نکته: الگوریتم brute-force یا exhaustive search یا generate and search یک روش غیر هوشمندانه برای حل مسئله است که تمام راه حل های ممکن را برای رسیدن به جواب چک می کند. مثل چک کردن تمام اعداد بین ۱ تا n برای بدست آوردن مجموعه مقسوم علیه های عدد n.

Merge sort :

یک الگوریتم مرتب سازی براساس تکنیک divide and conquer است.

- این الگوریتم درحالت worst-case از insertion sort بهتر عمل می کند و order کمتری دارد.

- هدف sorting یک آرایه $A[1..n]$

- چون در این الگوریتم با subproblem ها کار داریم مسئله را به این صورت می نویسیم.

- sort کردن آرایه $A[p..r]$ که در ابتدا $p=1$ و $r=n$ است.

- برای sort آرایه $A[p..r]$:

- بخش Divide : آرایه A را به دو زیر آرایه $A[p..q]$ و $A[q+1..r]$ تقسیم می کنیم، q میانه ی آرایه A باشد.

- بخش Conquer : به صورت recursively دو زیرآرایه $A[p..q]$ و $A[q+1..r]$ را حل می کنیم.

- بخش Combine : دو زیرآرایه مرتب شده ی $A[p..q]$ و $A[q+1..r]$ را ادغام می کنیم تا یک آرایه مرتب شده ی $A[p..r]$ بدست آید. برای این گام یک procedure تعریف می کنیم تا عمل ادغام را انجام دهد.

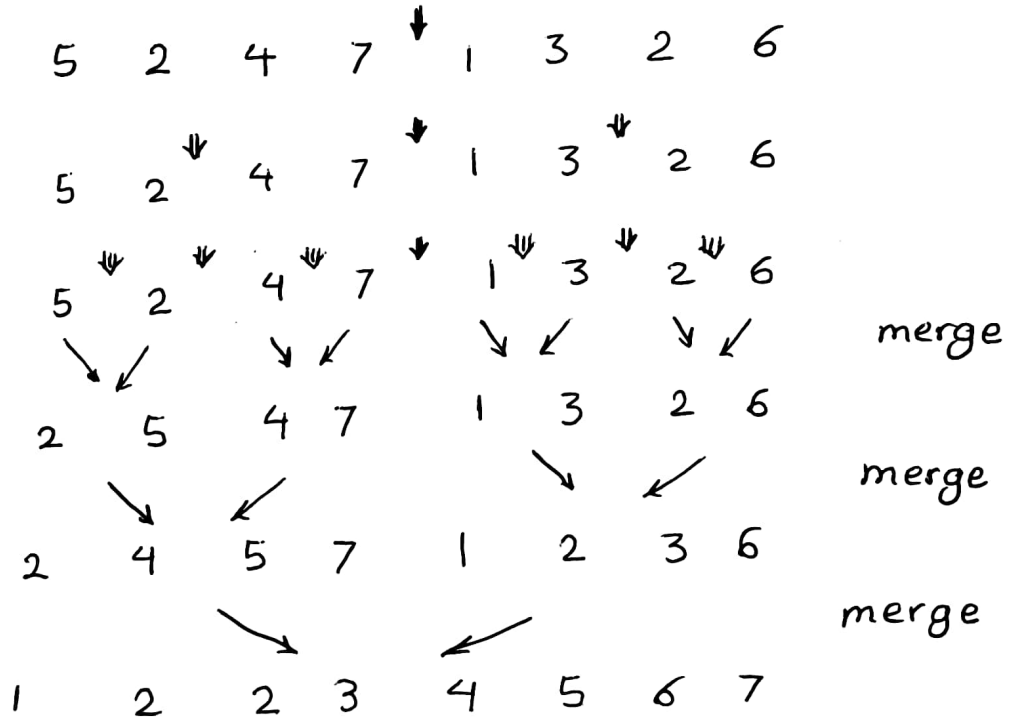
نکته: recursion زمانی به پایان می رسد که هر زیرآرایه فقط یک element داشته باشد که به طور بدیهی زیرآرایه یک عضوی sort شده است.

```
MERGE-SORT (A, p, r)
if p<r                                  ▷ check for base case
   then q ← ⌊(p+r)/2⌋                   ▷ Divide
        MERGE-SORT (A, p, q)            ▷ Conquer
        MERGE-SORT (A, q+1, r)          ▷ Conquer
        MERGE (A, p, q, r)              ▷ Combine
```

Initial call : MERGE-SORT (A, 1, n)

مثال :

یک آرایه ۸عضوی داده‌شده‌است :



Merging :

در پروسیجر Merge چه اتفاقی می افتد.

input : Array A , indices p, q, r

→ $p \le q < r$

→ A[p..q] & A[q+1..r] are non-empty and sorted.

output: A[p..r] is sorted.

درخروجی هر دو زیرآرایه مرتب شده در یک آرایه ادغام می‌شوند که آرایه نهایی هم مرتب شده است.

- در پروسیجر MERGE تعداد $n = r - p + 1$ تعداد عناصری است که از دو زیرآرایه در MERGE شرکت می‌کنند،
- پروسیجر MERGE در زمان $\Theta(n)$ انجام می‌شود.

نکته: معمولاً تعداد n به عنوان سایز general problem در نظر گرفته می‌شود. البته در اینجا n را برای سایز A[p..r] در نظر گرفته‌ایم.

ایده‌ی Linear-time merging:

- دو دسته کارت که هر دو مرتب شده و رو به بالا هستند را در نظر بگیرید. که کوچکترین کارت رو باشد.
- این دو دسته را به یک دسته مرتب شده رو به پایین (face-down) تبدیل و merge می‌کنیم.
- basic step:
  - از بین کارتهای روی دو دسته (دو کارت) کوچکتر را انتخاب کن.
  - این کارت را از روی دسته اش بردار، اکنون یک کارت جدید روی آن دسته داریم.
  - کارت انتخاب شده را face-down روی زمین بگذارید.
- گام basic step را آنقدر تکرار می‌کنیم تا یک دسته خالی شود.
- اگر یک دسته خالی شد، دسته دیگر را face-down روی دسته merge شده قرار بده.
- هر گام basic step زمان constant لازم دارد، چون فقط دو کارت رو را بررسی می‌کنیم.
- تعداد تکرارهای basic step کمتر یا مساوی n است. چون مجموع کارتهای دو دسته n است
- بنابراین، پروسیجر در زمان $\Theta(n)$ کار می‌کند.

```
MERGE (A, p, q, r)
n1 ← q - p + 1
n2 ← r - q
Create arrays L[1..n1+1] and R[1..n2+1]
for i ← 1 to n1
    do L[i] ← A[p + i - 1]
for j ← 1 to n2
    do R[j] ← A[q + j]
L[n1+1] ← ∞
R[n2+1] ← ∞
i ← 1
j ← 1
for k ← p to r
    do if L[i] ≤ R[j]
          then A[k] ← L[i]
               i ← i + 1
          else A[k] ← R[j]
               j ← j + 1
```

: Running time

دو for loop اول که $\Theta(n_1+n_2)=\Theta(n)$ ، آخرین forloop ، n بار اجرا دارد که هر بار اجرا constant است.

Total time : $\Theta(n)$

Example: A call of MERGE(9, 12, 16)