ساختمان داده‌ها (۲۲۸۲۲)

مدرس: حسین بومری

[زمستان ۹۹]

سوال ۸: لیست‌های مرتب شده

نگارنده: آئیریا محمدی

۱ـ بر روی عضو اول هر دو لیست یک اشاره‌گر در نظر بگیرید. در هر مرحله از اجرای الگوریتم مقادیری که این دو به آن‌ها اشاره می‌کنند را با هم مقایسه می‌کنیم و هر کدام که بزرگ تر بود اشاره‌گرش را به محل جدید می‌بریم (یک واحد به راست). هرگاه یکی از اشاره‌گر ها به انتها رسید تمام اعضای باقی مانده لیست دیگر را پشت هم می‌نویسیم.

بدترین حالت وقتی رخ می‌دهد که هر دو اشاره‌گر در یک نوبت به انتها برسند. یعنی $i = n - ۱$ و $j = n$.

در هر دور از اجرای الگوریتم تا وقتی که هیچ کدام از اشاره‌گر ها به انتها نرسیده است باید برای تشخیص عضوی که باید در آرایه جدید بنویسیم یک مقایسه انجام بدهیم. این کار تا وقتی انجام می‌شود که بالاخره یکی از اشاره‌گر ها به انتها برسد. سپس هرچه باقی مانده چیده می‌شود. در بدترین حالت تنها یک عضو (عضو اخر آرایه دیگر) است که بدون نیاز به مقایسه گذاشته می‌شود. به عبارتی $۲n - ۱$ مقایسه انجام شده است.

سودوکد الگوریتم مطرح شده در زیر آمده است.

```
MERGE(A, B, n)
 1  i, j, x : int = 0
 2  B : List[int] of size 2n
 3  while i < n or j < n
 4      if i = n
 5          B[x] = A[j]
 6          j = j+1
 7      else if j = n
 8          B[x] = A[i]
 9          i = i+1
10      else if A[i] < A[j]
11          B[x] = A[i]
12          i = i + 1
13      else
14          B[x] = A[j]
15          j = j+1
16      x = x+1
```

۲ـ برای ادغام کردن k لیست کافی است آن‌ها را دو به دو با هم ادغام کنیم. سپس لیست‌های جدید را دوباره با هم دو به دو ادغام کنیم تا وقتی که تنها یک لیست داشته باشیم.

اگر k زوج باشد مرتبه زمانی الگوریتم مطرح شده به شکل زیر خواهد بود:

$$T(nk) = T(nk/۲) + O(۲\frac{nk}{۲} - ۱)$$

رابطه بالا از آن‌جا ناشی می‌شود که الگوریتم ما در عمل مانند این است که خود k دسته را ابتدا دو گروه کنیم و این الگوریتم را بر روی هر گروه اعمال کنیم و سپس دو ابرلیستی که تشکیل می‌شود را ادغام کنیم.

بدون ایجاد اشکال در جواب می‌توانیم فرض کنیم که k به شکل ۲^m است. اگر این چنین نباشد ($۲^{m-۱} < k < ۲^m$) کافی است که $۲^m - k$ لیست با تعداد n عضو با مقدار مثلا منفی بی‌نهایت اضافه کنیم و در نهایت که به پاسخ رسیدیم تمام این عضوهای زائد را از چپ حذف کنیم (در $O(nk)$)

و محاسبه می‌شود:

$$T(nk) = O(nk \cdot log(nk))$$

۳ـ فرض کنیم زمان آن قابل کاهش باشد. به عبارتی $T(nk) = o(nk \cdot log(nk))$.

کافی است قرار دهیم $n = ۱$. خواهیم داشت:

$$T(k) = o(k \cdot logk)$$

این به این معنا است ما می‌توانیم برای مرتب کردن n عدد به این شکل عمل کنیم که در $O(n)$ هر عدد را در یک لیست خالی قرار دهیم و سپس این n لیست بدیهتا مرتب را با هم در $o(nlogn)$ ادغام کنیم. در حالی که می‌دانیم همچین چیزی نمی‌تواند ممکن باشد که الگوریتم سورت با زمان قطعی کمتر از $O(nlogn)$ داشت پس فرض غلط است.