**الگوریتم تقسیم و حل**

یکی از الگوریتم‌های پرکاربرد در زمینه برنامه‌نویسی و همچنین حوزه‌های دیگر «الگوریتم‌های تقسیم و حل» (Divide and Conquer Algorithms) هستند. در این نوع الگوریتم تنها کاری که انجام می‌دهیم این است که مسائل را به مسئله‌های کوچک‌تر تجزیه می‌کنیم. از این نظر این الگوریتم‌ها شبیه روابط بازگشتی هستند؛ اما باید بدانیم که کجا باید این فرایند تقسیم کردن را متوقف کنیم. فرض کنید 8 عدد زیر را در اختیار داریم .



و می‌خواهیم همه آن‌ها را با هم جمع کنیم. ما این مسئله را به 8 بخش تقسیم می‌کنیم که هر یک شامل یک مورد از این اعداد است :



سپس شروع به جمع زدن دو به دو اعداد می‌کنیم، در این حالت ما 4 جفت عدد به صورت زیر داریم :

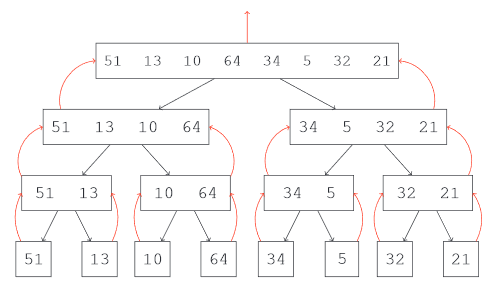


این فرایند را به همین ترتیب ادامه می‌دهیم .

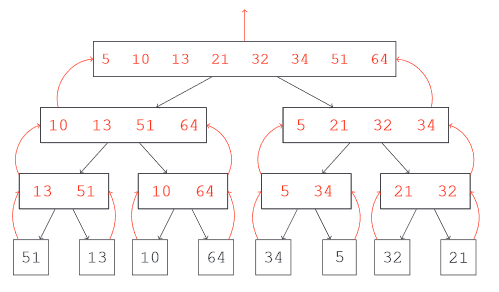
**مرتب‌سازی ادغامی**

مرتب‌سازی ادغامی نوعی از الگوریتم‌های مرتب‌سازی (sort) است. طرز کار این الگوریتم تقریباً به صورت زیر است :

* توالی n عدد را به دو نیمه تقسیم می‌کنیم.
* به طور بازگشتی دو نیمه را مرتب می‌کنیم.
* دو نیمه مرتب شده را دریک توالی مرتب منفرد ادغام می‌کنیم.



حل شده :



کد پایتونی :

|  |
| --- |
| def merge\_sort(arr): |
|  | middle\_len = len(arr) // 2 |
|  |  |
|  | if len(arr) > 1: |
|  | left = arr[:middle\_len] |
|  | right = arr[middle\_len:] |
|  |  |
|  | left = merge\_sort(left) |
|  | right = merge\_sort(right) |
|  | arr = [] |
|  | print("left: ", left) |
|  | print("right: ", right) |
|  | while len(left) > 0 and len(right) > 0: |
|  | if left[0] > right[0]: |
|  | arr.append(right[0]) |
|  | right.pop(0) |
|  | else: |
|  | arr.append(left[0]) |
|  | left.pop(0) |
|  | for elm in left: |
|  | arr.append(elm) |
|  | for elm in right: |
|  | arr.append(elm) |
|  |  |
|  | return arr |
|  |
|  |
|  | arr = [3, 5, 9, 1, 0, 10, 78, 125, -1, 0, 18] |
|  | print(arr) |
|  | print(merge\_sort(arr)) |

منبع کد : https://github.com/Teghfo/DjangoPaeez99/blob/main/W2/merge\_sort.py

گرد آورنده : محسن مرندی