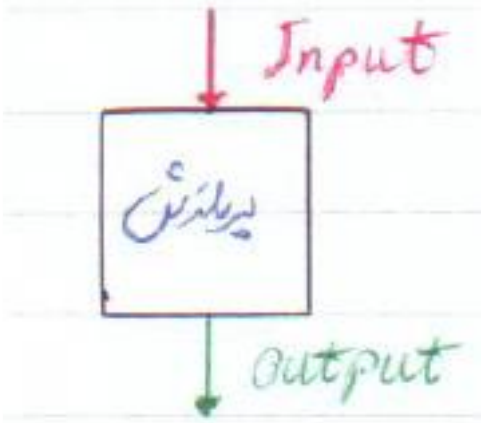
طراحی الگوریتم ها

پیچیدگی الگوریتم ها

استاد درس: مهدی جبل عاملی

تعریف الگوریتم

الگوریتم یک روال محاسباتی است که مجموعه ای از مقادیر را به عنوان ورودی می گیرد و مجموعه ای از مقادیر را به عنوان خروجی بر می گرداند.

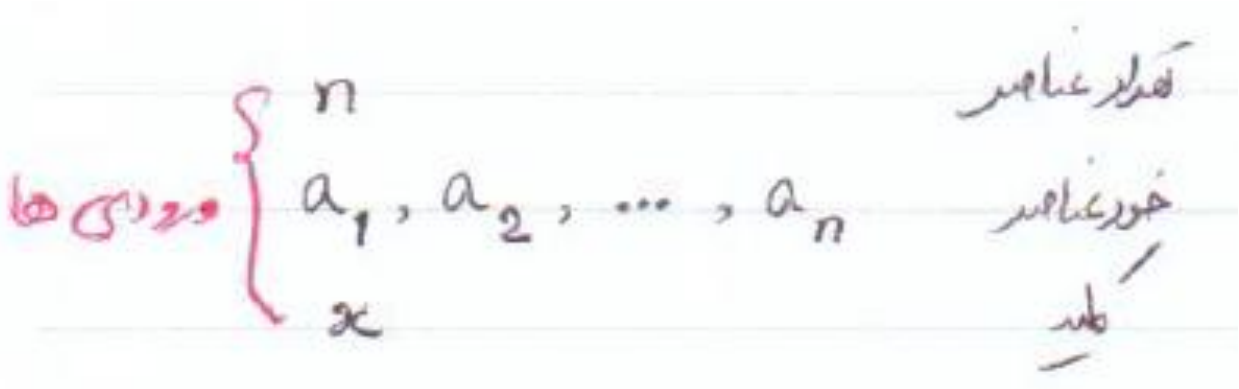


جستجو (Search)

- ورودی ها:

جستجو (Search)

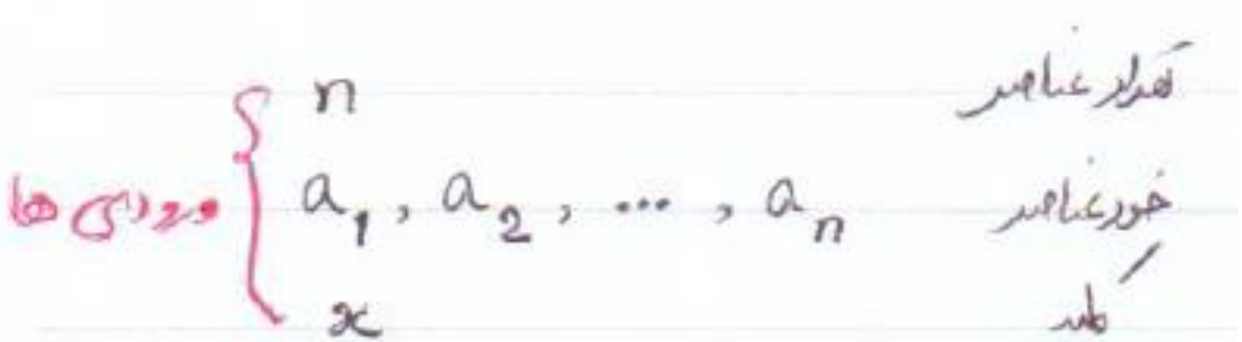
- ورودی ها:



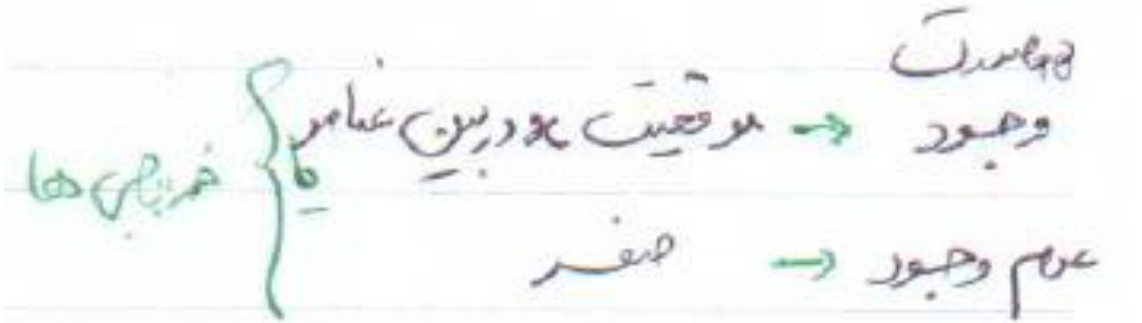
- خروجی :

جستجو (Search)

- ورودی ها:



- خروجی :

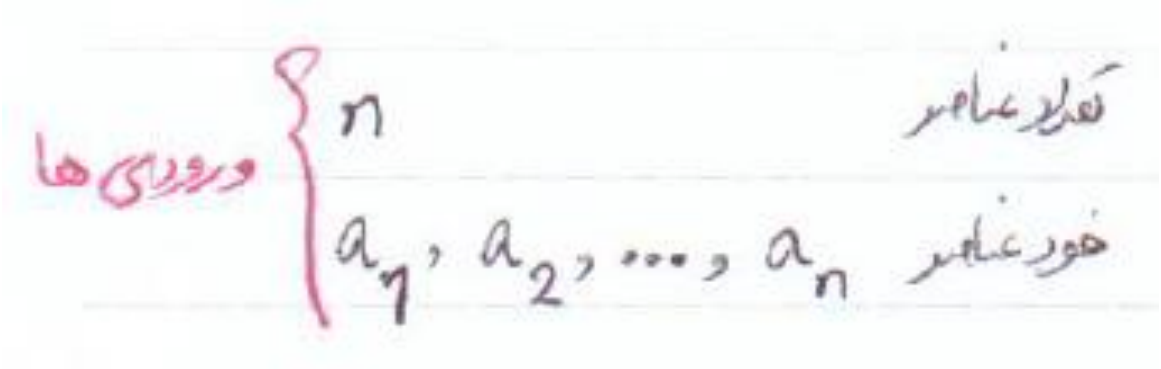


مرتب سازی (Sorting)

- ورودی ها:

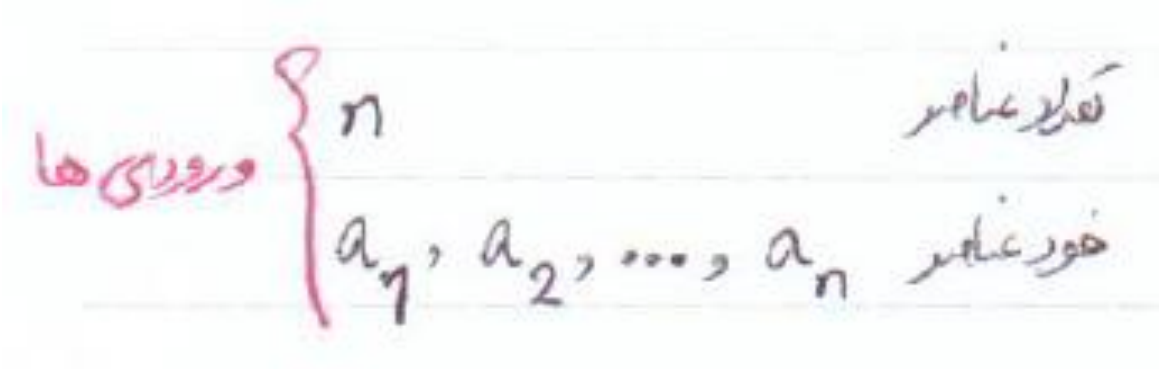
مرتب سازی (Sorting)

- ورودی ها:



مرتب سازی (Sorting)

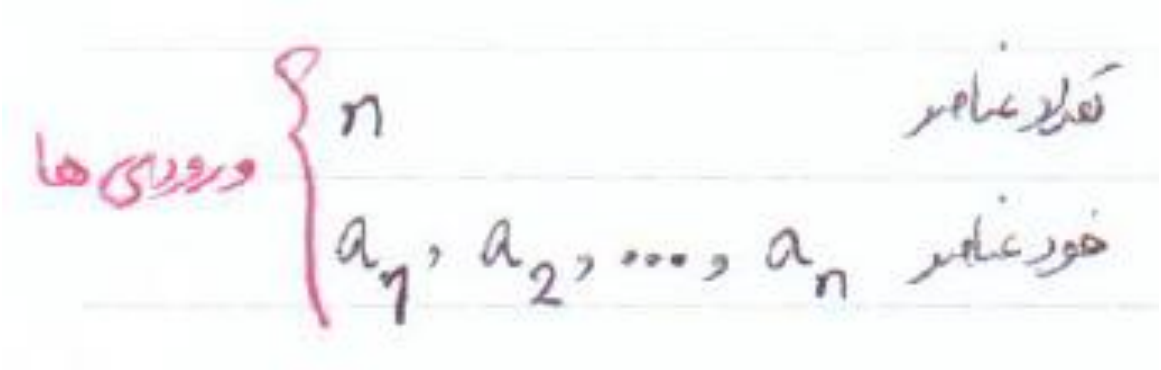
- ورودی ها:



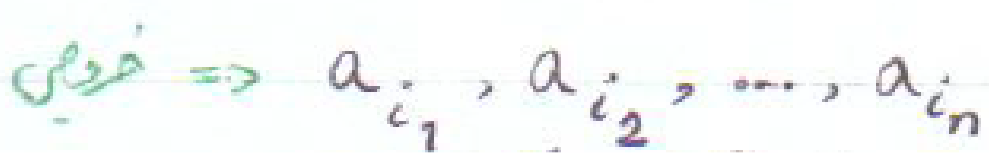
- خروجی :

مرتب سازی (Sorting)

- ورودی ها:

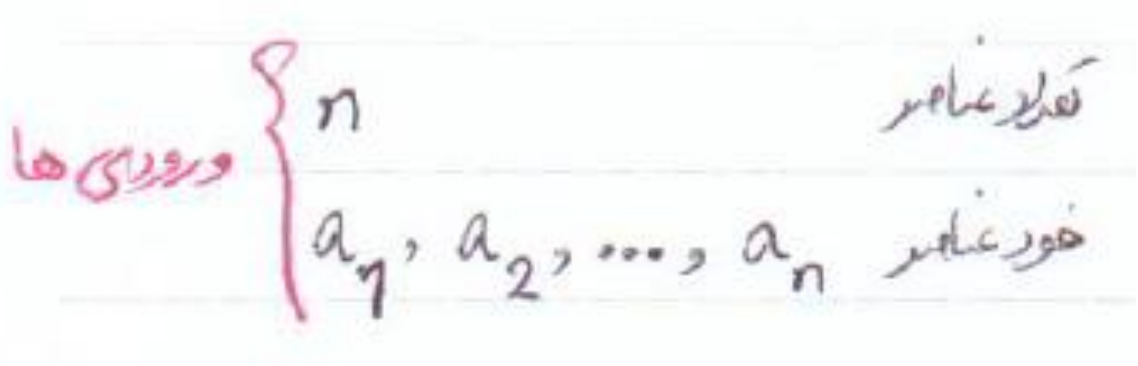


- خروجی :

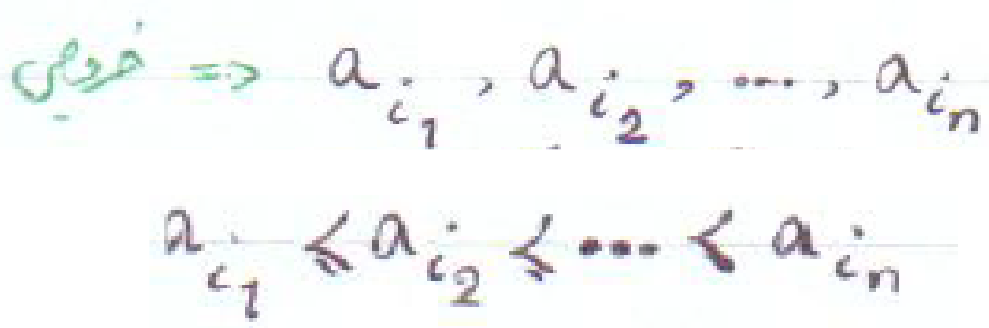


مرتب سازی (Sorting)

- ورودی ها:

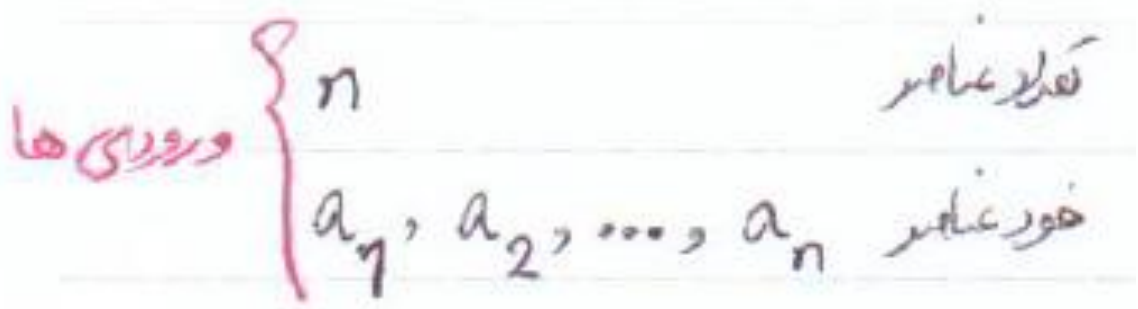


- خروجی :

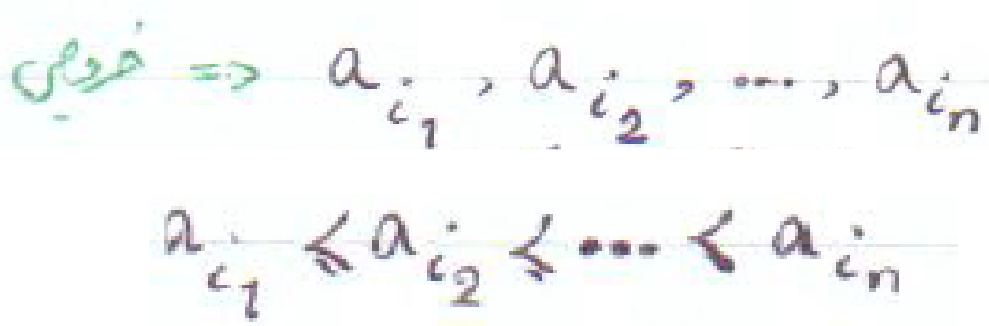


مرتب سازی (Sorting)

- ورودی ها:



- خروجی :



که i_1 تا i_n جایگشتی از یک تا n است

الگوریتم جستجوی خطی

```
index linear-search (n, A[1..n], x){
    i = 1;
    while (i <= n) and (A[i] ≠ x)
        i++;
    if (i > n)
        i = 0;
    return i; }
```

الگوریتم جستجوی دودویی

```
index Binary-search ( n , A[1..n] , x ) {
    l = 1 ;  H = n ;
    while ( l ≤ H )
    {
        M = ⌊ ( l + H ) / 2 ⌋ ;
        if  x = A [M]      return M ;
        if  x < A[M]       H = M - 1 ;
        else               l = M + 1 ;
    }
    return 0 ; }
```

مقایسه دو الگوریتم جستجو

```
index linear-search (n, A[1..n], x){
    i = 1;
    while (i ≤ n) and (A[i] ≠ x)
        i++;
    if (i > n)
        i = 0;
    return i; }
```

```
index Binary-search (n, A[1..n], x) {
    l = 1 ; H = n ;
    while (l ≤ H)
    {
        M = ⌊(l + H) / 2⌋ ;
        if x = A[M]      return M;
        if x < A[M]      H = M - 1 ;
        else             l = M + 1 ;
    }
    return 0 ; }
```

زمان اجرای الگوریتم (Running Time)

- تعداد دفعات اجرای عمل اصلی بر اساس اندازه ورودی برای یک نمونه(instance) از مساله.

(اگر تعداد دفعات اجرای عمل اصلی برای نمونه های یک مساله با اندازه ورودی یکسان، برابر باشد با T(n) نمایش داده می شود)

مجموع عناصر یک آرایه

```
number Add (n , A[1..n]) {
    sum = 0 ;
    for i=1 to n do
        sum = sum + A[i]
    return sum ;
}
```

مجموع عناصر یک آرایه

```
number Add (n , A[1..n]) {
    sum = 0 ;
    for i=1 to n do
        sum = sum + A[i]
    return sum ;
}
```

$T(n) = n$

کاهش زمان اجرای مساله مجموع عناصر یک آرایه (؟)

```
number Add (n , A[1..n]) {
    Sum = 0 ;
    For  i=1  to  n  do
        Sum = Sum + A[i]
    return  Sum ;
}
```

کاهش زمان اجرای مساله مجموع عناصر یک آرایه

```
number Add (n , A[1..n]) {
    sum = A[1];
    for i = 2 to n do
        sum = sum + A[i]
    return sum;
}
```

کاهش زمان اجرای مساله مجموع عناصر یک آرایه

```
number Add (n , A[1..n]) {
    Sum = A[1] ;
    for i = 2 to n do
        sum = sum + A[i]
    return sum ;
}
```

$\Rightarrow T(n) = n-1$

مجموع عناصر یک آرایه دو بعدی

```
number   Add2 (n , A[1..n , 1..n]) {
    Sum = 0 ;
    For   i=1   to   n   do
        For   j=1   to   n   do
            Sum = sum + A[i,j] ;
    return   sum ;
}
```

مجموع عناصر یک آرایه دو بعدی

اندازه ورودی؟؟؟

```
number   Add2 (n , A[1..n , 1..n]) {
    Sum = 0 ;
    For   i=1   to   n   do
        For   j=1   to   n   do
            Sum = sum + A[i,j] ;
    return   sum ;
}
```

مجموع عناصر یک آرایه دو بعدی

```
number  Add2 (n , A[1..n , 1..n]) {
        sum = 0 ;
        for  i=1  to  n  do
nxn     {
nx1     for  j=1  to  n  do
        {
        sum = sum + A[i,j] ;
        }
        }
        return  sum ;
}
```

مجموع عناصر یک آرایه دو بعدی

```
number  Add2 (n , A[1..n , 1..n]) {
        sum = 0 ;
        For   i=1   to   n   do
nxn     {
        nx1     For   j=1   to   n   do
        {       {       sum = sum + A[i,j] ;
        return  sum ;
}
```

$\Rightarrow T(n) = n^2$

مجموع عناصر یک آرایه دو بعدی غیر مربعی

```
number Add3 (n, m, A[1..n, 1..m]) {
    sum = 0;
    for i=1 to n do
        for j=1 to m do
            sum = sum + A[i,j]
    return sum;
}
```

مجموع عناصر یک آرایه دو بعدی غیر مربعی

اندازه ورودی؟؟؟

```
number Add3 (n, m, A[1..n, 1..m]) {
    sum = 0;
    for i=1 to n do
        for j=1 to m do
            sum = sum + A[i,j]
    return sum;
}
```

مجموع عناصر یک آرایه دو بعدی غیر مربعی

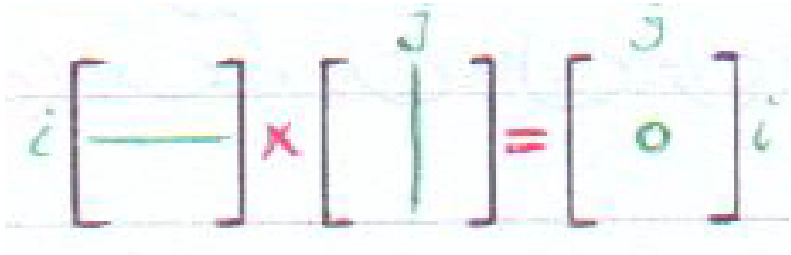
```
number Add3 (n, m, A[1..n, 1..m]) {
    sum = 0;
    for i=1 to n do
        for j=1 to m do
            sum = sum + A[i,j]
    return sum;
}
```

اندازه ورودی؟؟؟

$\Rightarrow \overline{T(n,m) = n \times m}$

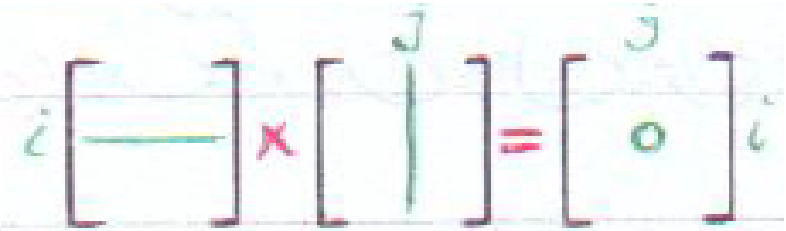
ضرب دو ماتریس

A B C



ضرب دو ماتریس

A B C

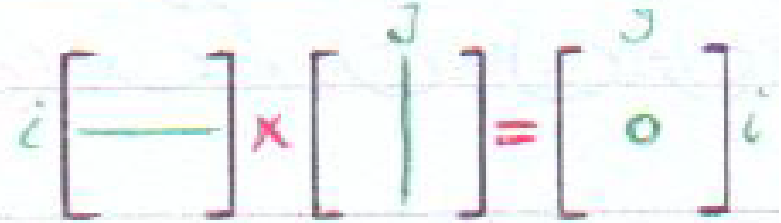



```
c[i,j] = 0;
for k=1 to n do
    c[i,j] = c[i,j] + A[i,k] * B[k,j];
```

```
Matrix Multiply (n, A[1..n, 1..n], B[1..n, 1..n]) {
    for i=1 to n do
        for j=1 to n do
        {
            c[i,j] = 0;
            for k=1 to n do
                c[i,j] = c[i,j] + A[i,k] * B[k,j];
        }
    return C; }
```

ضرب دو ماتریس

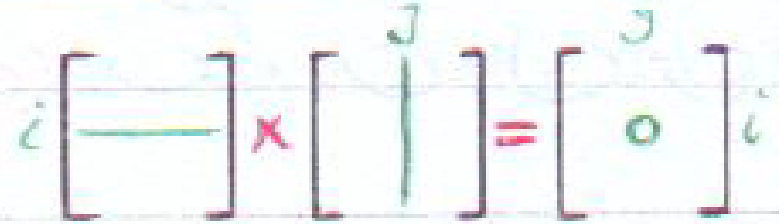
A B C



```
matrix Multiply (n, A[1..n, 1..n], B[1..n, 1..n]) {
    for i=1 to n do
        for j=1 to n do
        {
            c[i,j] = 0;
            for k=1 to n do
                c[i,j] = c[i,j] + A[i,k] * B[k,j];
        }
    return c; }
```

ضرب دو ماتریس

A B C

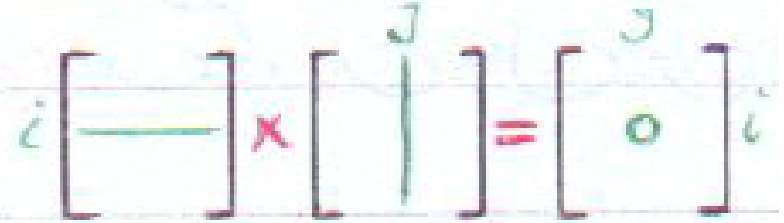


عمل اصلی؟؟؟؟

```
matrix Multiply (n, A[1..n, 1..n], B[1..n, 1..n]) {
    for i=1 to n do
        for j=1 to n do
        {
            c[i,j] = 0;
            for k=1 to n do
                c[i,j] = c[i,j] + A[i,k] * B[k,j];
        }
    return c; }
```

ضرب دو ماتریس

A B C



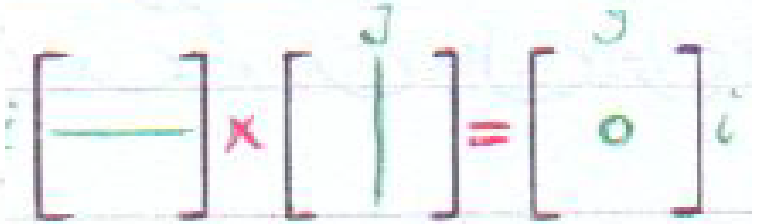
با فرض عمل اصلی=ضرب

ضرب دو ماتریس

```
Matrix Multiply (n, A[1..n, 1..n], B[1..n, 1..n]) {
    for i=1 to n do
        for j=1 to n do
        {
            c[i,j] = 0;
            for k=1 to n do
                c[i,j] = c[i,j] + A[i,k] * B[k,j];
        }
return C; }
```

n n×n n×1

A B C



با فرض عمل اصلی=ضرب

$$T(n) = n \times n \times n = n^3$$

ضرب دو ماتریس غیر مربعی

```
Matrix Multiply (n, m, p, A[1..n, 1..m], B[1..m, 1..p]) {
    for i = 1 to n do
        for j = 1 to p do
        {
            c[i, j] = 0;
            for k = 1 to m do
                c[i, j] = c[i, j] + A[i, k] * B[k, j];
        }
return c; }
```

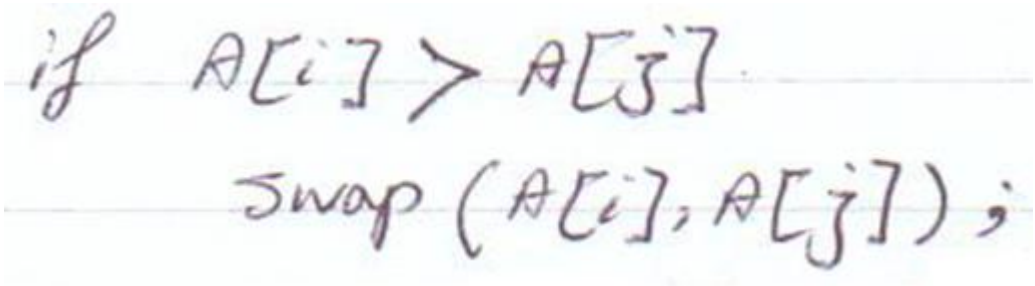
ضرب دو ماتریس غیر مربعی

```
Matrix Multiply (n, m, p, A[1..n, 1..m], B[1..m, 1..p]) {
    for i = 1 to n do
        for j = 1 to p do
        {
            c[i, j] = 0;
            for k = 1 to m do
                c[i, j] = c[i, j] + A[i, k] * B[k, j];
        }
return c; }
```

$T(n, m, p) = n \times m \times p$

مرتب سازی تعویضی(Exchange Sort)

مرتب سازی تعویضی(Exchange Sort)



مرتب سازی تعویضی

```
Exchange-sort (n, A[1..n]){
    for  i=1  to  n-1  do
        for  j=i+1  to  n  do
            if  A[i] > A[j]
                swap (A[i], A[j]);
}
```

مرتب سازی تعویضی

```
Exchange-sort (n, A[1..n]){
    for  i=1  to  n-1  do
        for  j=i+1  to  n  do
            if  A[i] > A[j]
                swap (A[i], A[j]);
}
```

- عمل اصلی: ؟

مرتب سازی تعویضی

```
Exchange-sort (n, A[1..n]){
    for  i=1  to  n-1  do
        for  j=i+1  to  n  do
            if  A[i] > A[j]
                swap (A[i], A[j]);
}
```

- عمل اصلی: مقایسه عناصر

مرتب سازی تعویضی

```
Exchange-sort (n, A[1..n]){
    for  i=1  to  n-1  do
        for  j=i+1  to  n  do
            if  A[i] > A[j]
                swap (A[i], A[j]);
}
```

- عمل اصلی: مقایسه عناصر

$(n-1)(n-(i+1)+1) = (n-1)(n-i)$!!!!!!

مرتب سازی تعویضی

```
Exchange-sort (n, A[1..n]){
    for  i=1  to  n-1  do
        for  j=i+1  to  n  do
            if  A[i] > A[j]
                swap (A[i], A[j]);
}
```

- عمل اصلی: مقایسه عناصر

$(n-1)(n-(i+1)+1) = (n-1)(n-i)$!!!!!!

مرتب سازی تعویضی

```
Exchange-sort (n, A[1..n]){
    for  i=1  to  n-1  do
        for  j=i+1  to  n  do
            if  A[i] > A[j]
                swap (A[i], A[j]);
}
```

تعداد دفعات حلقه خارجی	تعداد دفعات حلقه داخلی

مرتب سازی تعویضی

```
Exchange-sort (n, A[1..n]){
    for  i=1  to  n-1  do
        for  j=i+1  to  n  do
            if  A[i] > A[j]
                swap (A[i], A[j]);
}
```

تعداد تکرار جمله خ	ن
n-1	1

مرتب سازی تعویضی

```
Exchange-sort (n, A[1..n]){
    for i=1 to n-1 do
        for j=i+1 to n do
            if A[i] > A[j]
                swap (A[i], A[j]);
}
```

i	تعداد تکرار حلقه خ
1	n-1
2	n-2

مرتب سازی تعویضی

```
Exchange-sort (n, A[1..n]){
    for  i=1  to  n-1  do
        for  j=i+1  to  n  do
            if  A[i] > A[j]
                swap (A[i], A[j]);
}
```

i	تعداد تکرار حلقه ی
1	n-1
2	n-2
3	n-3

مرتب سازی تعویضی

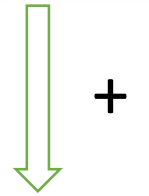
```
Exchange-sort (n, A[1..n]){
    for  i=1  to  n-1  do
        for  j=i+1  to  n  do
            if  A[i] > A[j]
                swap (A[i], A[j]);
}
```

i	تعداد تکرار حلقه ی
1	n-1
2	n-2
3	n-3
⋮	⋮
n-1	1

مرتب سازی تعویضی

```
Exchange-sort (n, A[1..n]){
    for  i=1  to  n-1  do
        for  j=i+1  to  n  do
            if  A[i] > A[j]
                swap (A[i], A[j]);
}
```

i	تعداد تکرار حلقه ی
1	n-1
2	n-2
3	n-3
⋮	⋮
n-1	1



+

مرتب سازی تعویضی

```
Exchange-sort (n, A[1..n]){
    for  i=1  to  n-1  do
        for  j=i+1  to  n  do
            if  A[i] > A[j]
                swap (A[i], A[j]);
}
```

i	تعداد تکرار حلقه ی
1	n-1
2	n-2
3	n-3
⋮	⋮
n-1	1

+

$$= \frac{n(n-1)}{2}$$

مرتب سازی تعویضی

```
Exchange-sort (n, A[1..n]){
    for  i=1  to  n-1  do
        for  j=i+1  to  n  do
            if  A[i] > A[j]
                swap (A[i], A[j]);
}
```

i	تعداد تکرار حلقه ی
1	n-1
2	n-2
3	n-3
⋮	⋮
n-1	1

+

$$T(n) = \frac{n(n-1)}{2}$$

نکته:

```
For  i = 1  to  n  do
    ....
    ....
    ....k(i)
    ....
    ....
```

نکته:

مرتب سازی تعویضی

```
Exchange-sort (n, A[1..n]){
    for i=1 to n-1 do
        for j=i+1 to n do
            if A[i] > A[j]
                swap (A[i], A[j]);
}
```

$\Leftarrow > n-i$

مرتب سازی تعویضی

```
Exchange-sort (n, A[1..n]){
    for i=1 to n-1 do
        for j=i+1 to n do
            if A[i] > A[j]
                swap (A[i], A[j]);
}
```

$\Leftarrow$ $n-i$

$$\Rightarrow T(n) = \sum_{i=1}^{n-1} (n-i)$$

مرتب سازی تعویضی

```
Exchange-sort (n, A[1..n]){
    for i=1 to n-1 do
        for j=i+1 to n do
            if A[i] > A[j]
                swap (A[i], A[j]);
}
```

$\Leftarrow$ $n-i$

$$\Rightarrow T(n) = \sum_{i=1}^{n-1} (n-i) = \frac{n(n-1)}{2}$$