به نام خدا

تمرین سوم

مدرسه:

فیلاگر

مؤلف:

سعید قادری‌زاده

زمستان ۱۴۰۲

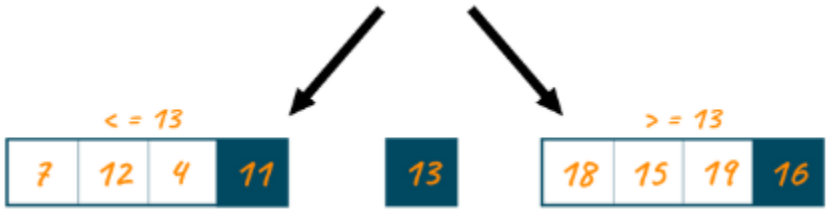
الگوریتم مرتب سازی سریع[1]

الگوریتم مرتب‌سازی سریع یکی از رایج‌ترین روش‌های مرتب کردن داده‌ها در علم کامپیوتر است که به‌دلیل مصرف حافظه کم، سرعت اجرای مناسب و پیاده‌سازی ساده در بسیاری از برنامه ها کاربرد دارد. این الگوریتم از روش تقسیم و غلبه[2] برای مرتب سازی عناصر آرایه استفاده می‌کند. به عبارت دیگر، داده‌ها را به بخش‌های کوچک‌تری تقسیم کرده و هر بخش را به صورت جداگانه مرتب می‌کند.

مراحل الگوریتم:

۱. انتخاب عنصر محوری[3]: اولین قدم انتخاب یک عنصر به عنوان عنصر محوری از آرایه است. این عنصر به عنوان معیاری برای مقایسه‌ی سایر عناصر قرار می‌گیرد که می تواند به طور تصادفی، اولین، آخرین یا عنصر وسط آرایه باشد.

۲. تقسیم آرایه: با استفاده از عنصر محوری، آرایه به دو بخش تقسیم می شود:
   - بخش چپ: شامل تمام عناصری است که کوچکتر از عنصر محوری هستند.
   - بخش راست: شامل تمام عناصری است که بزرگتر یا مساوی عنصر محوری هستند.



۳. مرتب سازی بازگشتی: الگوریتم مرتب سازی سریع به طور بازگشتی روی بخش های چپ و راست اعمال می‌شود. این فرآیند تا زمانی ادامه می‌یابد که هر دو بخش به طور کامل مرتب شوند.

[1] Quick Sort
[2] Divide and Conquer
[3] pivot

مزایای الگوریتم مرتب سازی سریع:

- زمان: پیچیدگی زمانی متوسط الگوریتم برابر $O(n \log n)$ است که به مراتب بهتر از الگوریتم های مرتب سازی با پیچیدگی $O(n^2)$ است.
- سادگی پیاده سازی: پیاده سازی الگوریتم نسبت به برخی الگوریتم‌های مرتب سازی پیچیده مانند مرتب سازی ادغام و هرمی ساده‌تر است.
- عملکرد بسیار خوب در مواقعی که آرایه‌ها به صورت تصادفی مرتب نشده باشند.

معایب الگوریتم مرتب سازی سریع:

- زمان: در بدترین حالت، پیچیدگی زمانی الگوریتم به $O(n^2)$ می رسد. این حالت زمانی رخ می دهد که آرایه تقریباً مرتب یا تقریباً به طور معکوس مرتب شده باشد.
- حافظه: پیچیدگی فضایی الگوریتم$O(nLogn)$ است، به دلیل استفاده از بازگشت، فضای حافظه بیشتری نسبت به برخی الگوریتم های مرتب سازی دیگر مصرف می‌کند. نیاز به حافظه‌ی اضافی برای ذخیره‌سازی زیرآرایه‌ها دارد.
- ناپایداری: الگوریتم ناپایدار است، به این معنی که ترتیب عناصر با مقادیر مساوی در آرایه اصلی پس از مرتب سازی حفظ نمی شود.
- محور: انتخاب محور تاثیر زیادی بر کارایی الگوریتم دارد. انتخاب محور به صورت تصادفی معمولا باعث می‌شود در بدترین حالت قرار نگیریم.

الگوریتم مرتب سازی هرمی[۴]

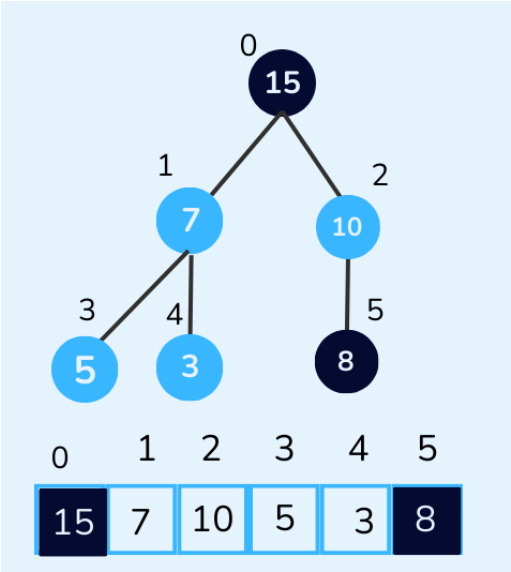
الگوریتم مرتب سازی هرمی، یک الگوریتم مرتب سازی مبتنی بر ساختار داده هرم دودویی[۵] است. این الگوریتم مشابه با مرتب سازی انتخابی[۶] عمل می‌کند، با این تفاوت که به جای جستجوی خطی برای یافتن بزرگترین عنصر، از ساختار داده هرم برای یافتن سریعتر بزرگترین عنصر استفاده می‌کند. به این صورت که عنصر بزرگتر را پیدا می‌کند و آن را در انتهای آرایه قرار می‌دهد. سپس این فرآیند برای عناصر باقی مانده تکرار می‌شود تا آرایه به طور کامل مرتب شود.

مراحل الگوریتم:

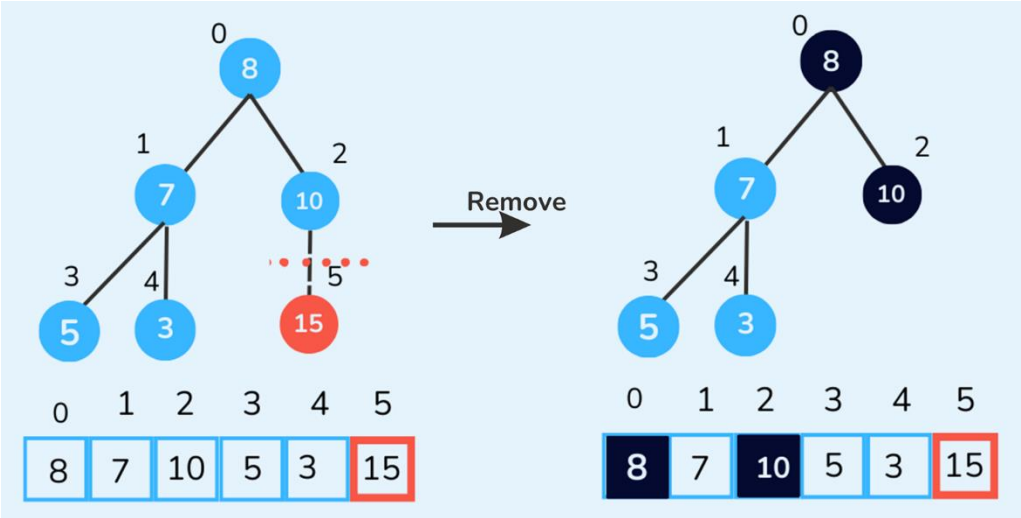
۱. ایجاد هرم: ابتدا آرایه ورودی را به یک هرم دودویی تبدیل می‌کنیم. در هرم دودویی، هر گره باید از هر دو فرزند خود بزرگتر باشد. سپس با مقایسه عناصر و جابجایی آن‌ها، هرم به گونه ای بازسازی می شود که بزرگترین عنصر در بالای هرم قرار گیرد. برای ساخت هرم می توان از الگوریتم های مختلفی مانند heapify استفاده کرد.

---

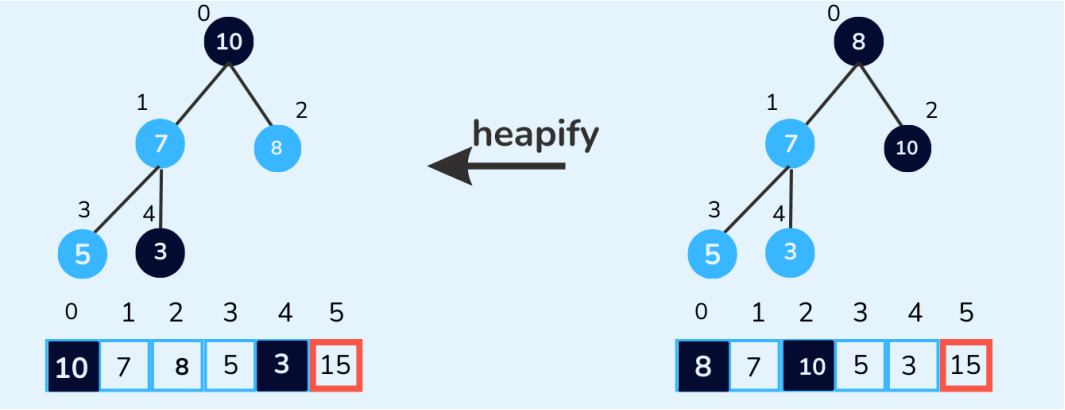
[۴] Heap Sort
[۵] Binary Heap
[۶] Selection Sort



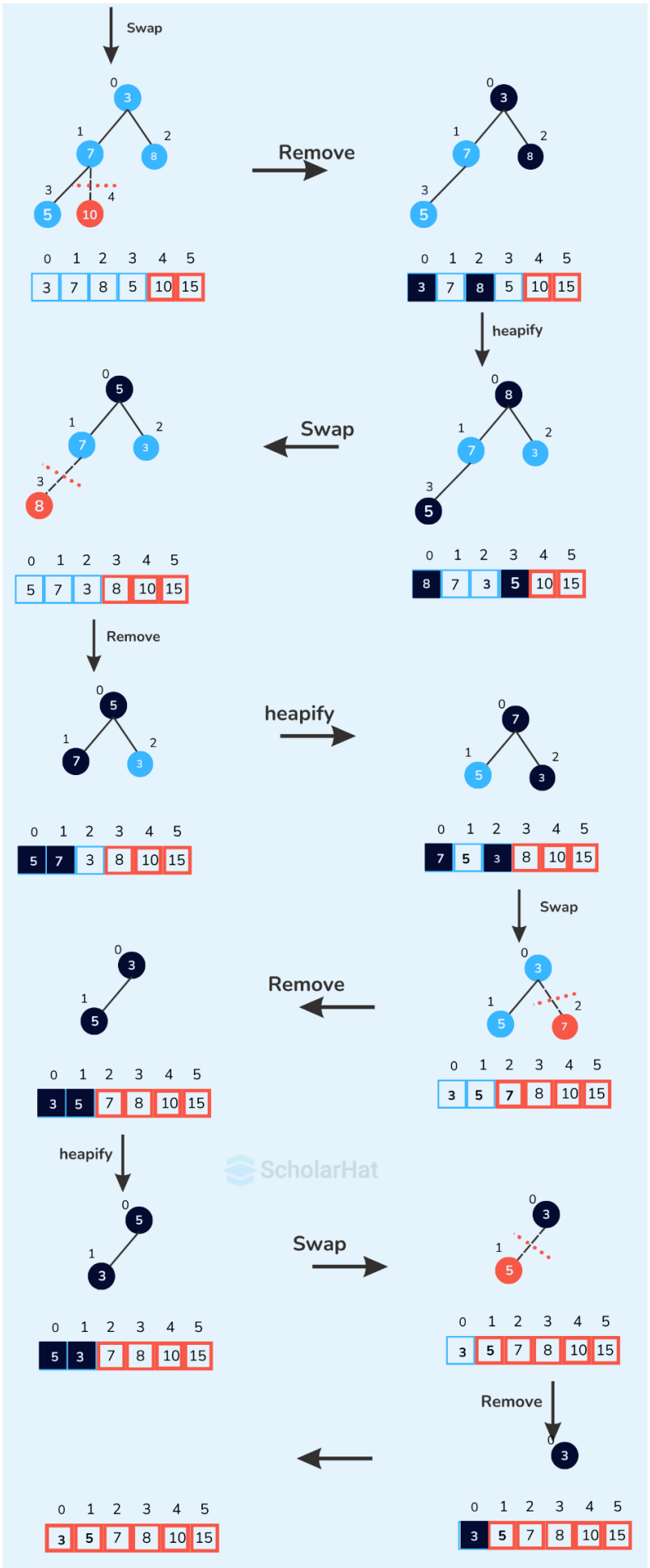
۲. حذف بزرگترین عنصر: بزرگترین عنصر (ریشه هرم) از هرم حذف می شود و در انتهای آرایه قرار می‌گیرد. بزرگترین عنصر که در ریشه هرم قرار دارد را با آخرین عنصر هرم جابجا می‌شود. اندازه هرم را یک واحد کاهش می‌دهیم.



۳. بازسازی هرم: هرم باقیمانده مجدداً بازسازی می شود تا بزرگترین عنصر باقی مانده در بالای هرم قرار گیرد.



۴. تکرار مراحل: مراحل ۲ و ۳ تا زمانی که فقط یک عنصر در هرم باقی بماند، تکرار می شود.



مزایای الگوریتم مرتب سازی هرمی:

- پیچیدگی زمانی: پیچیدگی زمانی متوسط الگوریتم برابر $O(n \log n)$ است که به مراتب بهتر از الگوریتم های مرتب سازی با پیچیدگی $O(n^2)$ است.
- مناسب برای آرایه های بزرگ: الگوریتم برای مرتب سازی آرایه های بزرگ بسیار کارآمد است.

معایب الگوریتم مرتب سازی هرمی:

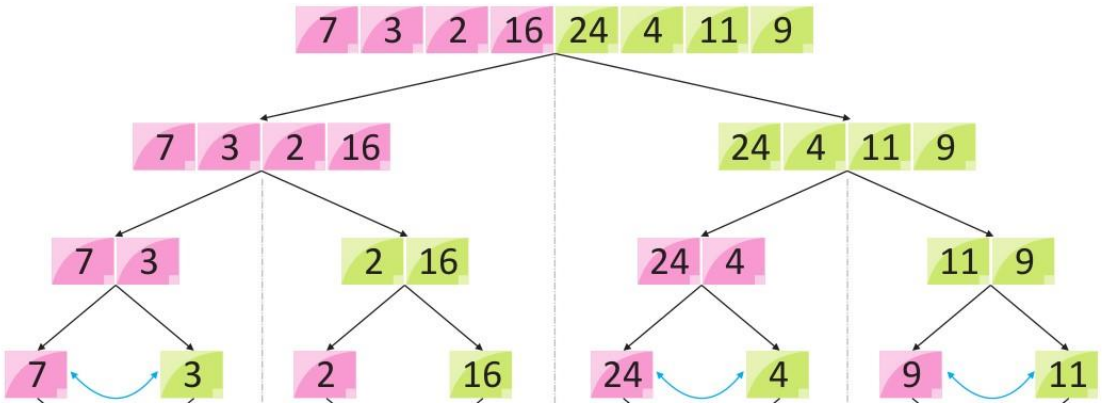
- حافظه: الگوریتم Heap Sort به دلیل استفاده از ساختار داده هرم، فضای حافظه بیشتری نسبت به برخی الگوریتم های مرتب سازی دیگر مانند مرتب سازی حبابی[۷] مصرف می کند. مصرف حافظه بیشتر به دلیل استفاده از ساختار داده هرم.
- ناپایداری: الگوریتم Heap Sort ناپایدار است، به این معنی که ترتیب عناصر با مقادیر مساوی در آرایه اصلی پس از مرتب سازی حفظ نمی شود.

الگوریتم مرتب سازی ادغامی[۸]

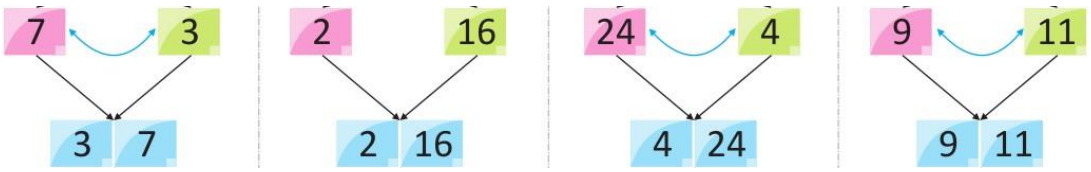
الگوریتم مرتب سازی ادغامی، یکی از الگوریتم‌های محبوب مرتب سازی است که به دلیل پایداری و زمان اجرای $O(n\ log\ n)$ در بسیاری از برنامه ها کاربرد دارد. این الگوریتم از روش تقسیم و غلبه برای مرتب‌سازی عناصر آرایه استفاده می‌کند. به این ترتیب که داده‌ها را به بخش‌های کوچک‌تر تقسیم، هر بخش را به طور جداگانه مرتب و سپس بخش‌های مرتب شده را ادغام می‌کند.

مراحل الگوریتم:

۱. تقسیم آرایه: آرایه به طور بازگشتی به دو بخش مساوی تقسیم می‌شود تا زمانی که هر بخش فقط یک عنصر داشته باشد.

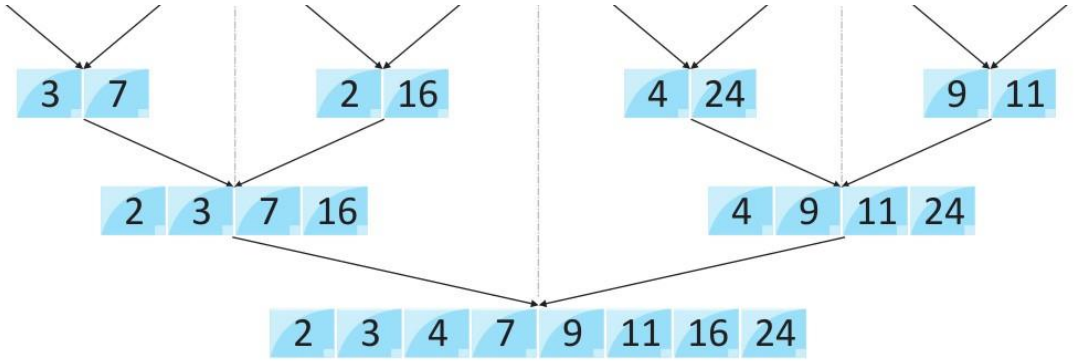


۲. مرتب سازی: هر آرایه تک‌عنصری، به طور پیش‌فرض مرتب شده است. الگوریتم مرتب‌سازی ادغامی به طور بازگشتی برای مرتب‌سازی هر بخش تقسیم‌شده به کار گرفته می‌شود. در هر مرحله، عناصر دو آرایه با هم مقایسه می‌شوند. عنصر کوچکتر در آرایه جدیدی که به عنوان آرایه مرتب‌شده عمل می‌کند، قرار می‌گیرد.



[۷] Bubble Sort
[۸] Merge Sort

۳. مقایسه و ادغام: دو آرایه مرتب شده با هم مقایسه می‌شوند. عنصر کوچکتر از هر دو آرایه انتخاب و در لیست نهایی مرتب شده قرار داده می‌شود. این فرآیند تا زمانی که هر دو لیست خالی شوند، ادامه می‌یابد. اگر عناصری در یکی از آرایه‌ها باقی مانده باشند، به انتهای آرایه نهایی مرتب‌شده اضافه می‌شوند.



مزایای الگوریتم مرتب سازی ادغامی:

- پایداری: الگوریتم Merge Sort پایدار است، به این معنی که ترتیب عناصر با مقادیر مساوی در آرایه اصلی پس از مرتب سازی حفظ می‌شود. پیاده‌سازی مستقل از نوع داده و پایدار است.
- زمان: پیچیدگی زمانی الگوریتم در هر حالت O(n log n) است، به این معنی که سرعت آن به طور قابل توجهی از الگوریتم های مرتب سازی با پیچیدگی $O(n^2)$ بهتر است.

معایب الگوریتم مرتب سازی ادغامی:

- مصرف حافظه: الگوریتم Merge Sort به دلیل نیاز به فضای ذخیره سازی برای دو بخش تقسیم شده آرایه، فضای حافظه بیشتری نسبت به برخی الگوریتم های مرتب سازی مصرف می‌کند.
- غیر بهینه برای آرایه های کوچک: الگوریتم Merge Sort برای مرتب سازی آرایه های کوچک بهینه نیست و در این حالت، الگوریتم های مرتب سازی حبابی و انتخابی سریعتر عمل می‌کنند.

مقایسه الگوریتم های مرتب سازی:

معیار / الگوریتم	پیچیدگی زمانی (بهترین حالت)	پیچیدگی زمانی (بدترین حالت)	حافظه	پایداری	پیاده سازی
Quick Sort	$O(n \log n)$	$O(n^2)$	$O(n \log n)$	ناپایدار	ساده
Merge Sort	$O(n \log n)$	$O(n \log n)$	$O(n)$	پایدار	متوسط
Heap Sort	$O(n \log n)$	$O(n \log n)$	$O(n)$	ناپایدار	متوسط

انتخاب الگوریتم مناسب:

انتخاب الگوریتم مرتب سازی مناسب به نیازهای شما بستگی دارد. اگر به دنبال الگوریتمی با سرعت بالا و حافظه کم هستید، Quick Sortانتخاب مناسبی است. اما اگر به دنبال الگوریتمی با پایداری و پیچیدگی زمانی تضمین شده هستید، Merge Sortیا Heap Sort انتخاب های بهتری هستند.

منابع:

- https://en.wikipedia.org/wiki/Quicksort
- https://www.geeksforgeeks.org/quick-sort/
- https://favtutor.com/blogs/quick-sort-cpp
- https://en.wikipedia.org/wiki/Heapsort
- https://www.geeksforgeeks.org/heap-sort/
- https://www.programiz.com/dsa/heap-sort
- https://en.wikipedia.org/wiki/Merge_sort
- https://www.geeksforgeeks.org/merge-sort/
- https://www.programiz.com/dsa/merge-sort