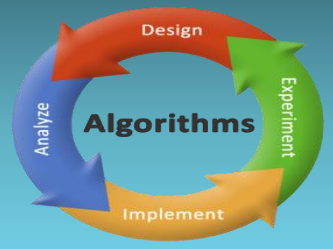
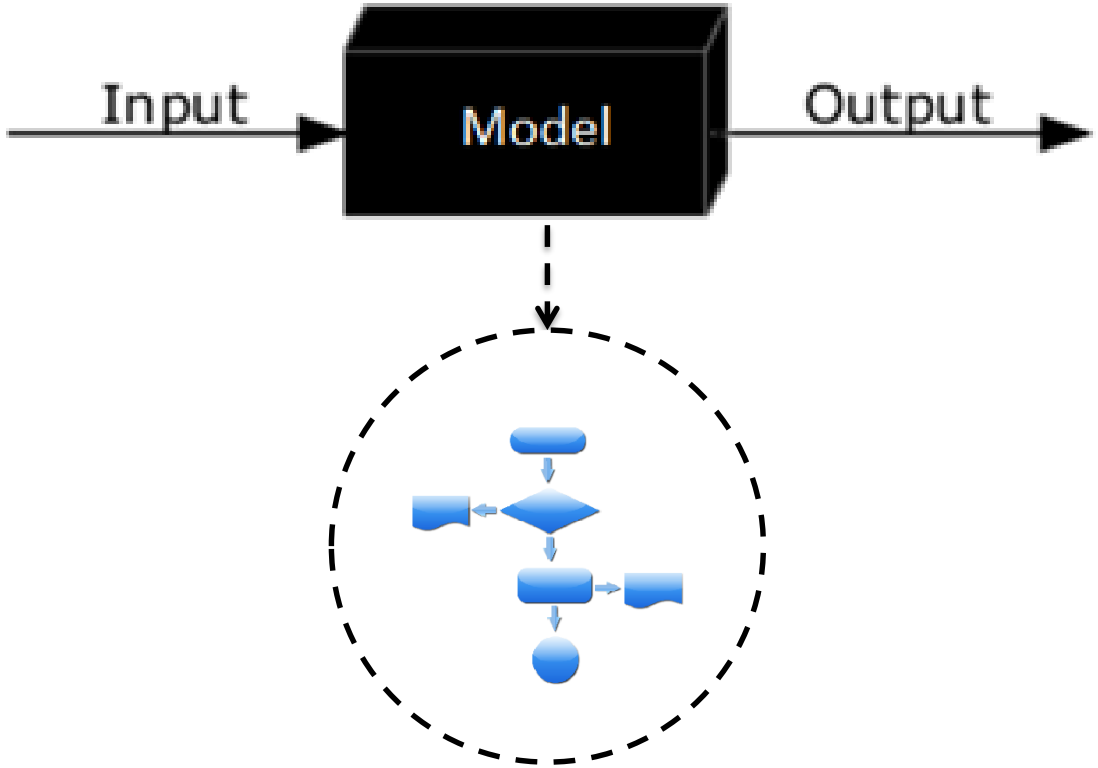
معرفی تحلیل الگوریتم

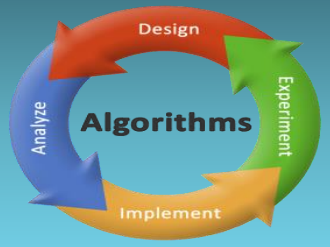


تعریف الگوریتم

❑الگوریتم:

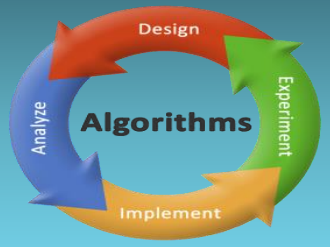
✓مجموعه‌ای متناهی از دستورالعمل‌ها است، که به ترتیب خاصی اجرا می‌شوند و مسئله‌ای را حل می‌کنند.





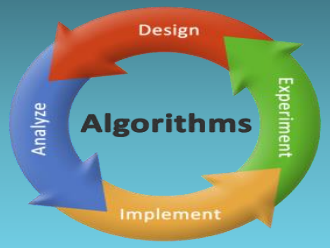
منابع مورد نیاز یک الگوریتم

- **حافظه (SPACE)**
  - ✓ حافظه‌ی مورد نظر ما RAMهست؛ زیرا برای آنکه بتوانیم روی داده‌ها پردازش انجام دهیم باید آنها را روی RAMبارگذاری بکنیم.
  - ✓ هرچند امروزه با پیشرفت تکنولوژی حافظه‌ها روز به روز در حال افزایش هستند ولی مسئله‌ی حافظه هنگام کار با داده‌های بزرگی مثل تصویر یا پایگاه‌داده‌ی گوگل همچنان باقی است.
- **زمان (TIME)**
  - ✓ زمانی که برای اجرای یک الگوریتم صرف می‌شود، زمانی است که صرف اجرای دستورات موجود در الگوریتم می‌شود.
  - ✓ این زمان، بسته به مثلا سخت‌افزار مورد استفاده، زبان برنامه‌نویسی و سایر برنامه‌های در حال اجرا در لحظه اجرای کد، متفاوت است.
  - ✓ بنابراین منطقی است که تعداد دفعات اجرای دستورات را محاسبه می‌کنیم.



تحلیل الگوریتم

- منظور از تحلیلی یک الگوریتم، محاسبه میزان منابع مورد استفاده آن است:
  - ✓ زمان
  - ✓ حافظه
  - ✓ پهنای باند
  - ✓ ...
- می‌خواهیم این تحلیل به گونه‌ای باشد که امکان مقایسه الگوریتم را با سایر الگوریتم‌ها فراهم نماید.
- به طور کلی هنگامی که یک الگوریتم را تحلیل می‌کنند، بیشتر به بحث زمان (پیچیدگی محاسباتی) می‌پردازند و کمتر به حافظه پرداخته می‌شود.
- زیرا همانطور که گفته شد، روز به روز حافظه‌ها بزرگتر می‌شوند ولی همچنان درمورد سرعت با مشکل روبرو هستیم.
- یک الگوریتم خوب و سریع می‌تواند ضعف سخت‌افزار را بپوشاند.
- برای درک بیشتر این موضوع به مسئله‌ی اسلاید بعدی دقت کنید.



کدام یک مهمتر است؟ الگوریتم یا سخت‌افزار؟

کامپیوتر A : 10^{11} دستورالعمل بر ثانیه
کامپیوتر B : 10^{8} دستورالعمل بر ثانیه

کامپیوتر A ۱۰۰۰ برابر سریع تر از کامپیوتر B هست.

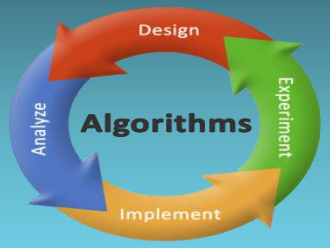
تعداد دستورالعمل های الگوریتم مرتب‌سازی درجی: $\mathbf{2n^2}$
تعداد دستورالعمل های الگوریتم مرتب‌سازی ادغامی: $\mathbf{50\ n\ log_{10}\ n}$

تعداد داده ها: ۱ میلیون

اگر کامپیوتر A از مرتب سازی درجی و کامپیوتر B از مرتب سازی ادغامی استفاده کند کدام کامپیوتر سریع‌تر مسئله را حل می‌کند؟

$\mathbf{Time_A} = [2*(10^6)^2 / 10^{11}] = \mathbf{20\ seconds}$

$\mathbf{Time_B} = [50*(10^6) * log_{10}(10^6) / 10^8] = \mathbf{3\ seconds!}$



محاسبه‌ی تعداد اجرا یک دستورالعمل

```
x=0;                                    //t1
For i=1 to n                            //t2*(n+1)
  x++                                   //t3*(n)
```

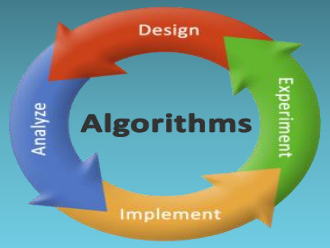
$T(n) = t1 + t2 \times n + t2 + t3 \times n = (t2 + t3)n + (t1 + t2)$

```
x=0;                                    //t1
For i=1 to n                            //t2*(n+1)
  For j=1 to m                          //t3*(m+1)(n)
    x=i*j+x;                            //t4*(m)(n)
```

$T(n,m) = (t4 + t3)mn + (t3 + t2)n + (t2 + t1)$

```
x=0;                                    //t1
For i=1 to n                            //t2*(n+1)
  For j=1 to i                          //t3*∑_{i=1}^{n}(i + 1)
    x=x+i*j                             //t4*∑_{i=1}^{n}(i)
```

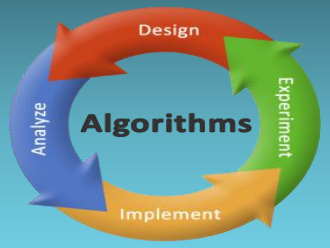
$T(n) = \mathrm{t1} + \mathrm{t2} + \mathrm{t2} \times n + t3 \times \left(\frac{n^2 + 3n}{2}\right) + \mathrm{t4} \times \left(\frac{n^2 + n}{2}\right)$



مرتب‌سازی

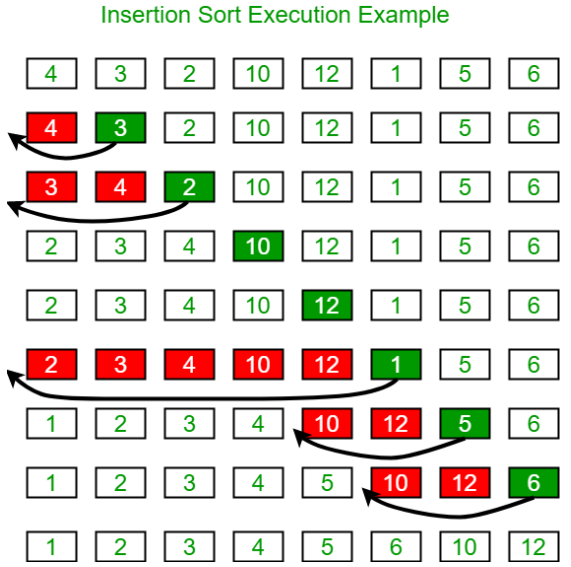
- مرتب‌سازی سریع (Quick Sort)
- مرتب‌سازی درجی (Insertion Sort)
- مرتب‌سازی ادغامی (Merge Sort)
- مرتب‌سازی مبنایی (Radix Sort)
- و غیره

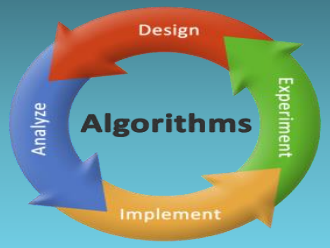
در بسیاری از مسائل امروزی، از الگوریتم‌های مرتب‌سازی استفاده می‌شود. برای مسئله‌ی مرتب‌سازی الگوریتم‌های زیادی وجود دارد که در فصل بعد به بررسی آن‌ها می‌پردازیم.



Insertion Sort

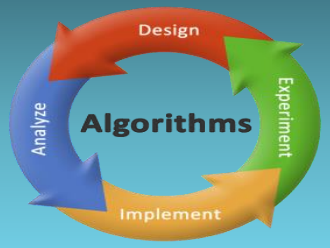
```
INSERTION-SORT(A)
1  for j = 2 to A.length
2      key = A[j]
3      // Insert A[j] into the sorted
           sequence A[1 . . j − 1].
4      i = j − 1
5      while i > 0 and A[i] > key
6          A[i + 1] = A[i]
7          i = i − 1
8      A[i + 1] = key
```





Analysis of insertion sort

Algorithm(A)	cost	times
for i = 2 to A.length	t1	n
key = A[j]	t2	n-1
j = i-1	t3	n-1
while j > 0 and A[j-1] > A[j]	T4	$\sum_{i=2}^{n} n_i$
t = A[j]	t5	$\sum_{i=2}^{n} n_i - 1$
j = j - 1	t6	$\sum_{i=2}^{n} n_i - 1$
end while	0	-
A[i+1] = key	t7	n-1
end for	0	-



بررسی زمان اجرای مرتب سازی درجی (Insertion Sort)

$$T(n)= t1\times 1 + t2\times n + t3\times (n-1) + \mathrm{t4}\times \sum_{1}^{n} n_i \; + t5\times \sum_{1}^{n} (n_i - 1) \; + t6\times \sum_{1}^{n} (n_i - 1) \; + t7\times (n-1)$$

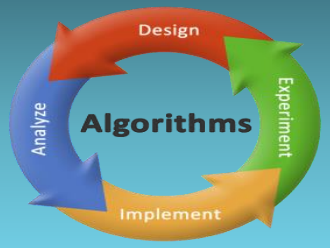
Best case:

$$T(n)=(t2+t3+t4+t7)n-(t3+t7-t1) = an + b \rightarrow o(n)$$

Worst case:

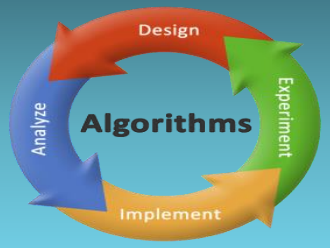
$$T(n) = t1\times 1 + t2\times n + t3\times (n-1) + \mathrm{t4}\times \frac{n(n+1)}{2} - 1 + t5\times \frac{n(n-1)}{2} \; + t6\times \frac{n(n-1)}{2} \; + t7\times (n-1)$$

$$T(n) = an^2 + bn + c \rightarrow o(n^2)$$



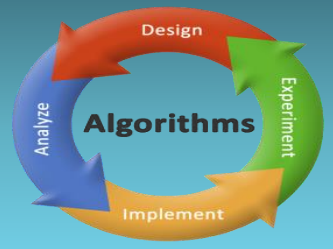
Merge Sort

```
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
```

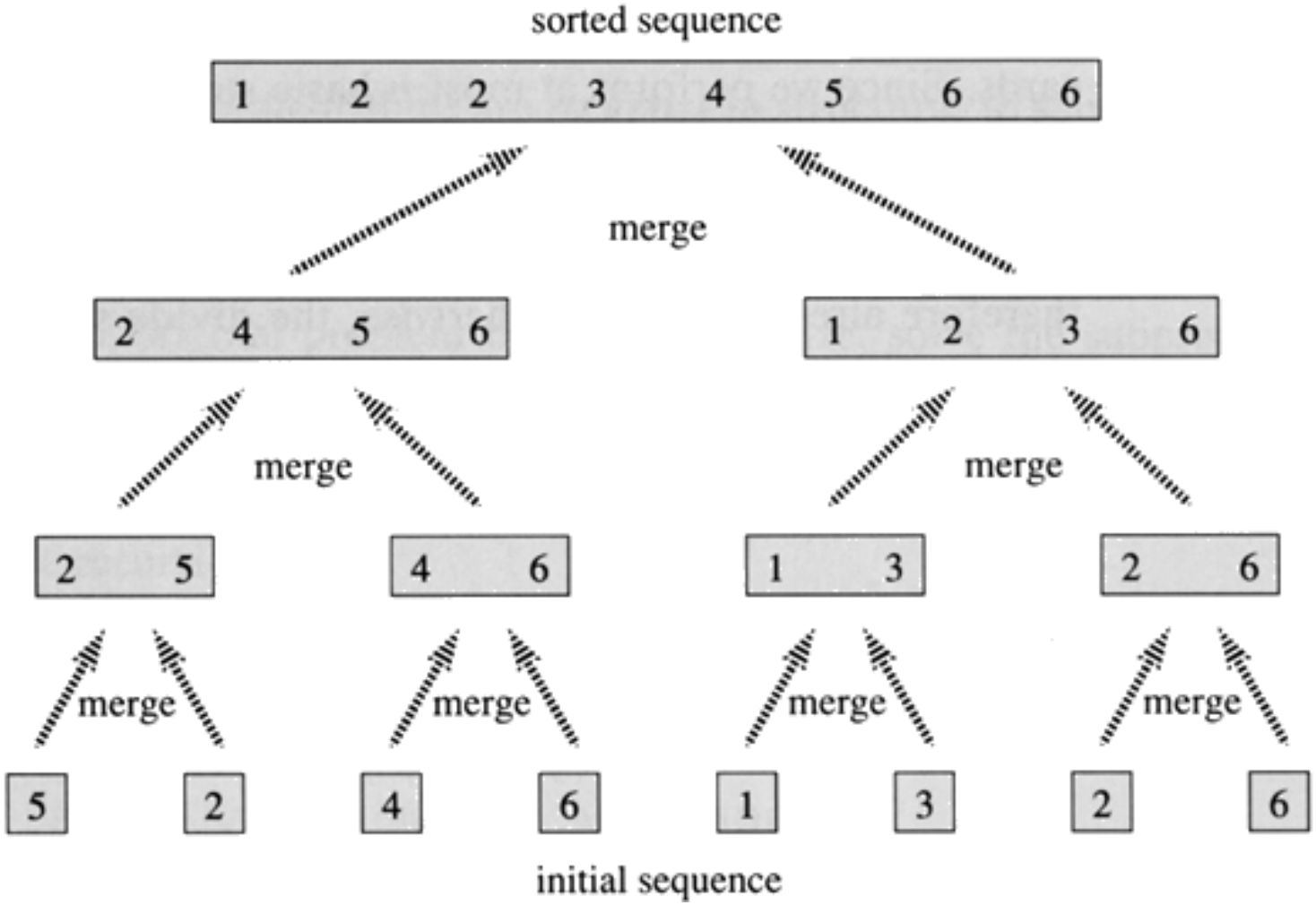


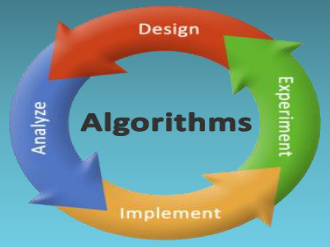
Merge Sort

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


Design
Experiment
Implement
Analyze
Algorithms

Merge Sort


sorted sequence
1 2 2 3 4 5 6 6
merge
2 4 5 6
1 2 3 6
merge
merge
2 5
4 6
1 3
2 6
merge
merge
merge
merge
5
2
4
6
1
3
2
6
initial sequence



مرتب سازی ادغامی (merge sort)

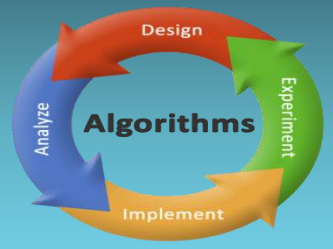
روش مرتب‌سازی ادغامی یک روش مرتب‌سازی مبتنی بر مقایسه‌ی عناصر با استفاده از **روش تقسیم و غلبه** است. این روش از مراحل بازگشتی زیر تشکیل یافته است:

۱- آرایه را به دو زیرآرایه با اندازه‌ی تقریبا یکسان تقسیم کن.

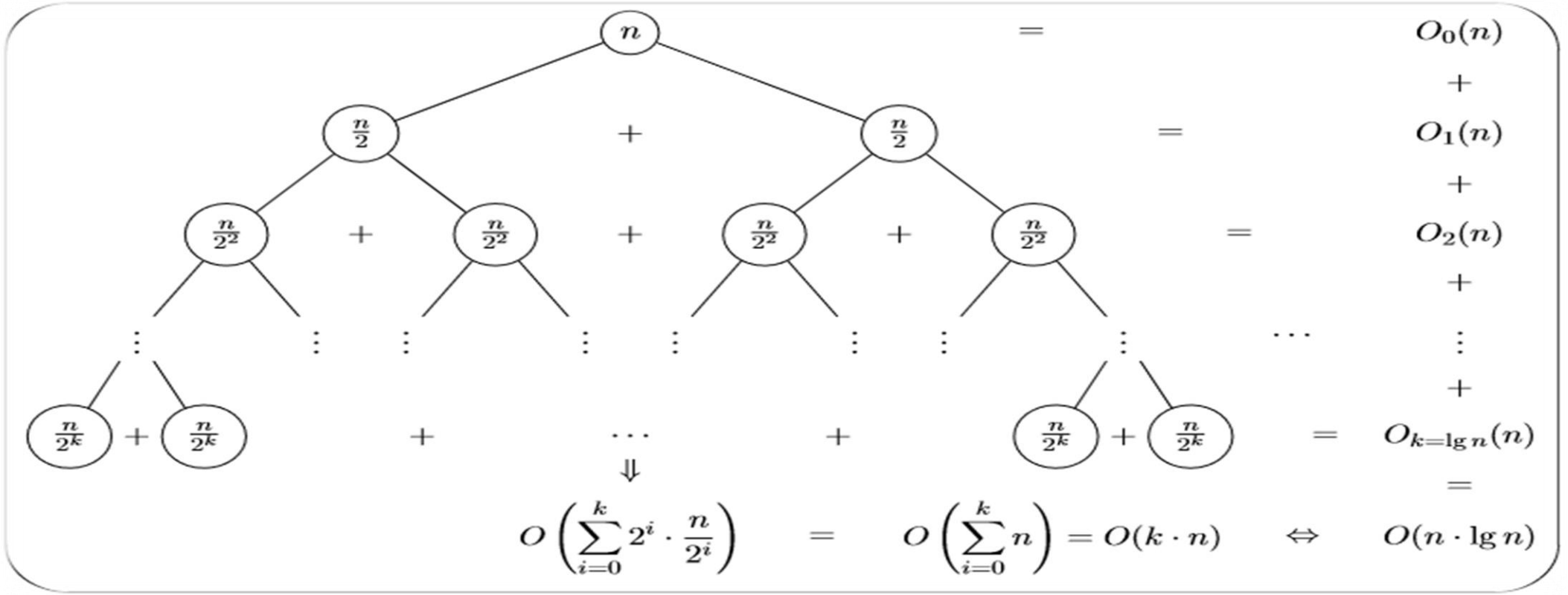
۲- دو زیرآرایه را به روش مرتب‌سازی ادغامی مرتب کن.

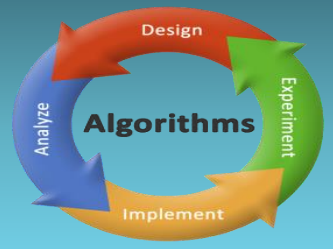
۳- دو زیرآرایه‌ی مرتب‌شده را ادغام کن.

6 5 3 1 8 7 2 4



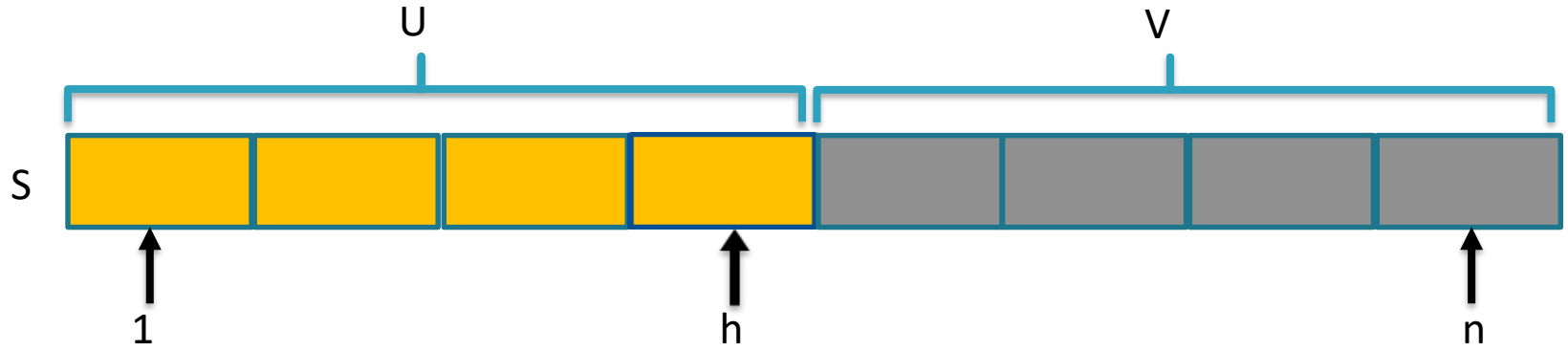
درخت مرتب سازی ادغامی (merge sort)



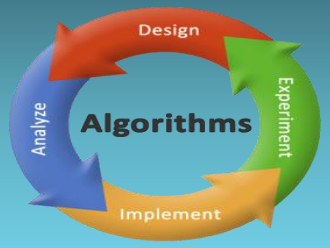


الگوریتم مرتب سازی ادغامی

```
Mergesort (S[] , n )
{
  h= [ n / 2 ] ;
  m = n – h ;
  if ( n > 1 )
      {
          copy S [ 1….h ] to U [ 1….h];
          copy S [h+1….n] to V  [1……m];
          mergesort (U,h );
          mergesort ( V,m );
          merge (U,h,V,m,S);
       }
}
```



$$T(n) = 2T(n/2) + n$$
$$\rightarrow T(n) \in \theta(nlgn)$$



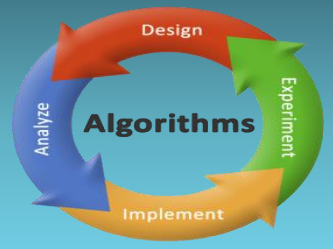
ویژگی‌های مرتب‌سازی ادغامی

مرتب‌سازی ادغامی ویژگی‌های زیر را دارد:

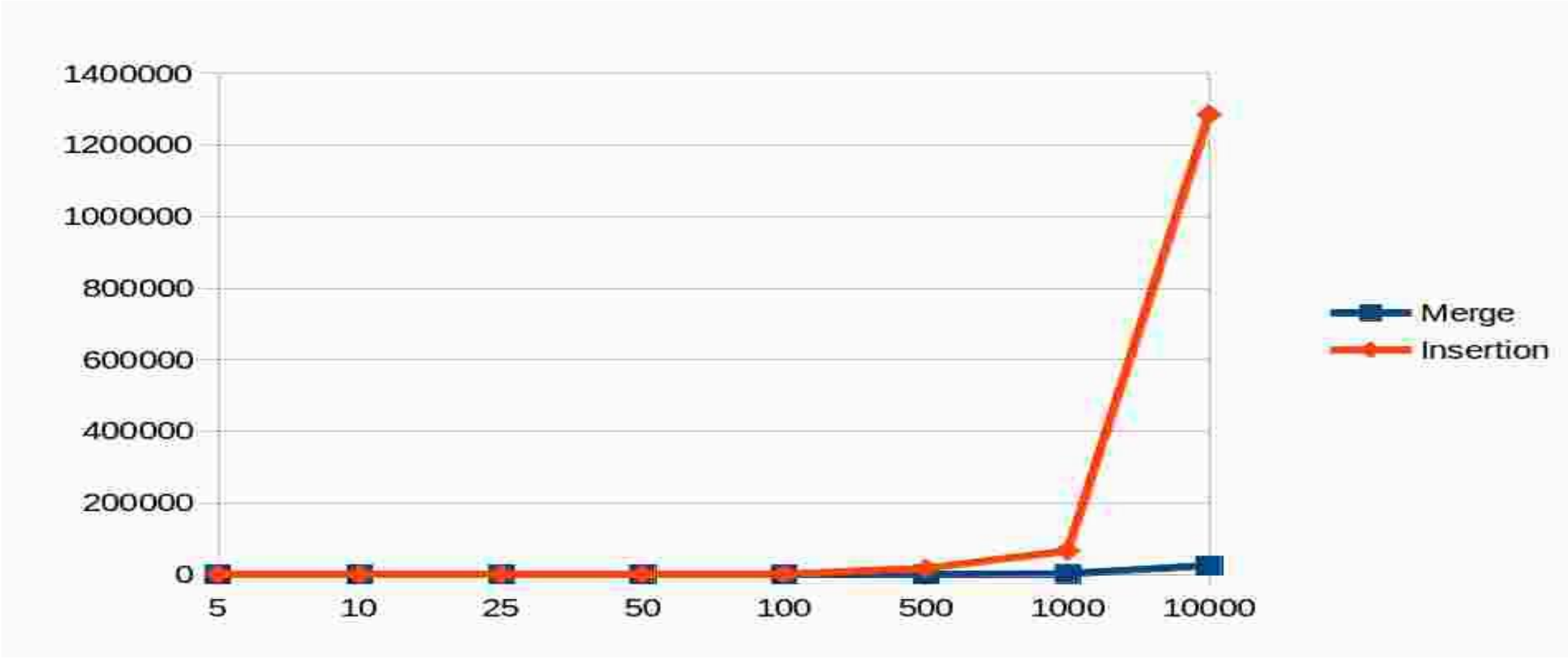
۱- پیچیدگی زمانی اجرای الگوریتم در تمامی حالات **θ(nlogn)** است؛ چرا که این الگوریتم تحت هر شرایطی آرایه را به دو قسمت کرده و مرتب‌سازی را انجام می‌دهد.

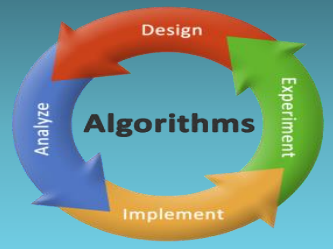
۲- پیچیدگی حافظه‌ی مصرفی بستگی به روش پیاده‌سازی مرحله‌ی ادغام دارد که تا **O(n)** افزایش می‌یابد. پیاده‌سازی درجای این الگوریتم حافظه‌ی مصرفی مرتبه‌ی **θ(1)**دارد؛ اما اجرای آن در بدترین حالت زمان‌بر است.

۳- الگوریتم مرتب‌سازی ادغامی با پیاده‌سازی فوق یک روش پایدار است. چنین الگوریتمی ترتیب عناصر با مقدار یکسان را پس از مرتب‌سازی حفظ می‌کند.

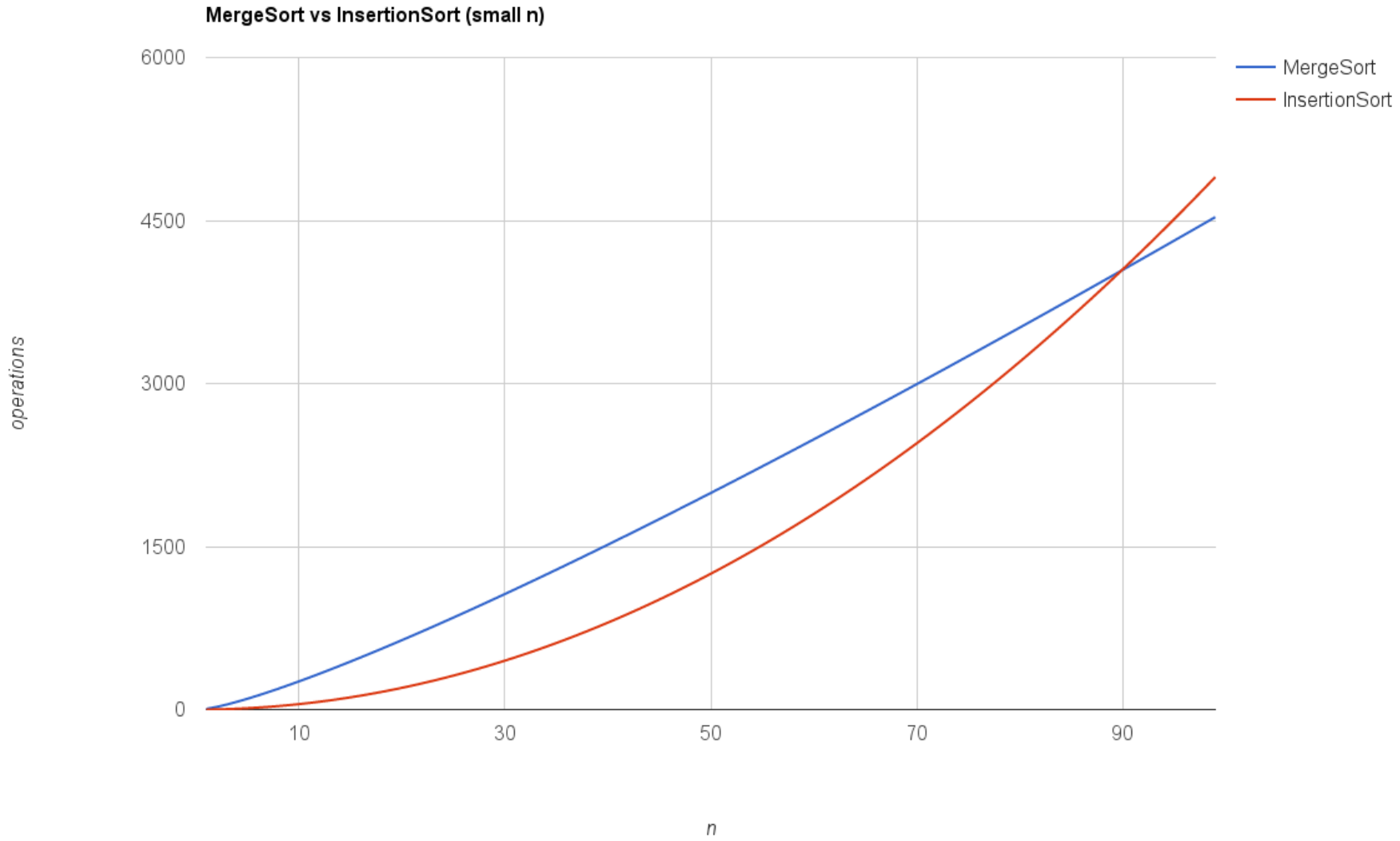


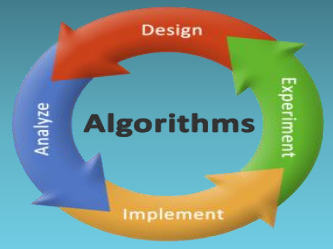
مقایسه دو اگوریتم مرتب سازی





مقایسه دو اگوریتم مرتب سازی برای مقادیر کوچک n





مقایسه دو اگوریتم مرتب سازی برای مقادیر بزرگ n