پرسش: الگوریتم چیست؟ چه تفاوتی با برنامه دارد؟ چه نقشی در مهندسی کامپیوتر دارد؟ چرا طراحی و آنالیز الگوریتم‌ها مهم است؟!!

گام‌های مهم در طراحی الگوریتم؛ مساله ← راه حل ← الگوریتم ← برنامه

- الگوریتم یک روند کامل گام به گام برای حل یک مساله است.
- الگوریتم ابزاری برای حل مسائل محاسباتی خوش‌تعریف است.

شکل ۱-۱: الگوریتم به عنوان یک جعبه سیاه

مثال: مرتب‌سازی

Input: sequence of number $a_1, a_2, \ldots, a_n$ ➔ { **SORTING** } ➔ **Output:** An ordered permutation of input $a'_1 \le a'_2, \ldots \le a'_n$.

چند مثال دیگر

الگوریتم جستجوی ترتیبی

Problem: Is the key x in the array S of n keys?
Inputs: positive integer n, array of keys S indexed from 1 to n, and a key x.
Output : *location*, the location of x in S (0 if x is not in S)

```
void seqsearch (int n, const keytype S[], keytype x,
                index& location)
 {
   location = 1;
   while (location <= n && S [location] != x )
       location ++;
   if (location > n)
       location = 0;
 }
```

Alg. 1-1

الگوریتم محاسبه مجموع عناصر یک آرایه

Problem: Add all numbers in the array S of n numbers.
Inputs: positive integer n, array of numbers S indexed from 1 to n.
Output: *sum*, the sum of the numbers in S

```
number sum (int n, const number S[])
 {
   index i;
   number result;
   result = 0;
   for (i = 1; i <= n ; i++)
       result = result + S[i];
   return result;
 }
```

Alg. 1-2

پرسش: شبه‌کد (Pseudocode) چیست؟

use **if** $(i \le x \le j)$ instead **if** $(i <= x \ \&\& \ x <= j)$
use **exchange** x and y instead ?

comparison	C++ Symbol
$x = y$	$x = = y$
$x \ne y$	$x \ ! = y$
$x \ge y$	$x > = y$
$x \le y$	$x < = y$

Operator	C++ Symbol
and	&&
or	‖
not	!

مرتب‌سازی تعویضی

```
void exchangesort (int n, keytype S[])
 {  index  i, j;
    for (i  = 1; i <= n ; i++)
        for (j  = i + 1; j <= n ; j++)
            if  (S[j] < S[i])
                exchange S[j] and S[i];
 }
```

Alg. 1-3

ضرب دو ماتریس

```
void matrixmult (int n, const number A[][], B[][],  number C[][])
 {
    index  i, j, k;
     for (i  = 1; i <= n ; i++)
        for ( j  = 1; j <= n ; j++)
            {  C[i][j] = 0;
               for (k  = 1; k <= n ; k++)
                    C[i][j] += A[i][k]×B[k][j] ;
            }
 }
```

Alg. 1-4

پرسش: کارآیی الگوریتم چیست و چگونه تعیین‌می‌شود؟ چه روش یا روش‌هایی برای تعیین آن، مناسب‌تر است؟

جستجوی دودویی

```
void binsearch (int n, const keytype S[], keytype x,
index&  location)
 {
   index low, high, mid;
    low = 1;   high = n;   location = 0;
    while ( low  <= high && location = = 0 )
      {   mid = ⌊(low + high )/2⌋;
          if ( x = = S[mid] )
             location = mid ;
          else if  ( x < S[mid] )
                   high = mid–1;
               else low = mid + 1;
      }
 }
```

Alg. 1-5

مقایسه کارآیی جستجوی ترتیبی و جستجوی دودویی

اندازه آرایه	تعداد مقایسه Sequential Search	تعداد مقایسه Binary Search
128	128	8
1024	1024	11
1048576	1048576	21
4294967296	4294967296	33

سری فیبوناچی (Fibonacci Sequence)

$\mathbf{f_0 = 0, f_1 = 1,}$ $\mathbf{f_n = f_{n-1} + f_{n-2}}$

$\mathbf{f_n}$: 0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144,…

پرسش: درخت بازگشتی برای $\mathbf{f_5}$ (جمله پنجم سری فیبوناچی)ترسیم کنید.

الگوریتم بازگشتی

```
int fib_Recursive (int n)
 {
   if ( n <= 1)
     return n;
   else
    return (fib_Recursive(n–1) + fib_Recursive (n–2));
 }
```

Alg. 1-6

الگوریتم تکراری

```
int fib_Iterative (int n)
 {
    index  i;   int  f [0..n];       f [0] = 0;
    if ( n > 0)
       {  f [1] = 1;
          for ( i = 2;  i <= n;  i ++ )
                f [i] = f [i–1]  + f [i–2];     }
    return f [n];
 }
```

Alg. 1-7

قضیه:

If $T(n)$ is the number of terms in the recursion tree corresponding to Alg. 1.6, then, for $n \geq 2$, $T(n) > 2^{n/2}$

اثبات: با استفاده از استقراء ثابت می‌شود.

مقایسه کارآیی این دو الگوریتم

n	n+1	$2^{n/2}$	Execution Time Using Alg. 1-7	Execution Time Using Alg. 1-6
40	41	1048576	41 ns	1048 ms
60	61	1.1x109	61 ns	1 s
80	81	1.1x1092	81 ns	18 min
100	101	1.1x1015	101 ns	13 days
120	121	1.2x1018	121 ns	36 years
160	161	1.2x1024	161 ns	3.8x107 years
200	201	1.3x1030	201 ns	4x1013 years

نکته: گاهی الگوریتم‌های متفاوت برای مساله‌ای یکسان، کارآیی‌ها و پیچیدگی‌های متفاوتی دارند. در ادامه چند مثال آمده است:

- جستجوی ترتیبی و جستجوی دودویی
- محاسبه جمله nام فیبوناچی به روش بازگشتی و تکراری
- مرتب‌سازی درجی و مرتب‌سازی حبابی(برای اطلاعات بیشتر به درس ساختمان داده‌ها مراجعه کنید)
- محاسبه حاصضرب ماتریس به روش مستقیم و روش سریع (در سری‌های بعدی درس، بیشتر توضیح داده خواهد شد)
- محاسبه توان به روش بازگشتی و تکراری (در سری های بعدی درس، بیشتر توضیح داده خواهد شد)

پرسش: منظور از آنالیز الگوریتم‌ها چیست؟

آنالیز الگوریتم‌ها

آنالیز پیچیدگی (Complexity Analysis)

- Time Complexity: CPU time, number of computations
- Memory Complexity: memory consumption
- Power Complexity: power consumption

آنالیز درستی (Correctness Analysis)

Develop a proof that the algorithm actually does what itis supposed to do

آنالیز پیچیدگی زمانی

۱- پیچیدگی زمانی برای همه موارد

Every-Case Time Complexity T(n)
The number of times the algorithms does the basic operation.

۲- پیچیدگی زمانی در بدترین حالت

Worst-Case Time Complexity W(n)
The maximum number of times the algorithm will ever do its basic operation.

۳- پیچیدگی زمانی در حالت میانگین

Average-Case Time Complexity A(n)
The average of the number of times the algorithms does the basic operation.

۴- پیچیدگی زمانی در بهترین حالت

Best-Case Time Complexity B(n)
The minimum number of times the algorithm will ever do its basic operation.

کاربردهایی از الگوریتم‌ها

- الگوریتم‌های جستجو و مسیریابی در اینترنت
- الگوریتم‌های بهینه‌یابی
- الگوریتم‌های زمان‌بندی
- الگوریتم‌های مرتب‌سازی
- الگوریتم‌های محاسبات عددی
- ....

تمرین‌ها

تمرین ۱-۱: برای هفت الگوریتم ارائه‌شده در این سری(سری اول)، آنالیز پیچیدگی زمانی T(n)، W(n)، A(n) و B(n) را بررسی کنید. پس از بررسی آنالیزهای فوق، چه نتیجه یا نتایجی می‌گیرید؟ روی نتایج به دست‌آمده، بحث کنید.

تمرین ۱-۲: تحقیق کنید که علاوه بر کاربردهای ذکر شده در بالا، چه کاربردهای دیگری برای الگوریتم‌ها، می‌توان نام برد.