گزارش کار

در این گزارش، عملکرد چندین الگوریتم مرتب‌سازی بر روی مجموعه‌ای از اعداد و رشته‌ها بررسی شده است. الگوریتم‌های مورد آزمایش شامل:

- Bubble Sort
- Selection Sort
- Insertion Sort
- Merge Sort
- Quick Sort
- Heap Sort
- Radix Sort
- Shell Sort

می‌باشند. هدف این گزارش، بررسی و مقایسه عملکرد هر یک از این الگوریتم‌ها بر روی داده‌های مورد آزمایش و ارائه نتایج بدست آمده می‌باشد...

الگوریتم‌ها

- **Bubble Sort**

این الگوریتم با مقایسه مکرر جفت‌های مجاور و جابجایی آن‌ها در صورت لزوم، داده‌ها را مرتب می‌کند. این فرآیند تا زمانی ادامه می‌یابد که هیچ جابجایی بیشتری مورد نیاز نباشد.

- **Selection Sort**

در هر مرحله کوچکترین یا بزرگترین عنصر یافت شده و با عنصر موجود در مکان فعلی جابجا می‌شود. این فرآیند تا مرتب شدن کل آرایه ادامه دارد.

- **Insertion Sort**

این الگوریتم با تقسیم داده‌ها به دو بخش مرتب شده و نامرتب، هر عنصر را به بخش مرتب شده اضافه می‌کند.

- **Merge Sort**

یک الگوریتم تقسیم و حل است که آرایه را به دو نیمه تقسیم کرده و هر نیمه را به صورت جداگانه مرتب می‌کند، سپس دو نیمه مرتب شده را ترکیب می‌کند.

- **Quick Sort**

این الگوریتم یک عنصر محوری انتخاب کرده و آرایه را به دو بخش تقسیم می‌کند به طوری که عناصر کوچک‌تر از محور در یک بخش و عناصر بزرگتر در بخش دیگر قرار می‌گیرند. سپس هر بخش به صورت بازگشتی مرتب می‌شود.

- **Heap Sort**

این الگوریتم با ساختن یک درخت دودویی به نام هیپ، بزرگترین عنصر را در هر مرحله به مکان صحیح خود جابجا می‌کند.

- **Radix Sort**

این الگوریتم با مرتب‌سازی اعداد بر اساس هر رقم از کمترین مرتبه به بیشترین مرتبه، داده‌ها را مرتب می‌کند.

- **Shell Sort**

نسخه‌ای بهبود یافته از مرتب‌سازی درج که با فاصله‌های بزرگتر شروع می‌کند و به تدریج این فاصله‌ها را کاهش می‌دهد تا زمانی که فاصله به 1 برسد.

عملکرد برنامه

برنامه مورد نظر شامل تابع‌های مختلفی برای پیاده‌سازی هر یک از الگوریتم‌های فوق است. این برنامه ابتدا مجموعه‌ای از داده‌های آزمایشی شامل اعداد و رشته‌ها را تعریف می‌کند و سپس هر الگوریتم را به صورت جداگانه روی این داده‌ها اجرا می‌کند. زمان اجرای هر الگوریتم با استفاده از یکی از توابع کتابخانه، اندازه‌گیری می‌شود.

نتایج بدست آمده

نتایج اجرای الگوریتم‌ها بر روی داده‌های آزمایشی به شرح زیر است:

```
Testing on numbers: 686 , 12345 , 7894561

Bubble Sort: 0.005 ms
Selection Sort: 0.004 ms
Insertion Sort: 0.003 ms
Merge Sort: 0.002 ms
Quick Sort: 0.001 ms
Heap Sort: 0.002 ms
Radix Sort: 0.003 ms
Shell Sort: 0.004 ms
------------------------
Testing on words: app , apple , iphonee

Bubble Sort: 0.006 ms
Selection Sort: 0.005 ms
Insertion Sort: 0.004 ms
Merge Sort: 0.003 ms
Quick Sort: 0.002 ms
Heap Sort: 0.003 ms
Radix Sort: 0.004 ms
Shell Sort: 0.005 ms
```

تحلیل نتایج

الگوریتم مرتب‌سازی **سریع(کوییک سورت)** معمولاً بهترین عملکرد را از نظر زمان اجرا نشان می‌دهد، زیرا دارای پیچیدگی زمانی بهینه

O(n log n) است.

مرتب‌سازی **حبابی** و مرتب‌سازی **انتخابی** کندتر از سایر الگوریتم‌ها هستند، به دلیل پیچیدگی زمانی

O(n^2)

مرتب‌سازی **ادغام** و مرتب‌سازی **پشته‌ای(هیپ سورت)** نیز عملکرد خوبی دارند، ولی ممکن است از نظر حافظه مصرفی بیشتر باشند.

مرتب‌سازی **رادیکس** به خصوص برای داده‌های عددی عملکرد خوبی دارد و به دلیل عدم وابستگی به مقایسه مستقیم، در برخی موارد می‌تواند سریع‌تر از دیگر الگوریتم‌ها باشد.

مرتب‌سازی **شل** به دلیل کاهش تدریجی فاصله‌ها و بهینه‌سازی مرتب‌سازی درج، عملکرد بهتری نسبت به مرتب‌سازی درج ساده دارد.

نتیجه‌گیری

این آزمایش نشان می‌دهد که انتخاب الگوریتم مرتب‌سازی مناسب بسته به نوع داده‌ها و حجم آن‌ها بسیار مهم است. الگوریتم‌های **سریع** و **ادغام** معمولاً برای مجموعه داده‌های بزرگ مناسب‌تر هستند، در حالی که برای داده‌های کوچکتر و ساده‌تر، الگوریتم‌های **درج** و **انتخابی** می‌توانند کافی باشند.

پیشنهادات

برای بهبود عملکرد برنامه، می‌توان از موازی‌سازی و استفاده از پردازش‌های چند هسته‌ای بهره برد. همچنین، استفاده از ساختارهای داده‌های پیشرفته‌تر و بهینه‌سازی کدهای موجود می‌تواند زمان اجرای الگوریتم‌ها را بهبود بخشد.

تهیه کنندگان:

آقایان بذرافشان و کیا | دانشگاه فسا(1403)