Merge Sort

مرتب‌سازی به روش ادغامی

الگوریتم مرتب‌سازی ادغامی در رده گروهی از الگوریتم‌ها که با عنوان "تقسیم و غلبه" شناخته می‌شوند قرار می‌گیرد

Divide and Conquer

یکی از نگرش‌های پایه‌ای در مبحث طراحی الگوریتم

نگرش تقسیم و غلبه

"فرا الگوریتم"

یعنی الگوریتمی که یک طراح الگوریتم (انسان) کمک می‌کند تا یک الگوریتم برای حل یک مسئله ابداع کند

۱) مسئله اصلی را P بنامید.

مسئله P را به تعدادی زیرمسئله (subproblem)

بشکنید: $P_1, P_2, \dots, P_k$

گام تقسیم

Divide

۲) هر یک از زیرمسئله‌های P_i را الف) اگر به‌قدر کافی "ساده" است ad-hoc

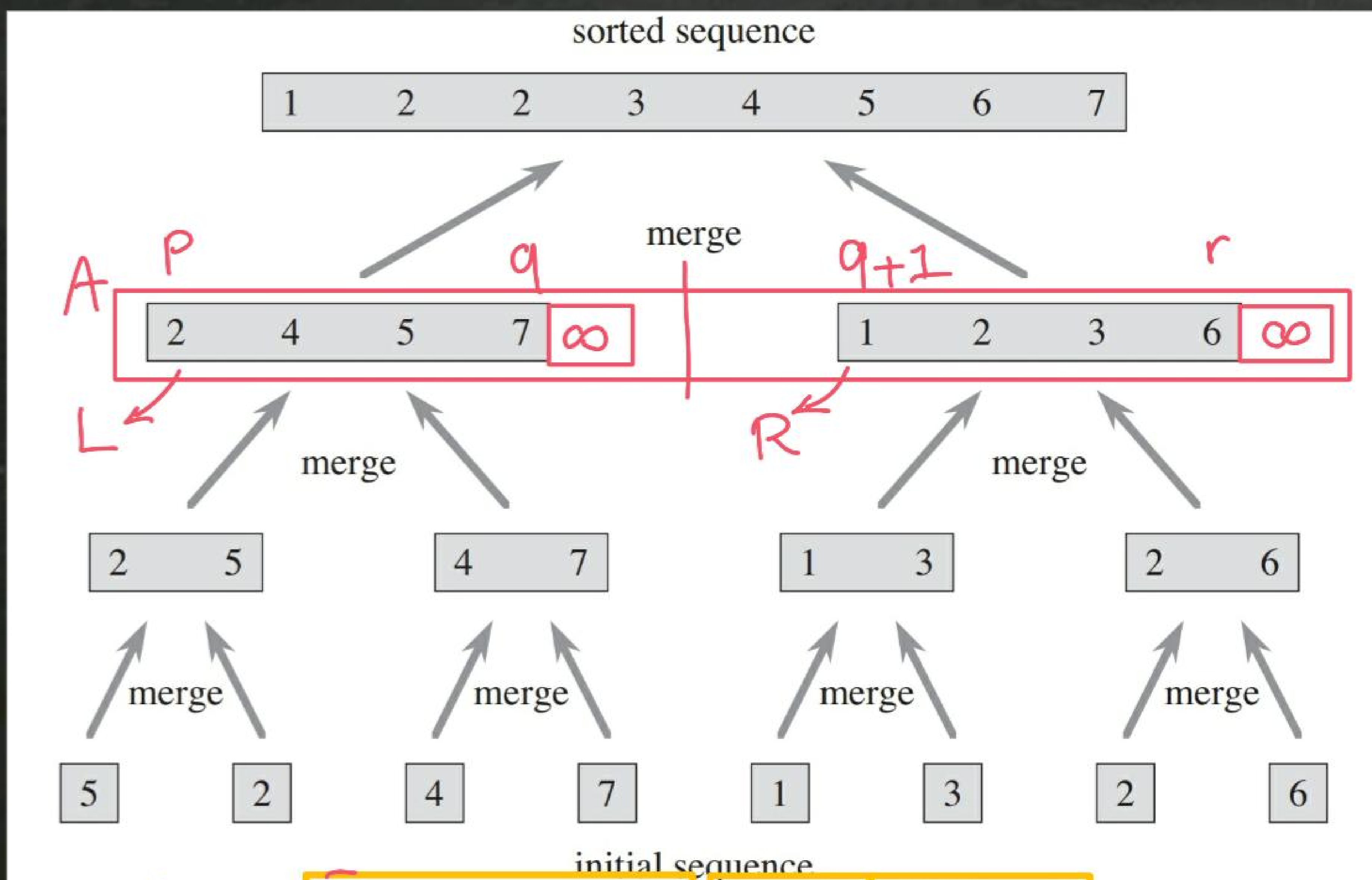
با یک روش مستقیم حل کنید و ب) اگر به‌قدر کافی ساده نیست روال فوق را

بطور بازگشتی تکرار کنید.

گام غلبه Conquer

۳) جواب مسئله P را با ترکیب جواب زیرمسئله‌های $P_1, \ldots, P_k$ به دست آورید.

گام ترکیب Combine



$A = \langle 2,5 \quad 4,7 \quad 1,3 \quad 2,6 \rangle$

الگوریتم مرتب‌سازی ادغامی مثالی از کاربست نگرش تقسیم و غلبه به شمار می‌آید

شبه کد الگوریتم مرتب سازی ادغامی

```
MERGE-SORT(A, p, r)
    if p < r
        q = ⌊(p + r)/2⌋
        MERGE-SORT(A, p, q)
        MERGE-SORT(A, q + 1, r)
        MERGE(A, p, q, r)
```

$\Theta(n \lg n)$

recursion

بازگشتی

تابع

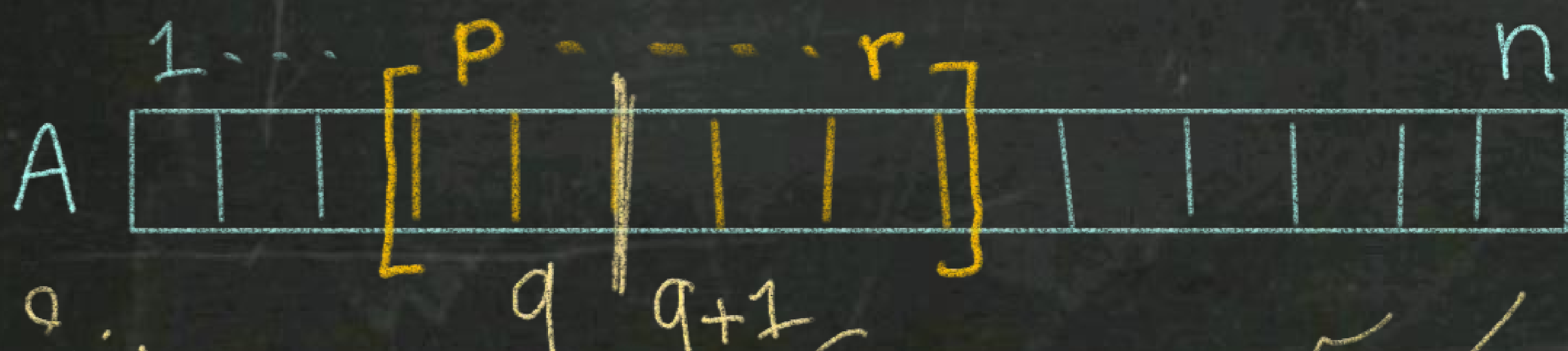
MergeSort(A, p, r)

زیر آرایه

$A[p, p+1, \ldots, r]$

را به روش مرتب سازی

ادغامی مرتب می کند.



(بدیهی است برای مرتب سازی کل آرایه باید این تابع را به صورت زیر فراخوانی شود)

MergeSort(A, 1, n)

الگوریتم ادغام

این الگوریتم با فرض اینکه عناصر در هر یک از زیرآرایه‌های $A[p,\dots,q]$ و $A[q+1,\dots,r]$ به ترتیب صعودی مرتب شده هستند آنها را به گونه‌ای با یکدیگر ادغام می‌کند که در نهایت زیرآرایه $A[p\dots r]$ به ترتیب صعودی مرتب باشد.

MERGE(A, p, q, r)

```
n1 = q - p + 1                                   طول زیرآرایه سمت چپ
n2 = r - q                                       طول زیرآرایه سمت راست
let L[1..n1 + 1] and R[1..n2 + 1] be new arrays
for i = 1 to n1
    L[i] = A[p + i - 1]
for j = 1 to n2
    R[j] = A[q + j]
L[n1 + 1] = ∞
R[n2 + 1] = ∞
i = 1
j = 1
for k = p to r
    if L[i] ≤ R[j]
        A[k] = L[i]
        i = i + 1
    else A[k] = R[j]
        j = j + 1
```

$n := r - p + 1$

$\Theta(n)$

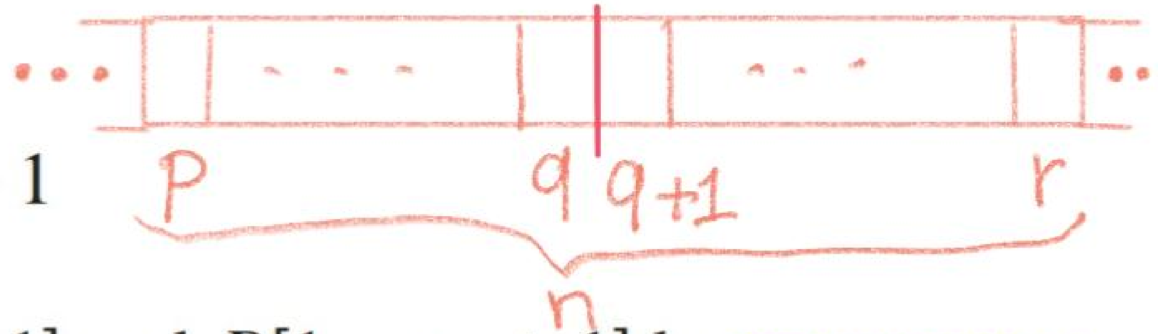
i: شمارنده روی L

j: شمارنده روی R

k: شمارنده روی A که قرار است جواب باشد

تحلیل هزینه زمان اجرای الگوریتم مرتب‌سازی ادغامی

MERGE(A, p, q, r)




```
1  n1 = q - p + 1
2  n2 = r - q
3  let L[1..n1 + 1] and R[1..n2 + 1] be new arrays
4  for i = 1 to n1
5      L[i] = A[p + i - 1]
6  for j = 1 to n2
7      R[j] = A[q + j]
8  L[n1 + 1] = ∞
9  R[n2 + 1] = ∞
10 i = 1
11 j = 1
12 for k = p to r
13     if L[i] ≤ R[j]
14         A[k] = L[i]
15         i = i + 1
16     else A[k] = R[j]
17         j = j + 1
```

$n := r - p + 1$

Lines 10–11: $\Theta(1)$

تعداد تکرار n بار

Lines 13–17: $\Theta(1)$ $\Rightarrow \Theta(n)$

ابتدا لازم است هزینه زمان اجرای الگوریتم ادغام (Merge) را بدست آوریم

$\Theta(1)$: خط‌های 1، 2 و 3

$\Theta(n_1)$: خط‌های 4 و 5

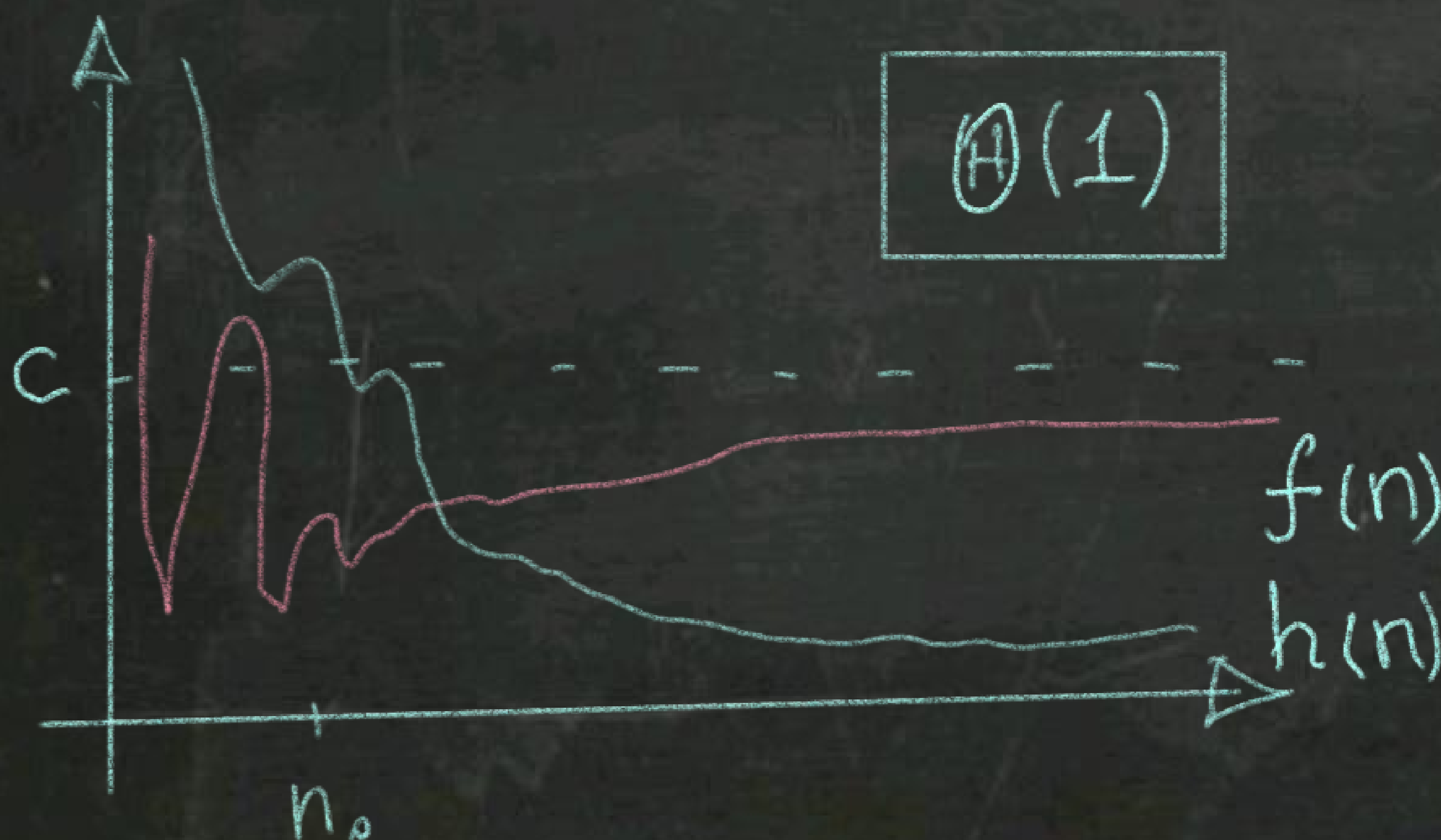
$\Theta(n_2)$: خط‌های 6، 7

$\Theta(1)$: خط‌های 8، 9

$T(n) = \Theta(n)$

دربارهٔ $O(1)$

$$O(1) = \{ f:\mathbb{Z}^{\geqslant 0} \to \mathbb{R} \mid \exists c>0 \text{ and } \exists n_0 \in \mathbb{Z}, \forall n \geqslant n_0,\ 0 \leqslant f(n) \leqslant c \}$$



به بیانِ دیگر، اگر زمان اجرای یکِ الگوریتم از مرتبهٔ $O(1)$ باشد یعنی در کل زمان اجرای آن الگوریتم وابسته به اندازهٔ ورودی یعنی n نیست و حداکثر یک مقدار ثابت است.

تحلیل زمان اجرای مرتب سازی ادغامی

```
MERGE-SORT(A, p, r)
  if p < r
      q = ⌊(p + r)/2⌋
      MERGE-SORT(A, p, q)
      MERGE-SORT(A, q + 1, r)
      MERGE(A, p, q, r)
```

$n := r - p + 1$

$\Theta(1)$

$T(\lfloor n/2 \rfloor)$

$T(\lfloor \frac{n+1}{2} \rfloor)$

$\Theta(n)$

برای تحلیل الگوریتمهایی که شکل بازگشتی دارند زمان اجرای الگوریتم را می کوشیم در فرم یک رابطهٔ بازگشتی بنویسیم

$T(n) :=$ طول زمان اجرای فراخوانی MergeSort(A,p,r)

بنابر این قرارداد، داریم

$$T(n) = \begin{cases} O(1) & \text{if } n = 1 \\ 2T(n/2) + \Theta(n) & \text{elsewhere} \end{cases}$$

حل رابطهٔ بازگشتی برای تعیین مرتبهٔ مجانبی

ابتدا برحسب O تحلیل می‌کنیم (یعنی ≥)

$n_0 = 1$

هدف حل رابطهٔ بازگشتی برای بدست آوردن یک رابطهٔ دقیق نیست بلکه تعیین مرتبهٔ مجانبی جواب کافی است

$$T(n) \leqslant \begin{cases} c_1 & \text{if } n=1 \\ & \text{elsewhere} \\ 2T(n/2)+c_2 n & \end{cases}$$

$$T(n) = 2\,T(n/2) + c_2 n$$

موقتاً فرض کنید n لزوماً یک توان کامل 2 است مثل $n = 2^s$

$$T(2^s) = 2\,T(2^{s-1}) + c_2 \cdot 2^s$$

قرار دادکنید $f(s) := T(2^s)$ و جایگذاری کنید

$$f(s) = 2\,\underline{f(s-1)} + c_2 \cdot 2^s$$

$$f(s) = 2\left(2 \cdot f(s-2) + c_2 \cdot 2^{s-1}\right) + c_2 \cdot 2^s$$

$$f(s) = 4 \cdot f(s-2) + \underbrace{c_2 \cdot 2^s + c_2 \cdot 2^s}_{2c_2 \cdot 2^s}$$

- - - -

$$f(s) = 2^i \cdot f(s-i) + i \cdot c_2 \cdot 2^s$$

قرار دهید $i = s$ ، داریم

$$f(s) = 2^s \cdot f(0) + s \cdot c_2 \cdot 2^s$$

یعنی

$$T(n) = n \cdot \underbrace{T(1)}_{c_1} + c_2 \cdot \lg n \cdot n$$

با توجه به اینکه $n = 2^s$

بنابراین $s = \lg n$

$$T(n) = c_1 n + c_2 n \lg n = \Theta(n \lg n)$$

می توان نشان داد اگر فرض $n = 2^s$ را کنار بگذاریم با همین روند استدلال فوق داریم

$$T(n) \leq c_1 n + c_2 n \lg n$$

یعنی

$$T(n) = O(n \lg n)$$

(در حالت کلی)

رویکردهای تحلیل مجانبی روابط بازگشتی

۱- روش جایگذاری، حدس و استقراء

۲- روش تحلیل درخت

۳- کاربست قضیه اساسی

جایگذاری، حدس و استقرا

۱- جایگذاری. ابتدا پارامتر n را به زیرمجموعه خاصی از اعداد صحیح محدود می‌کنیم

به قسمی که رابطه بازگشتی به ازای آن رده از اعداد شکل

ساده‌تری بگیرد.

در تحلیل مرتب‌سازی ادغامی، به عنوان مثال، گرفتیم $n = 2^s$

۲- حدس. سعی می‌کنیم شکل ساده‌تر شده رابطه بازگشتی را حل کنیم.

با این جواب، اصطلاحاً یک حدس برای حالت کلی فراهم کرده‌ایم

۳-۱ استقرا. به کمک تکنیک اثبات به روش استقرا، حدس را در حالت کلی اثبات می‌کنیم.

مثال. تحلیل زمان الگوریتم مرتب سازی ادغامی. (مرحله‌ٔ اثبات به کمک استقرا)

$$T(n) = \begin{cases} O(1) & \text{if } n = 1 \\ 2T(n/2) + \Theta(n) & \text{if } n > 1 \end{cases}$$

اثبات کنید:

$$T(n) = \Theta(n \lg n)$$

* تابع $n\lg n$ را طی جایگذاری $n=2^s$ توانستیم پیدا کنیم.

اثبات را یکبار برای O و یکبار برای Ω باید انجام دهیم.

در ادامه اثبات $T(n)=O(n\lg n)$ به کمک استقرا می آید. (Ω مشابه است)

کافی است ثابت کنیم:

$$\exists c>0,\ \exists n_0>0,\ \forall n \geqslant n_0,$$

$$T(n) \leqslant c \cdot n \cdot \lg n$$

برای اثبات از تکنیک استقرا قوی استفاده می کنیم.

طبق تکنیک استقراء قوی، می انگاریم رابطه به ازای هر $k < n$ برقرار است و از این

فرض می کوشیم رابطه را به ازای n بدست آوریم. $k < n$

$$T(k) \leqslant c \cdot k \cdot \lg k$$

- قرار دهید $k = n/2$. داریم:

$$T(n/2) \leqslant c \cdot \frac{n}{2} \lg \frac{n}{2}$$

- به رابطه بازگشتی در صورت سؤال برگردید و در آن جایگذاری کنید:

$$T(n) \leqslant 2T(n/2) + c_1 \cdot n \qquad \left(\begin{array}{l} c_1 > 0 \\ n \geqslant n_0 \end{array} \quad \text{مرتبه الگوریتم Merge} \right)$$

$$\Rightarrow T(n) \leq 2 \cdot c \cdot \frac{n}{2} \underbrace{\lg\left(\frac{n}{2}\right)}_{\lg n - 1} + c_1 \cdot n$$

$$= c \cdot n \cdot \lg n \underline{- c \cdot n + c_1 \cdot n}$$

کافی است ضریب c را $c \geq c_1$ بگیریم آنگاه $(c_1 - c) \cdot n \leq 0$

و بنابراین

$$T(n) \leq c \cdot n \cdot \lg n$$