۱.مقایسه جستجوی خطی و دودویی برای آرایه‌های مرتب‌شده و مرتب‌نشده:

در یک آرایه نامرتب، تنها گزینه ممکن جستجوی خطی است، زیرا جستجوی باینری نیاز به آرایه مرتب دارد. در این حالت، جستجوی خطی در بدترین حالت باید کل آرایه را بررسی کند، بنابراین پیچیدگی زمانی آن $O(n)$ خواهد بود.

در یک آرایه مرتب، جستجوی باینری بسیار کارآمدتر از جستجوی خطی است. در حالی که جستجوی خطی همچنان $O(n)$ زمان می‌برد، جستجوی باینری $O(\log n)$ زمان می‌برد، زیرا در هر مرحله نیمی از داده‌ها حذف می‌شوند.

بهترین حالت: در آرایه‌های مرتب و نامرتب، بهترین حالت برای جستجوی خطی $O(1)$ است، زمانی که مقدار مورد نظر در اولین خانه قرار دارد. در جستجوی باینری نیز بهترین حالت $O(1)$ است، زمانی که مقدار در وسط آرایه باشد.

بدترین حالت: در یک آرایه نامرتب، جستجوی خطی در بدترین حالت $O(n)$ مقایسه انجام می‌دهد، زیرا ممکن است نیاز به بررسی تمام عناصر باشد. اما در یک آرایه مرتب، جستجوی باینری فقط $O(\log n)$ مقایسه نیاز دارد که باعث کاهش چشمگیر زمان جستجو می‌شود.

میانگین حالت: در یک آرایه نامرتب، جستجوی خطی به طور میانگین $O(n)$ زمان می‌برد، زیرا باید نیمی از عناصر را بررسی کند. اما در یک آرایه مرتب، جستجوی باینری به طور میانگین $O(\log n)$ زمان می‌برد، که بسیار سریع‌تر از جستجوی خطی است.

کاربرد در عمل: اگر آرایه از قبل مرتب باشد، جستجوی باینری بسیار کارآمدتر از جستجوی خطی خواهد بود. اما اگر آرایه نامرتب باشد و بخواهیم ابتدا آن را مرتب کنیم، مرتب‌سازی به $O(n \log n)$ زمان نیاز دارد، که ممکن است برای یک جستجوی واحد به‌صرفه نباشد. اما اگر تعداد زیادی جستجو در آرایه انجام شود، جستجوی باینری پس از مرتب‌سازی عملکرد بهتری خواهد داشت.

تحلیل پیچیدگی زمانی جستجوی ۵ عدد در یک آرایه ۲۲ عنصری:

تحلیل پیچیدگی جستجوی خطی:

بهترین حالت: $O(1)$

- اگر عدد مورد نظر در ابتدای آرایه باشد، فقط یک مقایسه لازم است.
- از آنجا که باید ۵ عدد را جستجو کنیم، در بهترین حالت فقط ۵ مقایسه انجام می‌شود.
- پیچیدگی برای یک جستجو: $O(1)$
- پیچیدگی برای ۵ جستجو: $O(5) \rightarrow$ در نماد Big-O همان $O(1)$ در نظر گرفته می‌شود.

بدترین حالت: $O(n)$

- اگر عدد در آرایه وجود نداشته باشد یا در آخرین موقعیت قرار داشته باشد، باید همه ۲۲ عنصر بررسی شوند.
- برای ۵ جستجو، در بدترین حالت باید ۲۲ بار در ۵ نوبت جستجو کنیم $5*22=110$ مقایسه انجام خواهد شد.
- پیچیدگی برای یک جستجو: $O(22)=O(n)$
- پیچیدگی برای ۵ جستجو: $O(5n) = O(n)$

میانگین حالت: $O(n)$

- به طور میانگین، یک عدد در موقعیت وسط آرایه پیدا می‌شود $(n/2 = 11$ مقایسه برای هر جستجو).
- برای ۵ جستجو: $5 \times 11 = 55$ مقایسه.
- پیچیدگی برای یک جستجو: $O(n)$
- پیچیدگی برای ۵ جستجو: $O(n)$

تحلیل پیچیدگی جستجوی باینری:

پیش‌نیاز: آرایه باید مرتب باشد

- اگر آرایه مرتب نباشد، نیاز به مرتب‌سازی داریم که $O(n\log n)$ زمان می‌برد.
- مرتب‌سازی یک آرایه ۲۲ عنصری حدود $O(100) \approx O(22\log 22)$ عملیات نیاز دارد.
- اما اگر آرایه از قبل مرتب باشد، این هزینه محاسبه نمی‌شود.

بهترین حالت: $O(1)$

- اگر عدد در میانه آرایه باشد، در همان اولین مقایسه پیدا می‌شود.
- برای ۵ جستجو، ۵ مقایسه انجام می‌شود.
- پیچیدگی برای یک جستجو: $O(1)$
- پیچیدگی برای ۵ جستجو: $O(1)$

بدترین حالت: $O(\log n)$

- جستجوی باینری در هر مرحله، نیمی از عناصر باقی‌مانده را حذف می‌کند.
- تعداد مراحل مورد نیاز برای جستجو در یک آرایه ۲۲ عنصری:
  $\log 22 \approx 4.5 \approx 5$ مقایسه برای هر جستجو
- برای ۵ جستجو: $5 \times 5 = 25$ مقایسه.
- پیچیدگی برای یک جستجو: $O(\log n)$
- پیچیدگی برای ۵ جستجو: $O(\log n)$

میانگین حالت: $O(\log n)$

- به طور میانگین، عدد مورد نظر در $O(\log n)$ مقایسه پیدا می‌شود.
- برای ۵ جستجو: $5 * 5 = 25$ مقایسه.
- پیچیدگی برای یک جستجو: $O(\log n)$
- پیچیدگی برای ۵ جستجو: $O(\log n)$

چرا جستجوی باینری بهتر از جستجوی خطی است؟

چون $O(\log n)$ خیلی کندتر از $O(n)$ رشد می‌کند، برای داده‌های حجیم، جستجوی باینری بسیار سریع‌تر است. در همین مثال، در بدترین حالت، جستجوی باینری فقط ۲۵ مقایسه انجام می‌دهد، در حالی که جستجوی خطی ۱۱۰ مقایسه نیاز دارد. حتی در بدترین حالت، جستجوی باینری بسیار سریع‌تر از جستجوی خطی است.

چه زمانی جستجوی خطی بهتر است؟

- اگر آرایه نامرتب باشد، جستجوی باینری امکان‌پذیر نیست مگر اینکه ابتدا مرتب‌سازی انجام شود که هزینه‌بر است.
- برای آرایه‌های کوچک، تفاوت بین $O(n)$ و $O(\log n)$ ناچیز است، بنابراین جستجوی خطی ساده‌تر است.

۲.جستجوی دودویی در آرایه‌های بزرگ و مقایسه پیچیدگی زمانی مرتب‌سازی:

جستجوی عدد ۵۵ در یک آرایه مرتب با طول ۲۰۰

چون آرایه از قبل مرتب شده است، بهترین روش برای یافتن عدد ۵۵، جستجوی باینری است که پیچیدگی زمانی $O(\log ۲۰۰) \approx O(۷)$ دارد. یعنی در بدترین حالت، فقط ۷ مقایسه نیاز خواهد بود تا مقدار ۵۵ پیدا شود یا عدم وجود آن تأیید گردد.

مقایسه‌ی مرتب‌سازی سریع و مرتب‌سازی ادغامی

اگر آرایه را دوباره با مرتب‌سازی سریع و مرتب‌سازی ادغامی مرتب کنیم، تفاوت‌های آن‌ها به شرح زیر است:

- مرتب‌سازی سریع (Quick Sort):
  - حالت میانگین و بهترین حالت: $O(n \log n)$
  - بدترین حالت: $O(n^۲)$ (اگر محور ضعیف انتخاب شود، مثلاً همیشه کوچک‌ترین یا بزرگ‌ترین عنصر انتخاب شود)
  - نیاز به فضای اضافی: کم (درجا انجام می‌شود)
  - مناسب برای: زمانی که آرایه تا حدی مرتب است یا فضای ذخیره‌سازی اهمیت دارد.
- مرتب‌سازی ادغامی (Merge Sort):
  - پیچیدگی زمانی: $O(n \log n)$ در همه‌ی حالات (بهترین، میانگین، و بدترین)
  - پیچیدگی فضایی: $O(n)$ (نیاز به فضای اضافی برای ادغام)
  - مناسب برای: زمانی که پایداری (Stable Sorting) مهم است یا بدترین حالت $O(n \log n)$ تضمین‌شده باشد.

برای ۲۰۰ عنصر، Quick Sort معمولاً سریع‌تر است، زیرا ثابت‌های زمانی کوچک‌تری دارد و درجا (In-Place) اجرا می‌شود، مگر اینکه بدترین حالت رخ دهد.

چه زمانی جستجوی خطی بهتر از جستجوی باینری است؟

جستجوی خطی در موارد زیر بهتر از جستجوی باینری است:

1. آرایه کوچک باشد: برای آرایه‌های خیلی کوچک (مثلاً ۵ تا ۱۰ عنصر)، سربار محاسباتی جستجوی باینری ممکن است بیشتر از یک جستجوی ساده‌ی خطی باشد.
2. آرایه نامرتب باشد: جستجوی باینری فقط در آرایه‌های مرتب کار می‌کند. اگر مرتب‌سازی لازم باشد، هزینه‌ی آن $O(n \log n)$ خواهد شد، در حالی که جستجوی خطی $O(n)$ است.
3. فقط یک جستجو در آرایه نامرتب انجام شود: اگر قرار باشد فقط یک بار مقدار خاصی جستجو شود، جستجوی خطی سریع‌تر خواهد بود چون دیگر نیازی به مرتب‌سازی نیست.
4. داده‌ها در یک لیست پیوندی ذخیره شده باشند: از آنجایی که لیست پیوندی (Linked List) امکان دسترسی تصادفی (Random Access) ندارد، جستجوی باینری عملکرد خوبی ندارد و جستجوی خطی تنها گزینه‌ی ممکن است.

مثال:

فرض کنید یک آرایه‌ی نامرتب با ۱۰,۰۰۰ عنصر داریم و باید فقط یک عدد خاص را پیدا کنیم.
اگر ابتدا مرتب کنیم، پیچیدگی زمانی برابر $O(۱۰,۰۰۰ \log ۱۰,۰۰۰) \approx O(۱۳۰,۰۰۰)$ خواهد شد،
در حالی که جستجوی خطی در بدترین حالت حداکثر ۱۰,۰۰۰ مقایسه انجام می‌دهد.
در این حالت، جستجوی خطی بهتر است زیرا فقط یک بار جستجو انجام می‌شود و نیازی به مرتب‌سازی نداریم.

۳.مرتب‌سازی درجی:

الگوریتم مرتب‌سازی درج با انتخاب عناصر یک‌به‌یک و قرار دادن آن‌ها در موقعیت صحیح خود، کار می‌کند.

۱. حالت بدترین (O(n^۲

- این حالت زمانی رخ می‌دهد که آرایه به‌صورت نزولی مرتب شده باشد و نیاز به تغییر کامل ترتیب داشته باشد.
- هر عنصر باید با تمام عناصر قبلی مقایسه و جابه‌جا شود، که تعداد مقایسه‌ها و جابه‌جایی‌ها برابر است با: n(n-۱)/۲
- برای n=۵۰: ۴۹×۵۰۲=۱۲۲۵
  - پیچیدگی زمانی: O(n^۲)، یعنی حدود ۲۵۰۰ عملیات در بدترین حالت برای n=۵۰.

۲. حالت میانگین (O(n^۲

- در حالت میانگین، عناصر در موقعیت‌های تصادفی قرار دارند و باید در وسط آرایه درج شوند.
- به‌طور متوسط، نیمی از عناصر باید مقایسه و جابه‌جا شوند.
- تعداد عملیات همچنان به‌صورت درجه دوم (O(n^۲)) رشد می‌کند و تقریباً برابر است با: (n^۲)/۴
  - برای n=۵۰، حدود ۶۲۵ عملیات انجام می‌شود.
  - پیچیدگی زمانی: O(n^۲).

۳. حالت بهترین (O(n))

- این حالت زمانی رخ می‌دهد که آرایه از قبل مرتب باشد.
- هر عنصر فقط یک مقایسه انجام می‌دهد و جابجایی ندارد.
- تعداد عملیات برابر است با: n−۱
  - برای n=۵۰، فقط ۴۹ عملیات انجام می‌شود.
  - پیچیدگی زمانی: O(n)

مقایسه آرایه و لیست پیوندی برای بهینه‌سازی مصرف حافظه

استفاده از آرایه موجب بهینه‌تر شدن حافظه می‌شود، زیرا:

1. اختصاص حافظه پیوسته (Contiguous Memory Allocation): نیازی به ذخیره اشاره‌گرهای اضافی (مثل لیست پیوندی) نیست.
2. دسترسی سریع‌تر به داده‌ها: به دلیل حفظ محلیّت کش (Cache Locality)، پردازنده می‌تواند داده‌ها را سریع‌تر بارگذاری کند.
3. کاهش سربار حافظه: لیست پیوندی نیاز به حافظه اضافی برای ذخیره اشاره‌گرها دارد (به‌ویژه در لیست‌های پیوندی دوطرفه که دو برابر حافظه نیاز دارند).

چرا از لیست پیوندی برای مرتب‌سازی درج استفاده نکنیم؟

- اگرچه درج در لیست پیوندی O(۱) است، اما جستجو برای موقعیت مناسب هنوز O(n) زمان می‌برد، که در مجموع پیچیدگی را همچنان O(n^۲) نگه می‌دارد.
- عدم کارایی کش (Cache Inefficiency): لیست‌های پیوندی باعث افزایش خطاهای کش (Cache Misses) می‌شوند، که عملکرد را در عمل کندتر می‌کند.

نتیجه‌گیری:

برای بهینه‌سازی حافظه و عملکرد بهتر، آرایه نسبت به لیست پیوندی در مرتب‌سازی درج گزینه بهتری است.