برابر با O(n) است.
۱-ابتدا min, max آرایه داده شده باید پیدا شود که پیچیدگی O(n) دارد.
۲- سپس آرایه جدیدی با طول برابر با محدوده بین min , max تشکیل میشود و مقادیر آن برابر با ۰ مقدار دهی میشوند. این عملیات نیز O(n) است.

۳- الگوریتم در آرایه حلقه ای میزند و تعداد تکرار هر عنصر را در ارایه count ثبت میکند که باز هم دارای O(n) است.
۴- در مرحله بعد آرایه numbers را با توجه به اعداد موجود، in-place میکنیم.
در نتیجه این مراحل، پیچیدگی زمانی کل این الگوریتم O(n) است.

۵-اینسرژن سورت(insertion Sort) : پیچیدگی زمانی این الگوریتم برابر با(O(n^2 است.
۱- در ابتدا متغیر های iterationTime, setIsSorting, task تعریف میشونند که زمانبر نیستند.
۲-سپس یک حلقه for روی اعداد این آرایه اجرا میشود که پیچیدگی زمانی(O(n دارد.
۳-درون یک حلقه while از اندیس j به عقب حرکت می‌کنیم تا به اول آرایه برسیم و تا زمانی که عنصر numbers[j] از عنصر فعلی این حلقه بزرگتر باشد، اعضا این آرایه را رو به جلو حرکت میدهیم که در بدترین حالت دارای(O(n است.
۴-سپس عنصر فعلی را به در مکان مناسب در آرایه قرار میدهیم که O(1) است.
۵- در هر مرحله از حلقه for, تابع task فراخوانده میشود تا وضعیت جدید آرایه را به بخش ویژوالایزر ارایه کند که تاثیری در پیچیدگی زماتنی ندارد.
۶- در نهایت، مقدار setIsSorting به false تغییر پیدا میکند و آرایه مرتب شده برگردانده میشود.
بنابراین بطور کلی پیچیدگی این الگوریتم به صورت(O(n^2 است.

پروژه دوم: