مقایسه الگوریتم‌های مرتب‌سازی

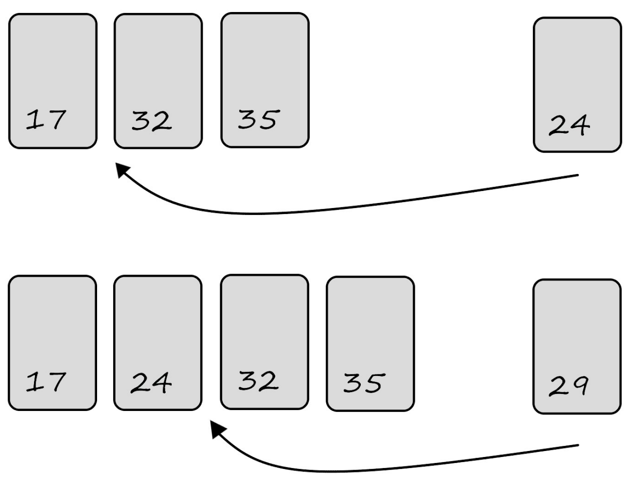
محمد ترابی - علی جعفرآبادی - رضا تاج‌گذاری

تیرماه ۱۴۰۲

۱ مرتب‌سازی درجی[1]

اگر یک دسته کارت به شما داده شود که اعداد ۱ تا ۵۰ روی آن نوشته شده است، چگونه آن را مرتب [2] می‌کنید؟ احتمالا اول تعداد کمی کارت برمی‌دارید و آن را مرتب می‌کنید؛ سپس بقیه کارت‌ها را یکی پس از دیگری نگاه می‌کنید و در جای مناسب میان کارت‌های مرتب شده قرار می‌دهید. شکل ۱ نمایی کلی از این روش مرتب‌سازی نشان می‌دهد.

وقتی کارت‌ها را با این روند مرتب می‌کنیم، همواره تعدادی از کارت‌ها مرتب شده است و کارت‌هایی که هنوز مرتب نشده، یکی پس از دیگری در دستهٔ کارت‌های مرتب شده درج [3] می‌شوند. اگر با این روش کارت‌ها را مرتب کنیم، درواقع از مرتب‌سازی درجی استفاده کرده ایم.



شکل ۱: مرتب کردن کارت‌ها به کمک مرتب‌سازی درجی

[1] insertion sort
[2] sort
[3] insert

الگوریتم مرتب‌سازی درجی

آرایه‌ای به طول یک همواره مرتب است؛ بنابراین از عنصر دوم شروع می‌کنیم و جلو می‌رویم. فرض کنیم که به عنصر i رسیده‌ایم، عناصر قبل از i مرتب‌اند. لذا از آخرین عنصر قبل از i شروع می‌کنیم و هر یک از عناصر بزرگتر از i را یک واحد به سمت راست انتقال [4] می‌دهیم. وقتی به عنصری کوچک تر از i یا به ابتدای آرایه رسیدیم متوقف می‌شویم و i را همانجا درج می‌کنیم. شبه‌کد مرتب‌سازی درجی را می‌توانید در الگوریتم ۱ که در ادامه آمده است مشاهده کنید [5] .

الگوریتم ۱ مرتب‌سازی درجی

```
procedure InsertionSort(arr, n)
    for i ← 1 to n − 1 do
        key ← arr[i]
        j ← i − 1
        while j >= 0 and arr[j] > key do
            arr[j + 1] ← arr[j]
            j ← j − 1
        arr[j + 1] ← key
```

اگر الگوریتم بالا را روی آرایهٔ $[۷, ۶, ۶, ۴, ۹]$ اجرا کنیم، آرایه بدین صورت مرتب می‌شود:

$$[۷,۶,۶,۴,۹] \to [۷,۷,۶,۴,۹] \to [۶,۷,۶,۴,۹] \to [۶,۷,۷,۴,۹] \to [۶,۶,۷,۴,۹] \to [۶,۶,۷,۷,۹] \to [۶,۶,۶,۷,۹] \to [۶,۶,۶,۷,۹] \to [۴,۶,۶,۷,۹] \to [۴,۶,۶,۷,۹]$$

توجه داشته باشید که پیچیدگی زمانی مرتب‌سازی درجی در بهترین حالت خطی است. همچنین بهترین حالت وقتی اتفاق می‌افتد که آرایه مرتب باشد. می‌توانیم نتیجه بگیریم که در مواقعی که آرایه مرتب و یا تقریبا مرتب است مرتب‌سازی درجی بسیار سریع عمل می‌کند. بنابراین اگر از قبل مطلع هستیم که معمولا داده‌های ما تقریبا مرتب است، استفاده از مرتب‌سازی درجی می‌تواند گزینه مناسبی باشد.

[4] shift

[5] توجه داشته باشید که الگوریتم بر پایه صفر است؛ یعنی عنصر اول آرایه $arr[۰]$ می‌باشد.

به عنوان مثال فرض کنید که یک لیست هزارتایی از اسامی دانشجویان یک موسسه در اختیار دارید که به ترتیب حروف الفبا مرتب شده اند. در سال جدید پنجاه دانشجو در موسسه نام نویسی می‌کنند و اسامی آن‌ها به آخر آرایه اضافه می‌شود. اگرچه روش‌های زیادی برای مرتب کردن لیست جدید دانشجویان وجود دارد، مرتب‌سازی درجی گزینه مناسبی محسوب می‌شود و از دیگر روش‌های مرتب‌سازی که در این مقاله توضیح داده شده، بهتر عمل می‌کند.

خوب است بدانیم می‌توان پیچیدگی زمانی مرتب سازی درجی را با تغییرات کمی بهبود بخشید که در این مقاله به آن نمی‌پردازیم. به عنوان مثال مرتب‌سازی شِل [۶] یک روش مرتب سازی است که از مرتب‌سازی درجی الهام گرفته شده و از نظر پیچیدگی زمانی از مرتب‌سازی درجی بهتر عمل می‌کند. [۱] [۲]

مراجع

[1] R. B. Frank, R. M.; Lazarus, "A high-speed sorting procedure," *Communications of the ACM*, p.20–22, 1960.

[2] R. Sedgewick, "A new upper bound for shellsort," *Journal of Algorithms*, p.159–173, 1986.

[6] shell sort