یادآوری جلسه نوزدهم — حد پایین مرتب‌سازی مقایسه‌ای- مرتب‌سازی خطی — مهدی داودزاده

در جلسه گذشته در مورد مرتب سازی سریع تصادفی بحث کردیم. همچنین حد پایینی برای مرتب‌سازی مقایسه‌ای تعیین کرده و در انتها بحث در مورد مرتب‌سازی خطی را آغاز کردیم.

مرتب‌سازی سریع تصادفی. مرتب‌سازی سریع تصادفی همان مرتب‌سازی سریع است که در آن عنصر محور به صورت تصادفی انتخاب می‌شود. می‌خواهیم نشان دهیم تعداد مقایسه‌های صورت گرفته در مرتب‌سازی تصادفی $O(n \log n)$ است. فرض کنید اعدادی که قرار است مرتب شوند از ۱ تا n هستند. در این‌صورت X_{ij} را به صورت زیر تعریف می‌کنیم:

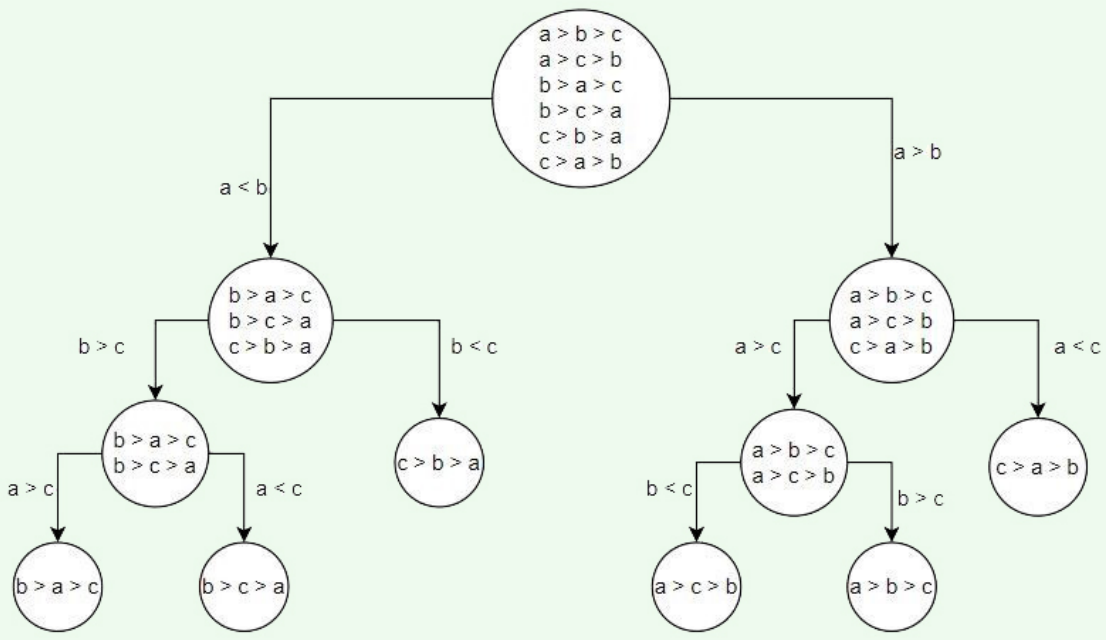
$$X_{ij} = \begin{cases} ۱ : i \text{ و } j \text{ مقایسه شوند} \\ ۰ : i \text{ و } j \text{ مقایسه نشوند} \end{cases}$$

از آنجایی که همیشه عناصر صرفا با محور مقایسه می‌شوند، احتمال این که عدد i با j مقایسه شود معادل این است که i یا j اولین محور انتخاب شده در بازه $[i, j]$ باشد. بنابراین متوسط مقایسه‌ها که برابر $\sum_{i,j} X_{ij}$ می‌باشد به صورت زیر می‌باشد:

$$E(\sum_{i,j} X_{ij}) = \underbrace{\sum_{i=۱}^{n} \underbrace{\sum_{k=۱}^{n-i} \frac{۲}{k+۱}}_{O(\log n)}}_{O(n \log n)}$$

لذا اگر عنصر محور را به صورت تصادفی انتخاب کنیم حالت متوسط اجرای این الگوریتم از مرتبه $O(n \log n)$ می‌باشد.

حد پایین برای مرتب‌سازی مقایسه‌ای. هر الگوریتم مقایسه ای در بدترین حالت یا حالت متوسط، به $\Omega(n \log n)$ مقایسه نیاز دارد. برای این کار از درخت تصمیم استفاده می‌کنیم. در درخت تصمیم در هر راس تمام جایگشت‌های ممکن بر اساس اطلاعاتی که داریم مشخص شده است. طبعا این مقایسه‌ها باید تا جایی ادامه یابد که به یک جایگشت برسیم.



در درخت تصمیم هر یک از برگ‌های آن یک جایگشت از عناصر آرایه هستند، بنابراین تعداد برگ ها $n!$ است و در نتیجه ارتفاع درخت از $\log(n!) = \Omega(n \log n)$ می‌باشد.

مرتب‌سازی خطی. در ادامه بحث مرتب‌سازی خطی را آغاز کردیم. در اولین مرحله **مرتب سازی شمارشی** را مورد بررسی قرار دادیم:

در این مرتب سازی n عدد را به عنوان ورودی دریافت کرده و می دانیم که همه اعداد کوچکتر از m هستند. آرایه B به طول m را در نظر می‌گیریم. ابتدا تمامی درایه‌های آرایه B برابر صفر هستند، سپس در هر مرحله هنگامی که به عدد i می‌رسیم مقدار خانه $B[i]$ را یک واحد افزایش می‌دهیم. برای به دست آوردن آرایه مرتب‌شده آرایه B را پیمایش کرده و به تعداد $B[i]$ عنصر i را چاپ می‌کنیم. این الگوریتم از $O(n+m)$ می‌باشد.

مشکل الگوریتم بالا این است که برای اعداد شخصیت قائل نیستیم و الگوریتم مرتب‌سازی پایدار نیست. برای رفع این مشکل هر خانه آرایه B را با خانه پیش از خود جمع می‌کنیم. سپس آرایه A را از راست به چپ پیمایش می‌کنیم، به ازای هر عدد i، مقدار خانه $B[i]$ را به عنوان مکان آن عدد در آرایه مرتب شده در نظر می‌گیریم و مقدار $B[i]$ را یک واحد کاهش می‌دهیم.