فصل ۷
عملیات ماتریسی

سید ناصر رضوی

rezavi@iust.ac.ir

۱۳۹۰

معرفی

- فهرست مطالب:
  - محاسبه ترانهاده یک ماتریس
  - محاسبه حاصل ضرب دو ماتریس
  - محاسبه حاصل ضرب یک ماتریس در یک بردار

- برخی از کاربردها:
  - حل یک دستگاه معادلات (فصل ۸)
  - بازنمایی و عملیات گراف ها (فصل ۱۰)

محاسبه ترانهاده یک ماتریس

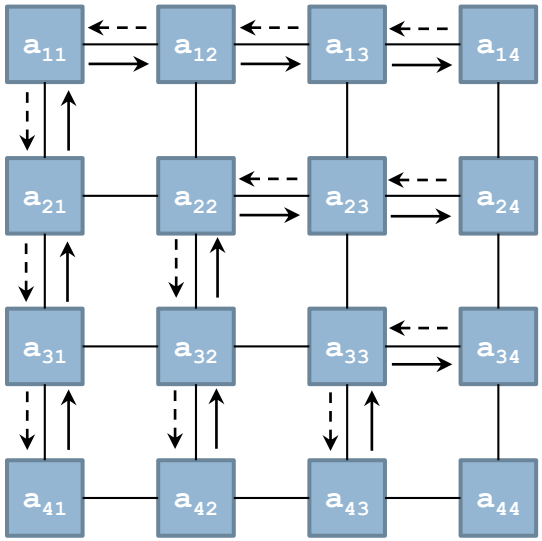
$$A = \begin{pmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ a_{31} & a_{32} & a_{33} & a_{34} \\ a_{41} & a_{42} & a_{43} & a_{44} \end{pmatrix} \xrightarrow{\text{ترانهاده}} A^T = \begin{pmatrix} a_{11} & a_{21} & a_{31} & a_{41} \\ a_{12} & a_{22} & a_{32} & a_{42} \\ a_{13} & a_{23} & a_{33} & a_{43} \\ a_{14} & a_{24} & a_{34} & a_{44} \end{pmatrix}$$

$O(n^2)$

```
procedure TRANSPOSE(A)
    for i = 2 to n do
        for j = 1 to i - 1 do
            a_ij ↔ a_ji
        end for
    end for
```

محاسبه ترانهاده یک ماتریس: اتصال توری

- اتصال توری شکل:
  - طبیعی ترین معماری برای انجام عملیات ماتریسی



- پردازنده P(i, j) سه ثبات دارد:
  - A(i, j)
    - برای ذخیره a_{ij} در ابتدا و a_{ji} در انتها
  - B(i, j)
    - برای ذخیره مقدار دریافتی از P(i, j+1) یا P(i-1, j)
  - C(i, j)
    - برای ذخیره مقدار دریافتی از P(i, j-1) یا P(i+1, j)

محاسبه ترانهاده یک ماتریس: اتصال توری


```
procedure MESH TRANSPOSE(A)
Step 1: do steps (1.1) and (1.2) in parallel
        (1.1) for i = 2 to n do in parallel
                  for j = 1 to i - 1 do in parallel
                      C(i-1, j) ← (a_ij, j, i)
                  end for
              end for

        (1.2) for i = 1 to n - 1 do in parallel
                  for j = i + 1 to n do in parallel
                      B(i, j-1) ← (a_ij, j, i)
                  end for
              end for.
```

محاسبه ترانهاده یک ماتریس: اتصال توری


```
Step 2: do steps (2.1) and (2.2) and (2.3) in parallel
        (2.1) for i = 2 to n do in parallel
                  for j = 1 to i - 1 do in parallel
                      while P(i, j) receives input from its neighbors do
                        (i) if (a_ij, m, k) is received from P(i+1, j)
                            then send it to P(i-1, j)
                            end if
                       (ii) if (a_ij, m, k) is received from P(i-1, j)
                            then if (i = m and j = k)
                                 then A(i, j) ← a_ij
                                 else send (a_ij, m, k) to P(i+1, j)
                                 end if
                            end if
                      end while
                  end for
              end for
```

پردازنده های واقع در مثلث پایینی

محاسبه ترانهاده یک ماتریس: اتصال توری


```
(2.2) for i = 1 to n do in parallel
          while P(i, i) receives input from its neighbors do
                (i) if (a_ij, m, k) is received from P(i+1, i)
                    then send it to P(i, i+1)
                    end if
               (ii) if (a_ij, m, k) is received from P(i, i+1)
                    then send it to P(i+1, i)
                    end if
          end while
      end for
```

پردازنده های واقع در قطر اصلی

محاسبه ترانهاده یک ماتریس: اتصال توری


```
(2.3) for i = 1 to n - 1 do in parallel
          for j = i + 1 to n do in parallel
              while P(i, j) receives input from its neighbors do
                (i) if (a_ij, m, k) is received from P(i, j+1)
                    then send it to P(i, j-1)
                    end if
                (ii) if (a_ij, m, k) is received from P(i, j-1)
                    then if (i = m and j = k)
                         then A(i, j) ← a_ij
                         else send (a_ij, m, k) to P(i, j+1)
                         end if
                    end if
              end while
          end for
      end for.
```

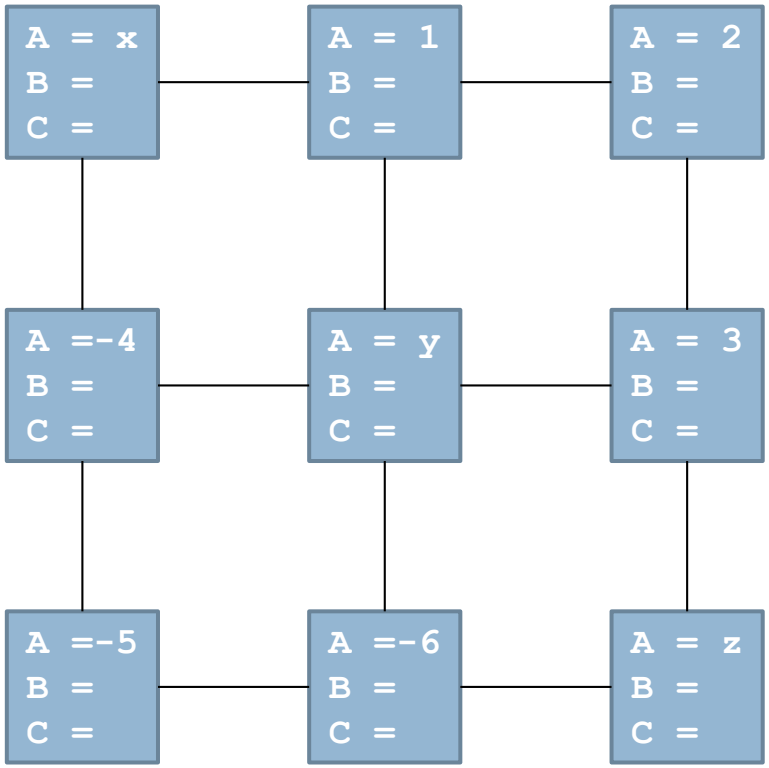
پردازنده های واقع در مثلث بالایی

تحلیل

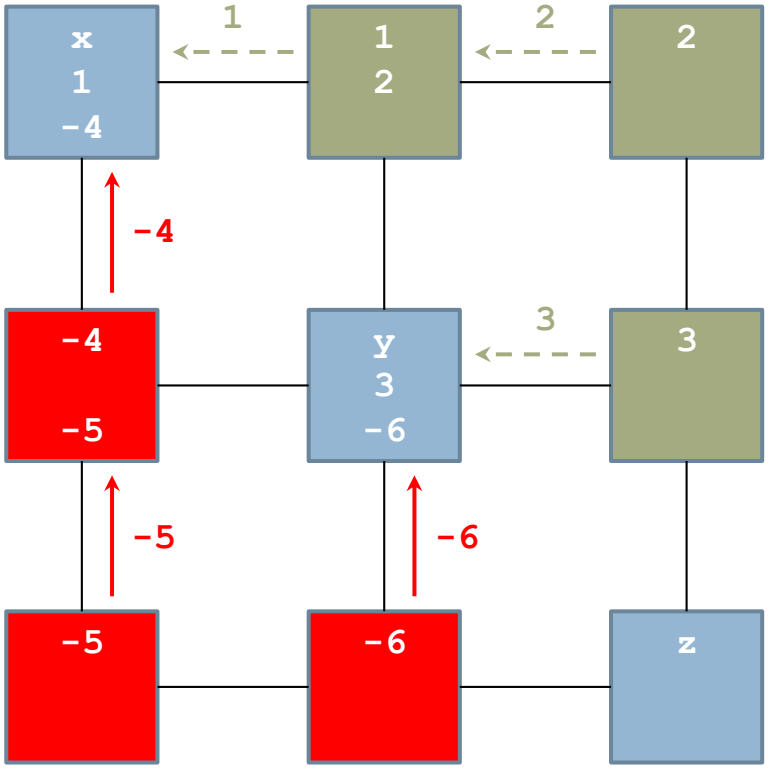
- زمان اجرا:
  - طولانی ترین مسیری که یک عنصر باید طی کند حداکثر $2(n-1)$ مرحله دارد، و بنابراین:
    - $t(n) = O(n)$
- هزینه:
  - $c(n) = t(n) \times P(n) = O(n^3)$

در نتیجه الگوریتم ارائه شده هزینه-بهینه نیست.

مثال

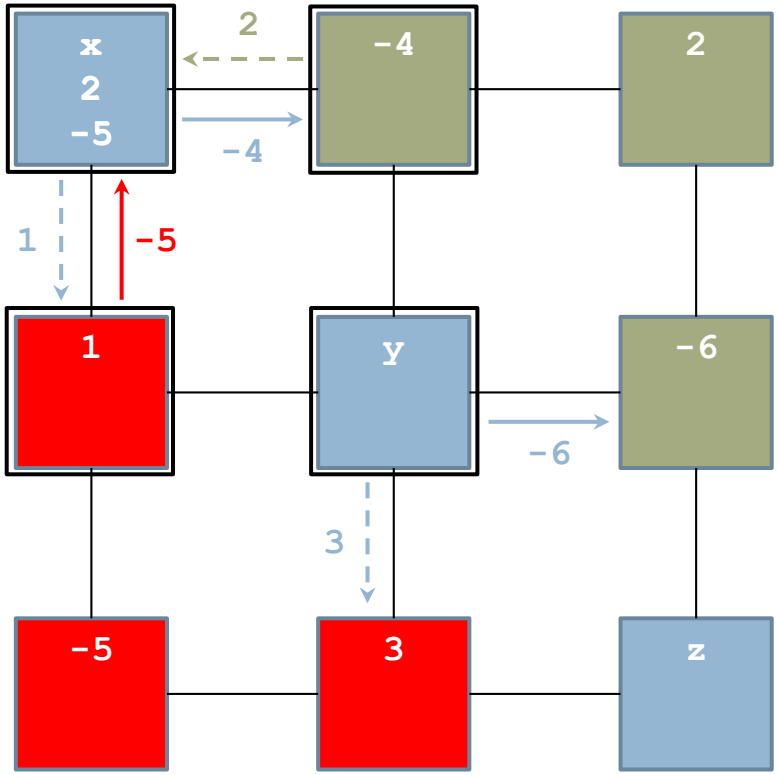


(الف) در ابتدا

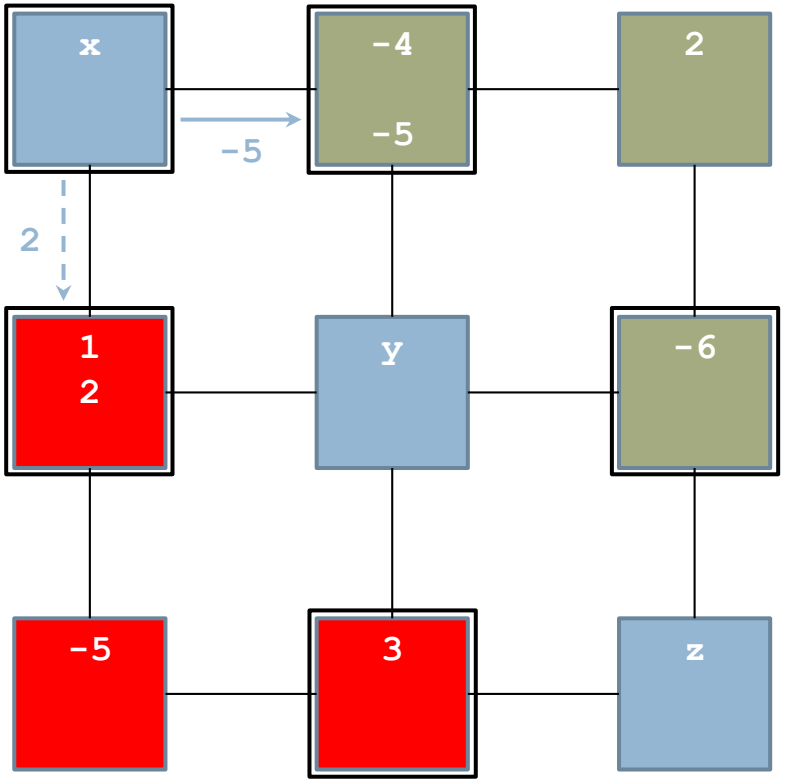


(ب) پس از مرحله ۱

مثال

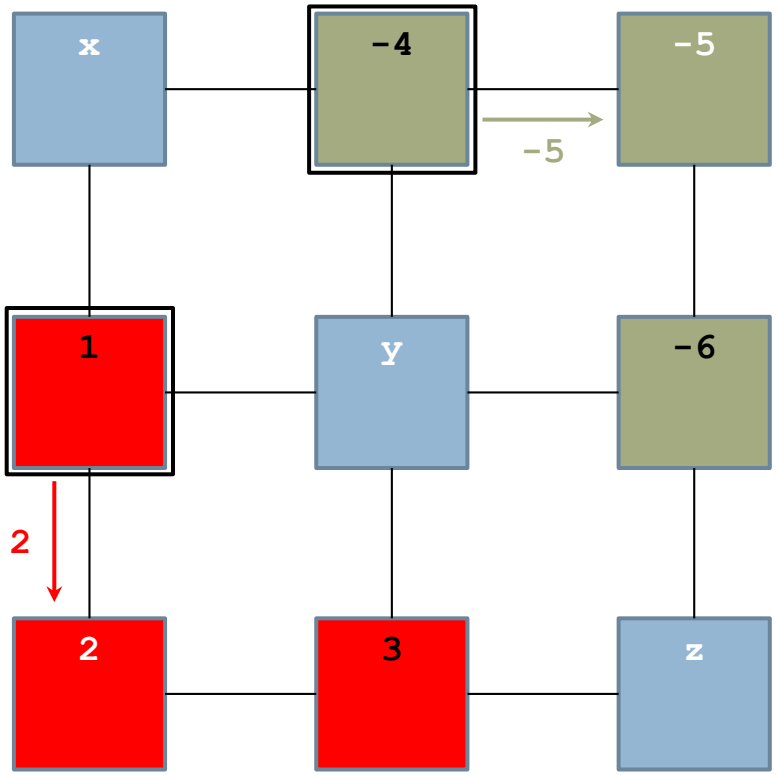


(ج) پس از تکرار اول مرحله ۲

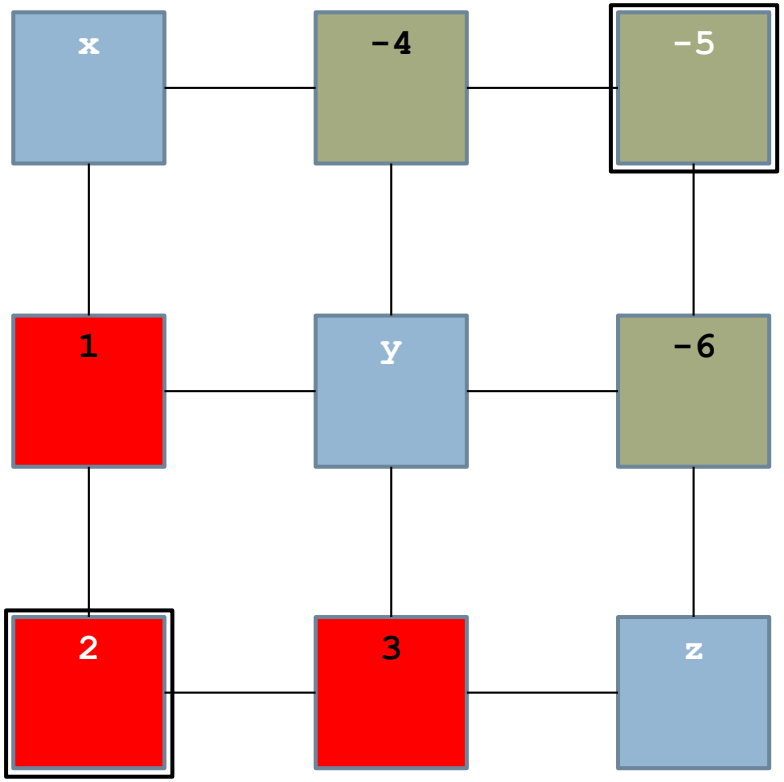


(د) پس از تکرار دوم مرحله ۲

مثال



(ه) پس از تکرار سوم مرحله ۲

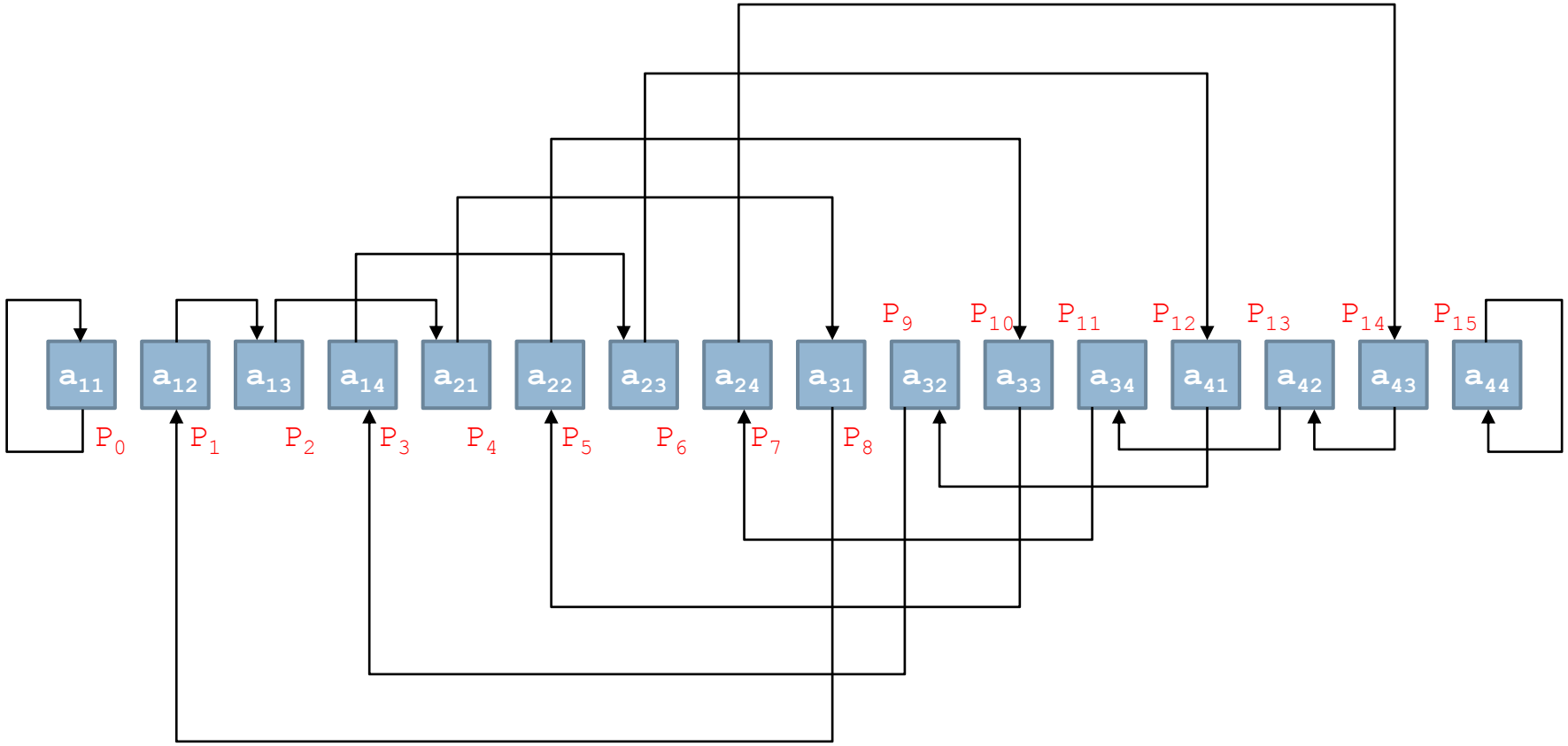


(و) پس از تکرار چهارم مرحله ۲

محاسبه ترانهاده یک ماتریس: اتصال در هم

- در این معماری، پردازنده P_k به پردازنده P_m متصل است اگر نمایش دودویی m با یک واحد شیفت چرخشی به سمت چپ از نمایش دودویی k به دست آید.
  - مثال:
- $P_{0000} \rightarrow P_{0000}$، $P_{0001} \rightarrow P_{0010}$، $P_{0010} \rightarrow P_{0100}$
- همچنین، عنصر a_{ij} در ابتدا در پردازنده P_k ذخیره می شود، به گونه ای که:
- $k = 2^q(i - 1) + (j - 1)$

محاسبه ترانهاده یک ماتریس: اتصال در هم



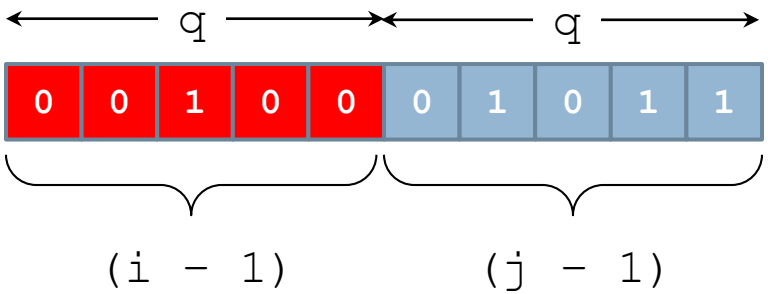
محاسبه ترانهاده یک ماتریس: اتصال در هم

- اکنون پردازنده ای با اندیس k را در نظر بگیرید به طوری که نمایش دودویی k شامل 2q بیت است. در این صورت اگر:

$$k = 2^q(i - 1) + (j - 1)$$

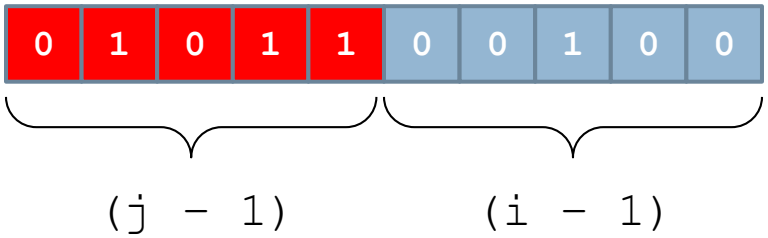
آنگاه q بیت پر ارزش تر k شامل (i - 1) و q بیت کم ارزش تر آن شامل (j - 1) است.

- مثال: (q = 5, i = 5, j = 12)



محاسبه ترانهاده یک ماتریس: اتصال در هم

- در این صورت پس از q بار انجام عمل shuffle (شیفت چرخشی به چپ)، عنصری که در ابتدا در P_k قرار داشت اکنون در P_s قرار دارد به گونه ای که:
- $s = 2^q(j - 1) + (i - 1)$



به عبارت دیگر، عنصر a_{ij} به مکان عنصر a_{ji} منتقل شده است.

محاسبه ترانهاده یک ماتریس: اتصال در هم

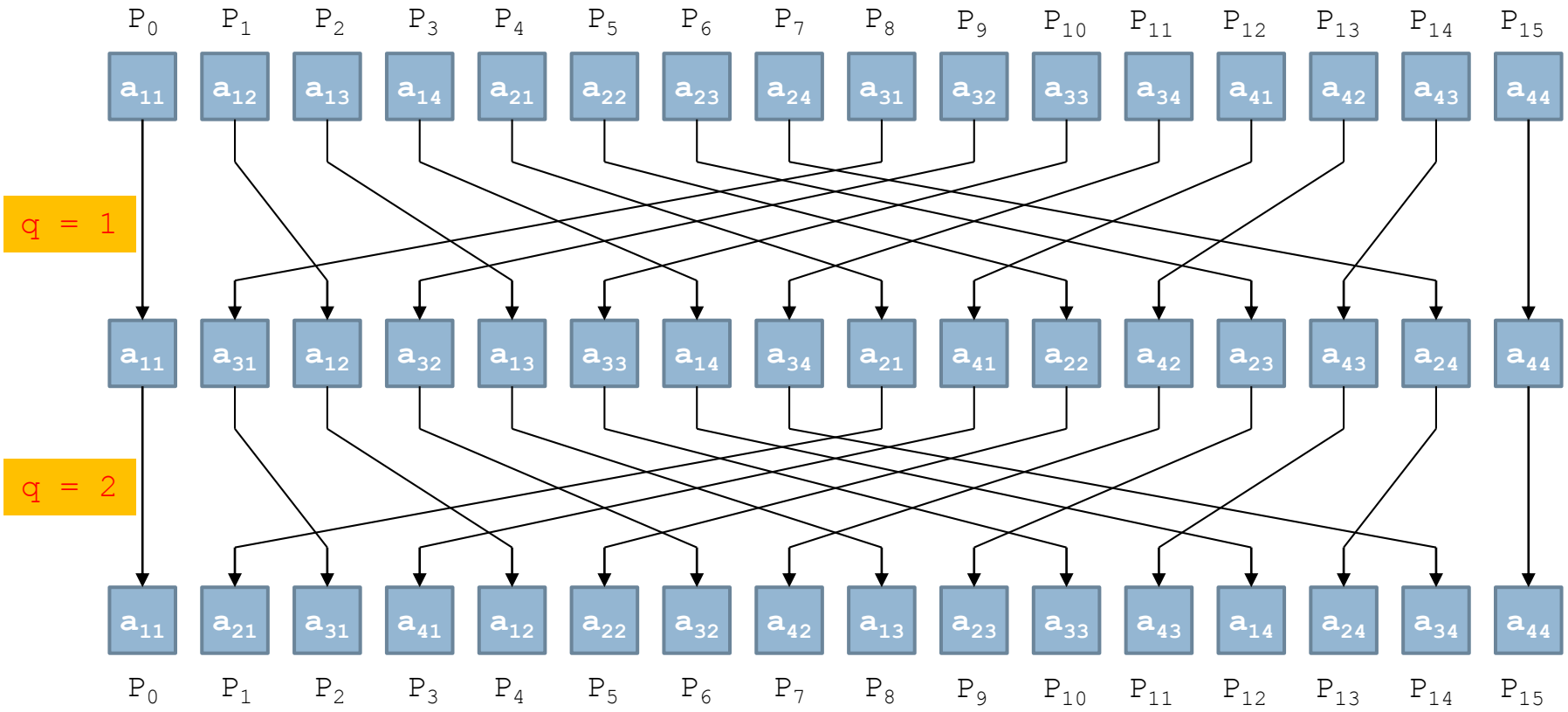

```
procedure SHUFFLE TRANSPOSE(A)
    for i = 1 to q do
        for k = 1 to 2^2q - 2 do in parallel
            P_k sends its data to P_{2k mod(2^2q -1)}
        end for
    end for.
```

O(log n)

$n = 2^{2q}$

- تحلیل:
  - $t(n) = O(\log n)$
  - $p(n) = n^2$
  - $c(n) = O(n^2 \log n)$
- در نتیجه الگوریتم فوق هزینه-بهینه نیست.

مثال


P0 P1 P2 P3 P4 P5 P6 P7 P8 P9 P10 P11 P12 P13 P14 P15
a11 a12 a13 a14 a21 a22 a23 a24 a31 a32 a33 a34 a41 a42 a43 a44
q = 1
a11 a31 a12 a32 a13 a33 a14 a34 a21 a41 a22 a42 a23 a43 a24 a44
q = 2
a11 a21 a31 a41 a12 a22 a32 a42 a13 a23 a33 a43 a14 a24 a34 a44
P0 P1 P2 P3 P4 P5 P6 P7 P8 P9 P10 P11 P12 P13 P14 P15

محاسبه ترانهاده یک ماتریس: مدل EREW

- ویژگی ها:
  - یک الگوریتم هزینه-بهینه به منظور محاسبه ترانهاده
  - با استفاده از $(n^2 - n)/2$ پردازنده
  - برای مدل SIMD SM EREW

- برای سادگی فرض می کنیم هر پردازنده دارای دو اندیس i و j است به گونه ای که:
- $2 \le i \le n, \quad 1 \le j \le i - 1,$
- در این الگوریتم پردازنده P_{ij} مکان دو عنصر a_{ij} و a_{ji} را با یکدیگر تعویض می کند.

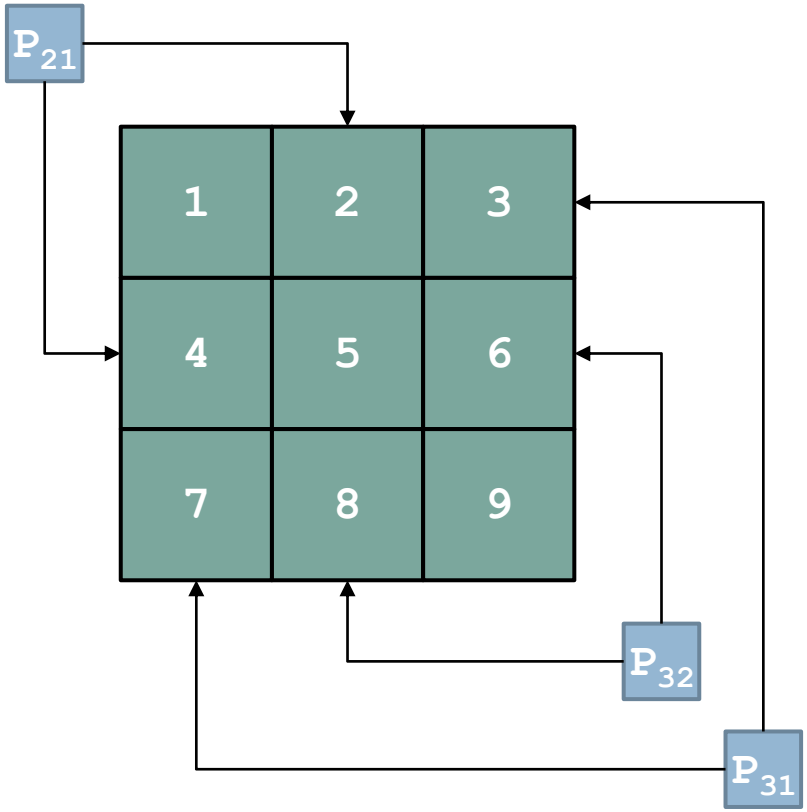
محاسبه ترانهاده یک ماتریس: مدل EREW

O(1)

```
procedure EREW TRANSPOSE(A)
    for i = 2 to n do in parallel
        for j = 1 to i - 1 do in parallel
            a_ij ↔ a_ji
        end for
    end for.
```

- تحلیل:
  - $t(n) = O(1)$
  - $p(n) = (n^2 - n)/2$
  - $c(n) = O(n^2)$
  - در نتیجه الگوریتم فوق هزینه-بهینه است.

مثال



ضرب ماتریس در ماتریس

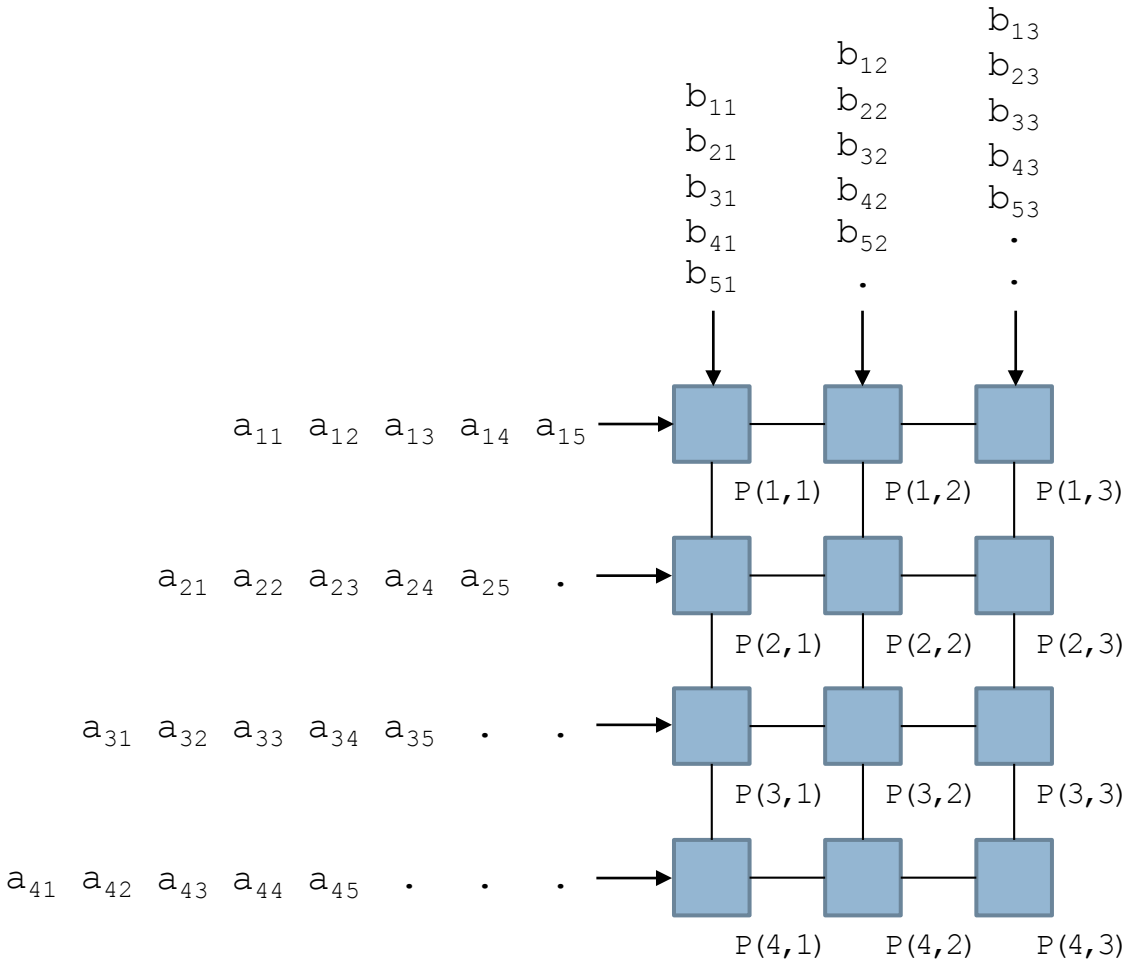
$$\begin{pmatrix} & & & \\ a_{i1} & a_{i2} & \cdots & \\ & & & \end{pmatrix}_{m\times n} \times \begin{pmatrix} & b_{1j} & \\ & b_{2j} & \\ & \vdots & \\ & b_{nj} & \end{pmatrix}_{n\times k} = \begin{pmatrix} & & \\ & c_{ij} & \\ & & \end{pmatrix}_{m\times k}$$

$$c_{ij} = \sum_{s=1}^{n} a_{is} \times b_{sj}, \qquad 1 \le i \le m, \quad 1 \le j \le k.$$

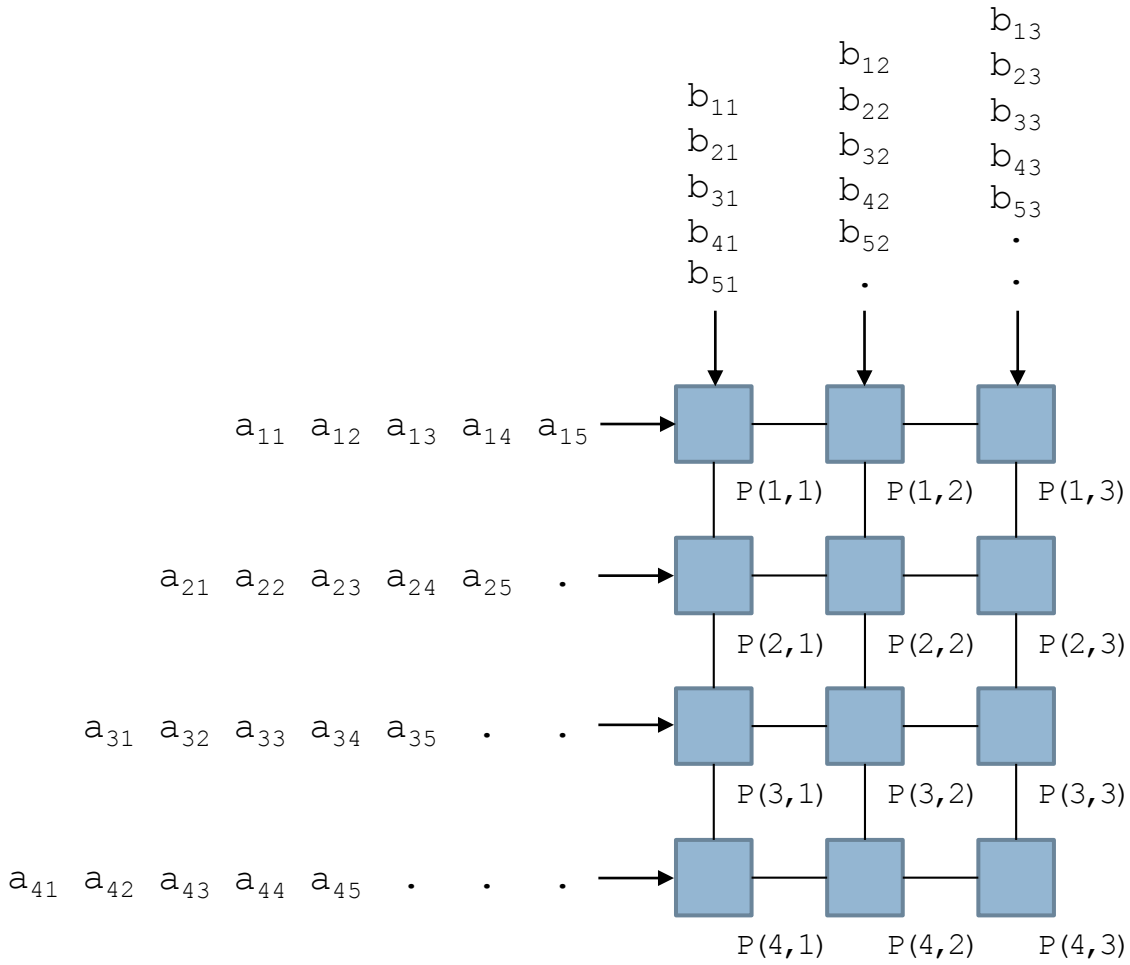
ضرب ماتریس در ماتریس: الگوریتم استاندارد


```
procedure MATRIX MULTIPLICATION(A, B, C)
    for i = 1 to m do
        for j = 1 to k do
            (1) c_ij ← 0
            (2) for s = 1 to n do
                    c_ij ← c_ij + (a_is × b_sj)
                end for
        end for
    end for.
```

ضرب ماتریس در ماتریس: ساختار توری



ضرب ماتریس در ماتریس: ساختار توری



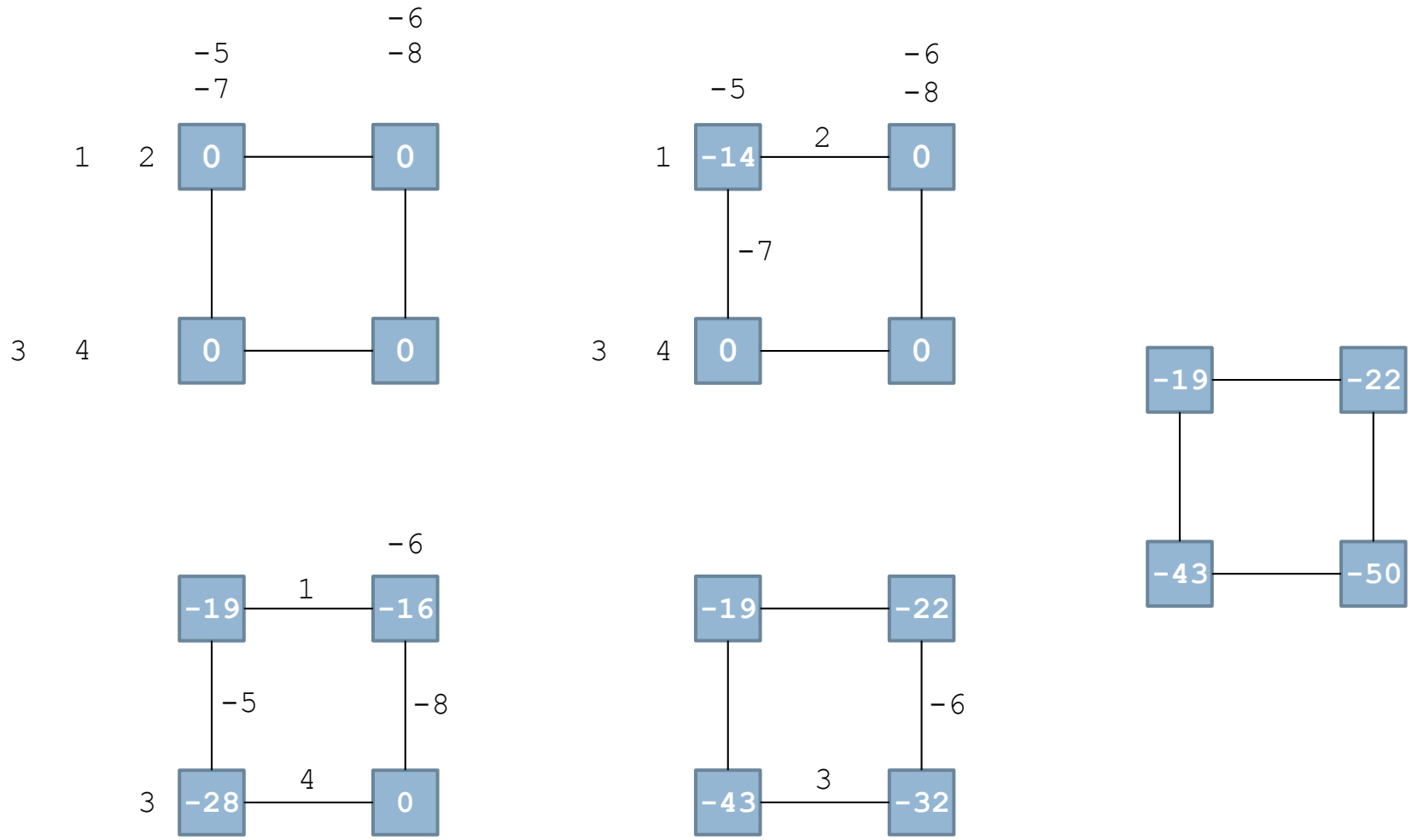
ضرب ماتریس در ماتریس: ساختار توری


```
procedure MESH MATRIX MULTIPLICATION(A, B, C)
    for i = 1 to m do in parallel
        for j = 1 to k do in parallel
          (1) c_ij ← 0
          (2) while P(i, j) receives two inputs a and b do
                (i) c_ij ← c_ij + (a × b)
               (ii) if i < m then send b to P(i+1, j)
                    end if
              (iii) if j < k then send a to P(i, j+1)
                    end if
            end while
        end for
    end for.
```

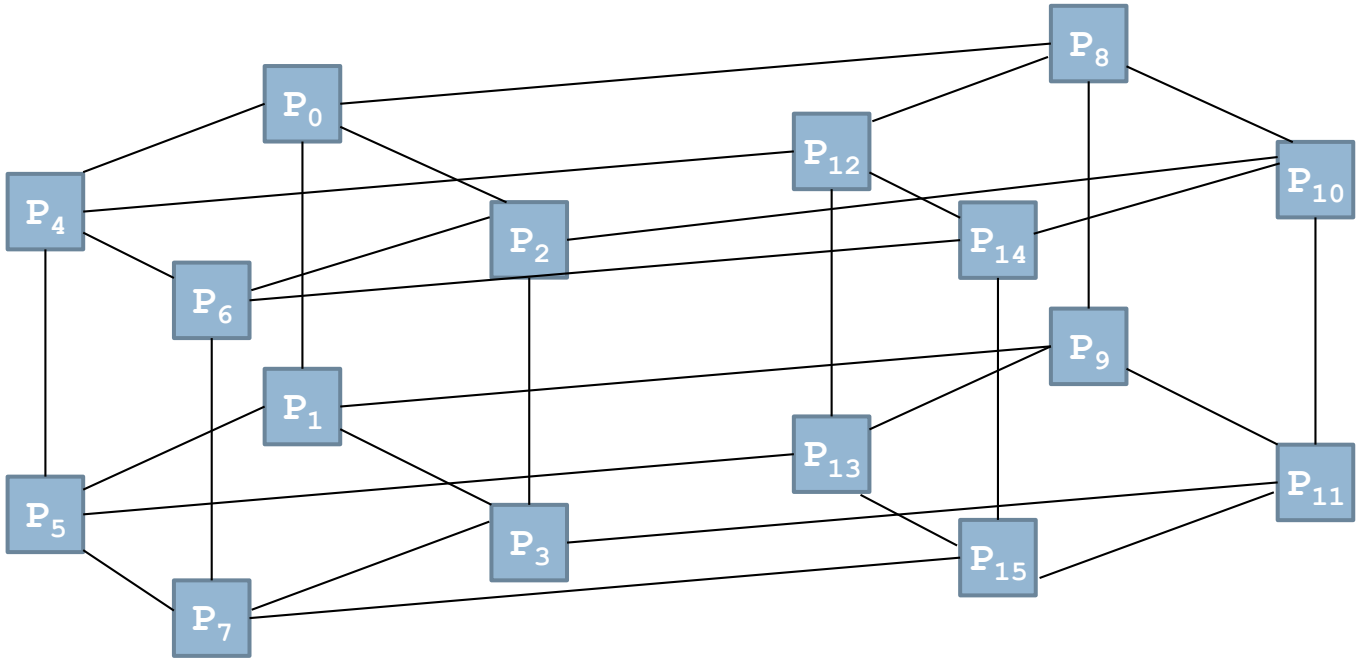
ضرب ماتریس در ماتریس: ساختار توری

- تحلیل:
  - عناصر a و b پس از $m + k + n - 2$ مرحله به پردازنده $P(m, k)$ می رسند.
  - از آنجا که پردازنده $P(m, k)$ آخرین پردازنده ای است که کار خود را به پایان می رساند، در نتیجه زمان اجرای الگوریتم نیز متناسب با همین مقدار یعنی $O(m + k + n)$ است.
  - با این فرض که m و k هر دو کوچکتر یا مساوی n هستند، خواهیم داشت:
    - $t(n) = O(n)$
    - $p(n) = O(n^2)$
    - $c(n) = O(n^3)$

مثال


-5
-7
-6
-8
1
2
0
0
3
4
0
0
-5
-6
-8
1
-14
2
0
-7
3
4
0
0
-19
-22
-43
-50
-6
-19
1
-16
-5
-8
3
-28
4
0
-19
-22
-6
-43
3
-32

ضرب ماتریس در ماتریس: ساختار مکعب



$$n = 2^q, \quad N = n^3 = 2^{3q}$$

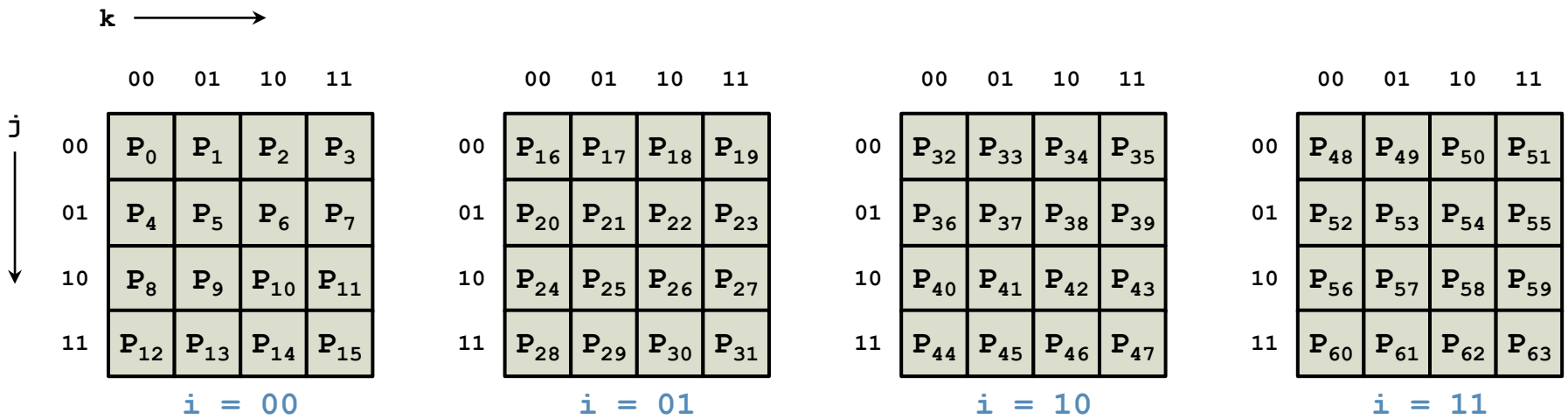
ضرب ماتریس در ماتریس: ساختار مکعب

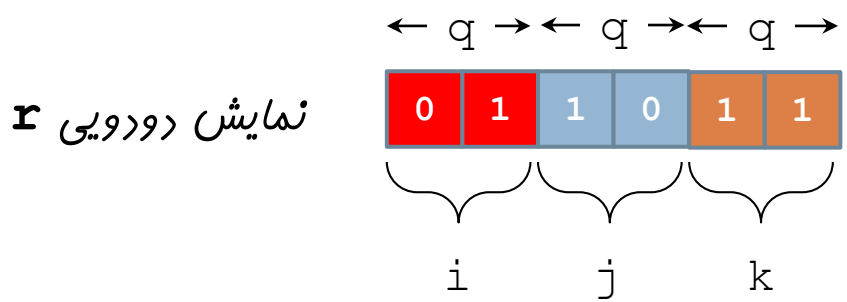
- در معماری مکعب می توان فرض کرد پردازنده ها در یک آرایه سه بعدی قرار گرفته اند.

- پردازنده P_r در مکان (i, j, k) قرار می گیرد، به گونه ای که:
- $r = 2^{2q} \times i + 2^q \times j + k = i \cdot n^2 + j \cdot n + k$

- همچنین هر پردازنده P_r سه ثبات به نام های A_r و B_r و C_r دارد.

ضرب ماتریس در ماتریس: ساختار مکعب





ضرب ماتریس در ماتریس: ساختار مکعب

- در ابتدا پردازنده P_s واقع در مکان **(0, j, k)** در ثبات های A_s و B_s خود به ترتیب عناصر a_{jk} و b_{jk} را ذخیره می کند.

- ثبات های سایر پردازنده ها شامل مقدار صفر است.

- در انتهای محاسبات، ثبات C_s مقدار c_{jk} را ذخیره می کند، به گونه ای که:

$$c_{jk} = \sum_{i=0}^{n-1} a_{ik} \times b_{kj}$$

ضرب ماتریس در ماتریس: ساختار مکعب

- مراحل الگوریتم:
  - مرحله ۱) توزیع عناصر ماتریس های **A** و **B** در میان n^3 پردازنده

$$A(i, j, k) = a_{ji}, \quad B(i, j, k) = b_{ik}$$

  - مرحله ۲) محاسبه حاصل ضرب زیر در هر پردازنده:

$$C(i, j, k) = A(i, j, k) \times B(i, j, k)$$

  - مرحله ۳) محاسبه مجموع زیر:

$$\sum_{i=0}^{n-1} C(i, j, k)$$

ضرب ماتریس در ماتریس: ساختار مکعب


```
procedure CUBE MATRIX MULTIPLICATION(A, B, C)
Step 1: for m = 3q - 1 downto 2q do
O(q)        for all r in {N, r_m = 0} do in parallel
                (1.1) A_r(m) = A_r
                (1.2) B_r(m) = B_r
            end for
        end for.
Step 2: for m = q - 1 downto 0 do
O(q)        for all r in {N, r_m = r_2q+m} do in parallel
                A_r(m) ← A_r
            end for
        end for.
```

ضرب ماتریس در ماتریس: ساختار مکعب


```
Step 3:  for m = 2q - 1 downto q do
O(q)         for all r in {N, r_m = r_(q+m)} do in parallel
                 B_r(m) ← B_r
             end for
         end for.
Step 4:  for r = 1 to N do in parallel
O(1)         C_r ← A_r × B_r
         end for.
Step 5:  for m = 2q to 3q - 1 do
O(q)         for r = 1 to N do in parallel
                 C_r ← C_r + C_r(m)
             end for
         end for.
```

مثال: $(q = 2)$

	00	01	10	11
00	17 -7	23 -25	27 -19	3 -5
01	9 -18	1 -30	14 -28	16 -12
10	31 -13	26 -21	22 -11	8 -32
11	15 -20	4 -2	10 -6	29 -24

i = 00

	00	01	10	11
00	0 0	0 0	0 0	0 0
01	0 0	0 0	0 0	0 0
10	0 0	0 0	0 0	0 0
11	0 0	0 0	0 0	0 0

i = 01

	00	01	10	11
00	0 0	0 0	0 0	0 0
01	0 0	0 0	0 0	0 0
10	0 0	0 0	0 0	0 0
11	0 0	0 0	0 0	0 0

i = 10

	00	01	10	11
00	0 0	0 0	0 0	0 0
01	0 0	0 0	0 0	0 0
10	0 0	0 0	0 0	0 0
11	0 0	0 0	0 0	0 0

i = 11

در ابتدا عناصر ماتریس های ورودی **A** و **B** در ثبات های $\mathbf{A_r}$ و $\mathbf{B_r}$ پردازنده های زیر بارگذاری می شوند:

$\mathbf{P_0}$, $\mathbf{P_1}$, $\mathbf{P_2}$, ..., $\mathbf{P_{15}}$

مثال: $(q = 2)$

	00	01	10	11
00	17 -7	23 -25	27 -19	3 -5
01	9 -18	1 -30	14 -28	16 -12
10	31 -13	26 -21	22 -11	8 -32
11	15 -20	4 -2	10 -6	29 -24

i = 00

	00	01	10	11
00	0 0	0 0	0 0	0 0
01	0 0	0 0	0 0	0 0
10	0 0	0 0	0 0	0 0
11	0 0	0 0	0 0	0 0

i = 01

	00	01	10	11
00	17 -7	23 -25	27 -19	3 -5
01	9 -18	1 -30	14 -28	16 -12
10	31 -13	26 -21	22 -11	8 -32
11	15 -20	4 -2	10 -6	29 -24

i = 10

	00	01	10	11
00	0 0	0 0	0 0	0 0
01	0 0	0 0	0 0	0 0
10	0 0	0 0	0 0	0 0
11	0 0	0 0	0 0	0 0

i = 11

m = 5

Step 1: **for** $m = 3q - 1$ **downto** $2q$ **do**

for all r **in** $\{N, r_m = 0\}$ **do in parallel**

(1.1) $A_{r(m)} = A_r$

(1.2) $B_{r(m)} = B_r$

end for

end for.

مثال: $(q = 2)$

	00	01	10	11
00	17 -7	23 -25	27 -19	3 -5
01	9 -18	1 -30	14 -28	16 -12
10	31 -13	26 -21	22 -11	8 -32
11	15 -20	4 -2	10 -6	29 -24

i = 00

	00	01	10	11
00	17 -7	23 -25	27 -19	3 -5
01	9 -18	1 -30	14 -28	16 -12
10	31 -13	26 -21	22 -11	8 -32
11	15 -20	4 -2	10 -6	29 -24

i = 01

	00	01	10	11
00	17 -7	23 -25	27 -19	3 -5
01	9 -18	1 -30	14 -28	16 -12
10	31 -13	26 -21	22 -11	8 -32
11	15 -20	4 -2	10 -6	29 -24

i = 10

	00	01	10	11
00	17 -7	23 -25	27 -19	3 -5
01	9 -18	1 -30	14 -28	16 -12
10	31 -13	26 -21	22 -11	8 -32
11	15 -20	4 -2	10 -6	29 -24

i = 11

m = 4

Step 1: **for** $m = 3q - 1$ **downto** $2q$ **do**

for all r **in** $\{N, r_m = 0\}$ **do in parallel**

(1.1) $A_{r(m)} = A_r$

(1.2) $B_{r(m)} = B_r$

end for

end for.

مثال: $(q = 2)$

	00	01	10	11
00	17 -7	23 -25	17 -19	23 -5
01	9 -18	1 -30	9 -28	1 -12
10	31 -13	26 -21	31 -11	26 -32
11	15 -20	4 -2	15 -6	4 -24

i = 00

	00	01	10	11
00	17 -7	23 -25	17 -19	23 -5
01	9 -18	1 -30	9 -28	1 -12
10	31 -13	26 -21	31 -11	26 -32
11	15 -20	4 -2	15 -6	4 -24

i = 01

	00	01	10	11
00	27 -7	3 -25	27 -19	3 -5
01	14 -18	16 -30	14 -28	16 -12
10	22 -13	8 -21	22 -11	8 -32
11	10 -20	29 -2	10 -6	29 -24

i = 10

	00	01	10	11
00	27 -7	3 -25	27 -19	3 -5
01	14 -18	16 -30	14 -28	16 -12
10	22 -13	8 -21	22 -11	8 -32
11	10 -20	29 -2	10 -6	29 -24

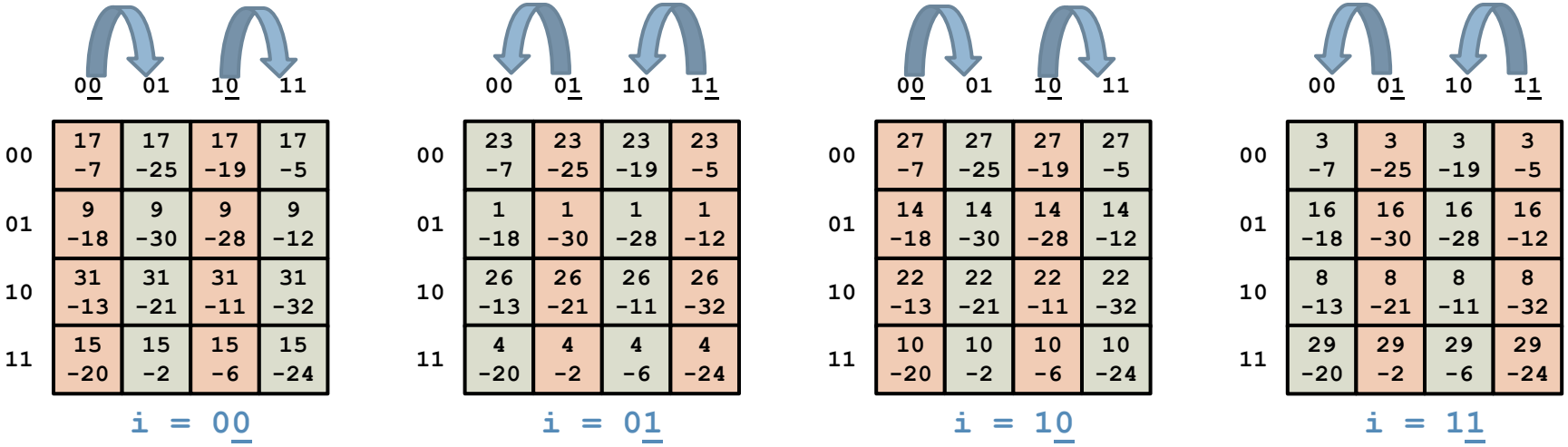
i = 11

m = 1

$r_1 = r_5$

```
Step 2: for m = q - 1 downto 0 do
            for all r in {N, r_m = r_(2q+m)} do in parallel
                A_r(m) ← A_r
            end for
        end for.
```

مثال: $(q = 2)$



m = 0

$r_0 = r_4$

```
Step 2: for m = q - 1 downto 0 do
            for all r in {N, r_m = r_(2q+m)} do in parallel
                A_r(m) ← A_r
            end for
        end for.
```

مثال: $(q = 2)$

	00	01	10	11
00	17 -7	17 -25	17 -19	17 -5
01	9 -18	9 -30	9 -28	9 -12
10	31 -7	31 -25	31 -19	31 -5
11	15 -18	15 -30	15 -28	15 -12

i = 00

	00	01	10	11
00	23 -7	23 -25	23 -19	23 -5
01	1 -18	1 -30	1 -28	1 -12
10	26 -7	26 -25	26 -19	26 -5
11	4 -18	4 -30	4 -28	4 -12

i = 01

	00	01	10	11
00	27 -13	27 -21	27 -11	27 -32
01	14 -20	14 -2	14 -6	14 -24
10	22 -13	22 -21	22 -11	22 -32
11	10 -20	10 -2	10 -6	10 -24

i = 10

	00	01	10	11
00	3 -13	3 -21	3 -11	3 -32
01	16 -20	16 -2	16 -6	16 -24
10	8 -13	8 -21	8 -11	8 -32
11	29 -20	29 -2	29 -6	29 -24

i = 11

m = 3

$r_3 = r_5$

```
Step 3: for m = 2q - 1 downto q do
            for all r in {N, r_m = r_(q+m)} do in parallel
                B_r(m) ← B_r
            end for
        end for.
```

مثال: $(q = 2)$

	00	01	10	11
00	17 -7	17 -25	17 -19	17 -5
01	9 -7	9 -25	9 -19	9 -5
10	31 -7	31 -25	31 -19	31 -5
11	15 -7	15 -25	15 -19	15 -5

i = 00

	00	01	10	11
00	23 -18	23 -30	23 -28	23 -12
01	1 -18	1 -30	1 -28	1 -12
10	26 -18	26 -30	26 -28	26 -12
11	4 -18	4 -30	4 -28	4 -12

i = 01

	00	01	10	11
00	27 -13	27 -21	27 -11	27 -32
01	14 -13	14 -21	14 -11	14 -32
10	22 -13	22 -21	22 -11	22 -32
11	10 -13	10 -21	10 -11	10 -32

i = 10

	00	01	10	11
00	3 -20	3 -2	3 -6	3 -24
01	16 -20	16 -2	16 -6	16 -24
10	8 -20	8 -2	8 -6	8 -24
11	29 -20	29 -2	29 -6	29 -24

i = 11

m = 2

$r_2 = r_4$

```
Step 3: for m = 2q - 1 downto q do
            for all r in {N, r_m = r_(q+m)} do in parallel
                B_r(m) ← B_r
            end for
        end for.
```

مثال: $(q = 2)$

	00	01	10	11
00	-119	-425	-323	-85
01	-63	-225	-171	-45
10	-217	-775	-589	-155
11	-105	-375	-285	-75

i = 00

	00	01	10	11
00	-414	-690	-644	-276
01	-18	-30	-28	-12
10	-468	-780	-728	-312
11	-72	-120	-112	-48

i = 01

	00	01	10	11
00	-351	-567	-297	-864
01	-182	-294	-154	-448
10	-286	-462	-242	-704
11	-130	-210	-110	-320

i = 10

	00	01	10	11
00	-60	-6	-18	-72
01	-320	-32	-96	-384
10	-160	-16	-48	-192
11	-580	-58	-174	-696

i = 11

Step 4: **for** $r = 1$ **to** N **do in parallel**

$C_r \leftarrow A_r \times B_r$

end for.

مثال: $(q = 2)$

	00	01	10	11
00	-533	-1115	-976	-361
01	-81	-255	-199	-57
10	-685	-1555	-1317	-467
11	-177	-495	-397	-123

i = 00

	00	01	10	11
00	-533	-1115	-976	-361
01	-81	-255	-199	-57
10	-685	-1555	-1317	-467
11	-177	-495	-397	-123

i = 01

	00	01	10	11
00	-411	-573	-315	-936
01	-502	-326	-250	-832
10	-446	-478	-290	-896
11	-710	-268	-284	-1016

i = 10

	00	01	10	11
00	-411	-573	-315	-936
01	-502	-326	-250	-832
10	-446	-478	-290	-896
11	-710	-268	-284	-1016

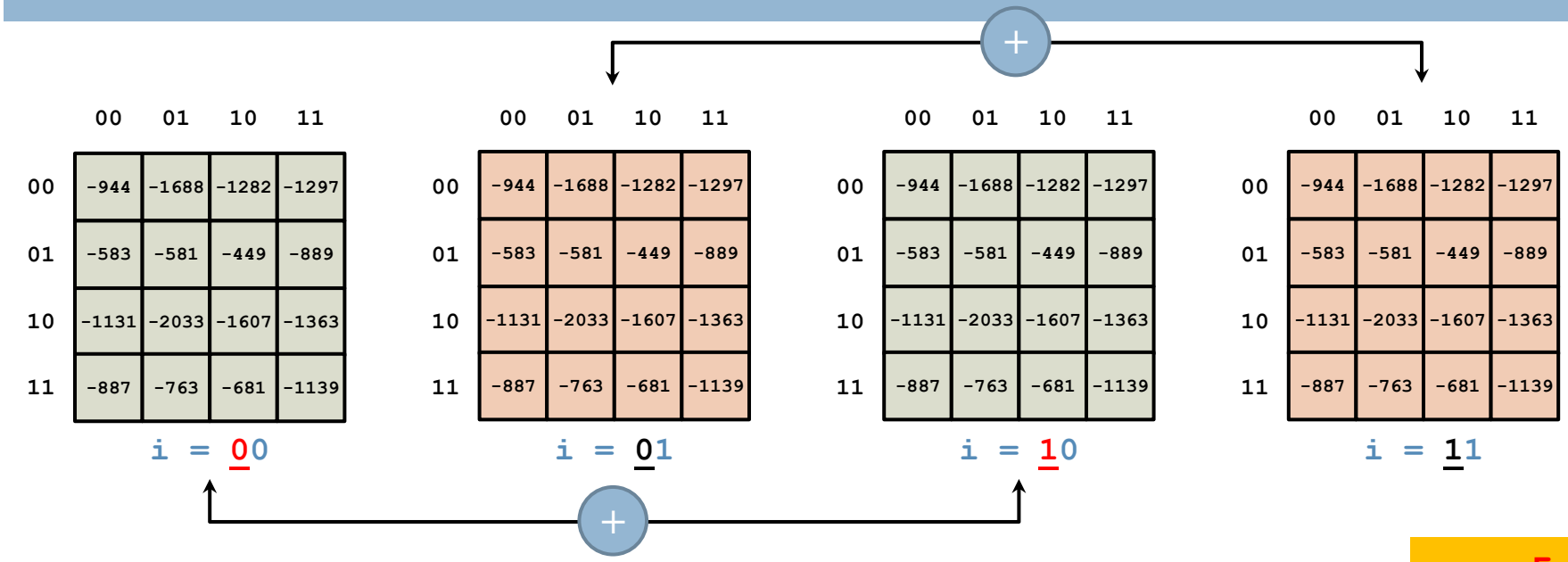
i = 11

+ +

m = 4

```
Step 5: for m = 2q to 3q - 1 do
            for r = 1 to N do in parallel
                C_r ← C_r + C_r(m)
            end for
        end for.
```

مثال: $(q = 2)$



m = 5

```
Step 5: for m = 2q to 3q - 1 do
            for r = 1 to N do in parallel
                C_r ← C_r + C_r(m)
            end for
        end for.
```

ضرب ماتریس در ماتریس: مدل CRCW


```
procedure CRCW MATRIX MULTIPLICATION(A, B, C)
    for i = 1 to m do in parallel
        for j = 1 to k do in parallel
            for s = 1 to n do in parallel
                (1) c_ij ← 0
                (2) c_ij ← a_is × b_sj
            end for
        end for
    end for.
```

تحلیل

- زمان اجرا:
  - $t(n) = O(1)$
- تعداد پردازنده ها:
  - $p(n) = n^3$
- هزینه:
  - $c(n) = n^3 \times O(1) = O(n^3)$

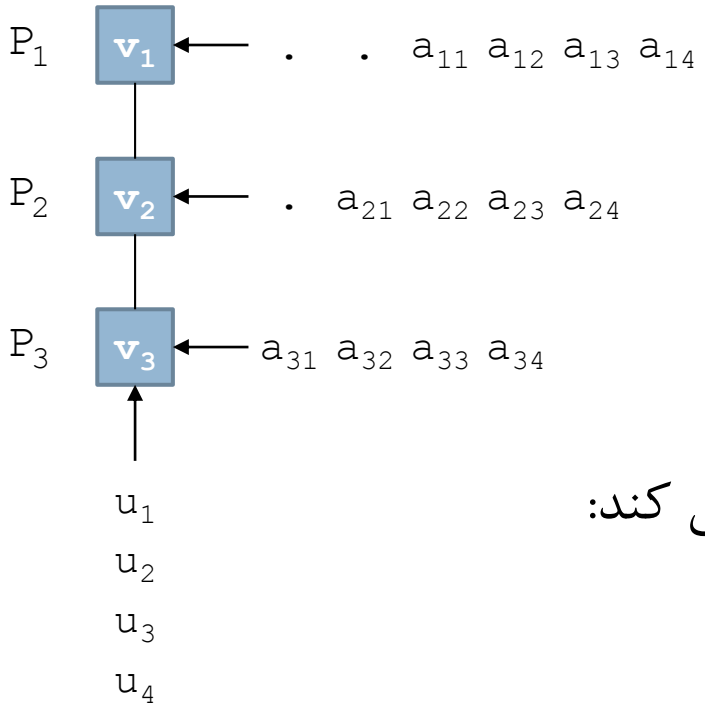
ضرب ماتریس در بردار

$$\begin{pmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ a_{31} & a_{32} & a_{33} & a_{34} \end{pmatrix}_{m\times n} \times \begin{pmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \end{pmatrix}_{n\times 1} = \begin{pmatrix} v_1 \\ v_2 \\ v_3 \end{pmatrix}_{m\times 1}$$

$$v_j = \sum_{k=1}^{n} a_{jk} \times u_k , \qquad 1 \le j \le m$$

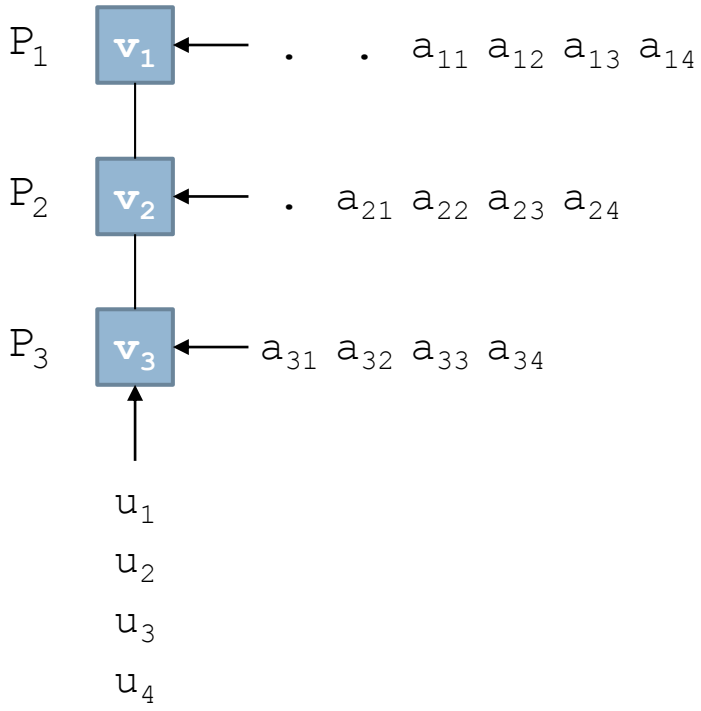
- واضح است که این مسأله یک حالت خاص از مسأله ضرب ماتریس در ماتریس است. اما در اینجا به منظور نمایش نحوه استفاده از دو شبکه اتصالی زیر، این مسأله را به صورت جداگانه بررسی می کنیم:
  - شبکه اتصالی آرایه خطی
  - شبکه اتصالی درختی

ضرب ماتریس در بردار: آرایه خطی



- هر پردازنده P_i سه ثبات دارد: **a** و **u** و **v**
- هر زمان که P_i دو ورودی a_{ij} و u_j را دریافت می کند:
  - a_{ij} را در **a** و u_j را در **u** ذخیره می کند،
  - **a** را در **u** ضرب می کند،
  - حاصل را با v_i جمع می کند،
  - و u_j را به پرادزنده P_{i-1} ارسال می کند.

ضرب ماتریس در بردار: آرایه خطی



ضرب ماتریس در بردار: آرایه خطی

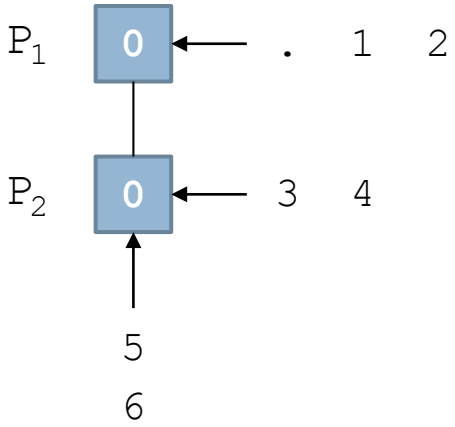

```
procedure LINEAR MV MULTIPLICATION (A, U, V)
    for i = 1 to m do in parallel
        (1) v_i ← 0
        (2) while P_i receives two input a, u
            (2.1) v_i ← v_i + (a × u)
            (2.2) if (i > 1) then send u to P_(i-1)
                  end if
            end while
    end for.
```

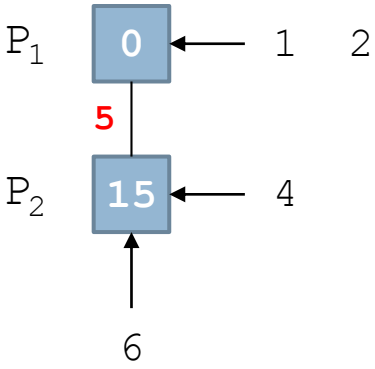
تحلیل

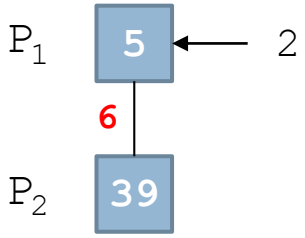
- تحلیل:
  - عنصر **a** پس از $m + n - 1$ مرحله به پردازنده P_1 می رسد و چون این پردازنده آخرین پردازنده ای است که کار خود را به پایان می رساند، خواهیم داشت:
    - $t(n) = O(m + n - 1)$
  - اگر فرض کنیم $m \leq n$، آنگاه:
    - $t(n) = O(n)$
    - $p(n) = O(m) = O(n)$
    - $c(n) = O(n^2)$

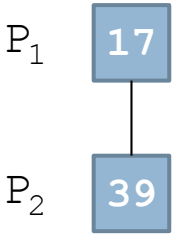
مثال

$$\begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix} \times \begin{pmatrix} 5 \\ 6 \end{pmatrix} = \begin{pmatrix} 17 \\ 39 \end{pmatrix}$$





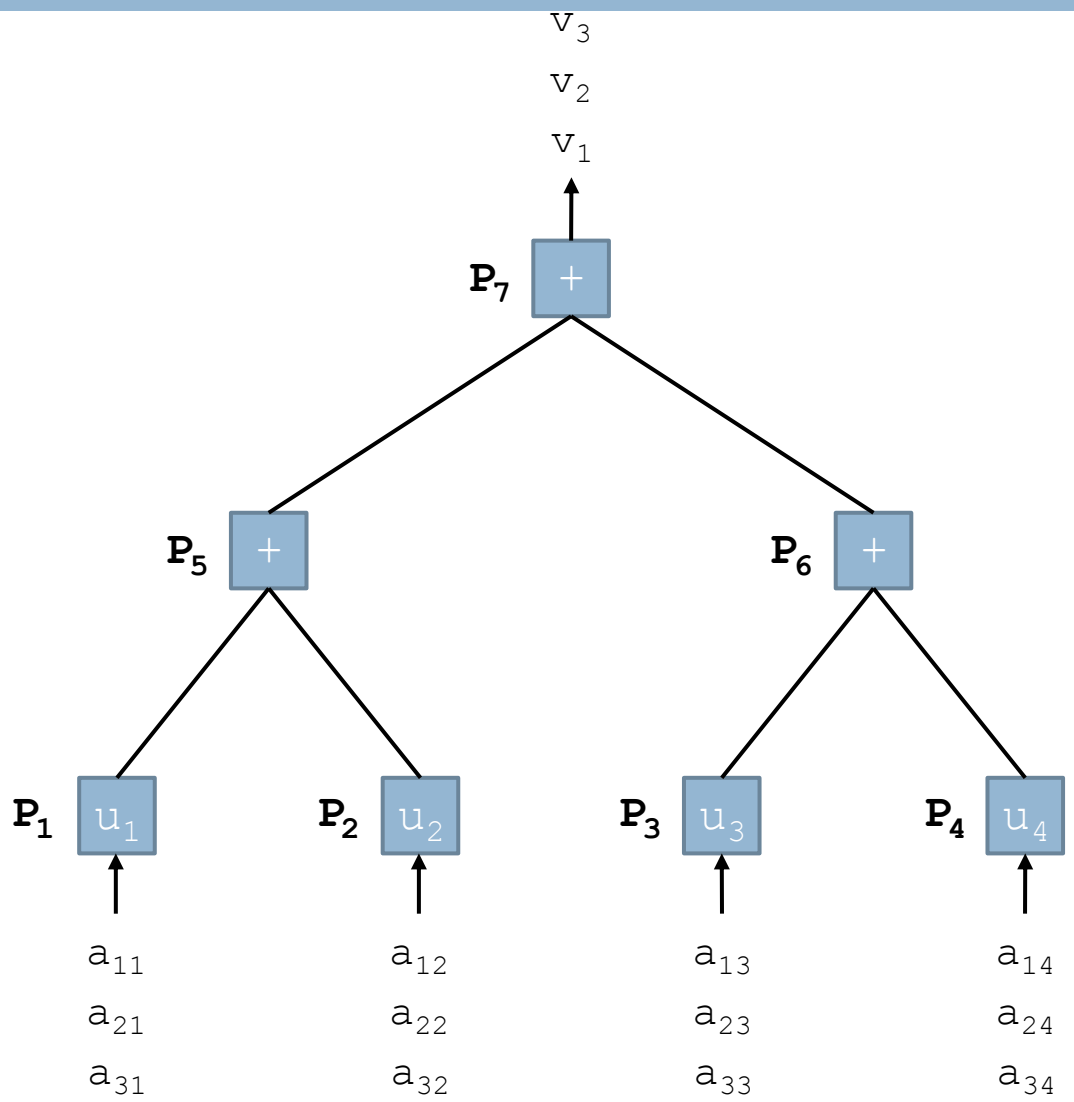




ضرب ماتریس در بردار: شبکه اتصالی درختی

- هدف:
  - کاهش تعداد مراحل از $m + n - 1$ به $m + \log n - 1$
  - با استفاده از ساختار درختی

ضرب ماتریس در بردار: شبکه اتصالی درختی



ضرب ماتریس در بردار: شبکه اتصالی درختی

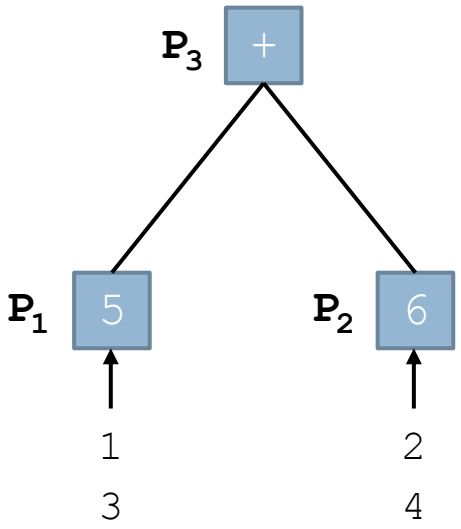

```
procedure TREE MV MULTIPLICATION (A, U, V)
   do step 1 and 2 in parallel
      (1) for i = 1 to n do in parallel
             for j = 1 to m do
             (1.1) compute u_i × a_ij
             (1.2) send result to parent
             end for
          end for
      (2) for i = n + 1 to 2n - 1 do in parallel
             while P_i receives two input do
             (2.1) compute the sum of two inputs
             (2.2) if (i < 2n - 1) then send the result to parent
                                   else produce the result as output
                   end if
             end while
          end for.
```

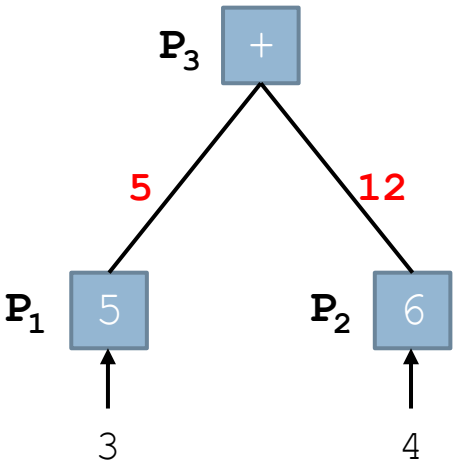
تحلیل

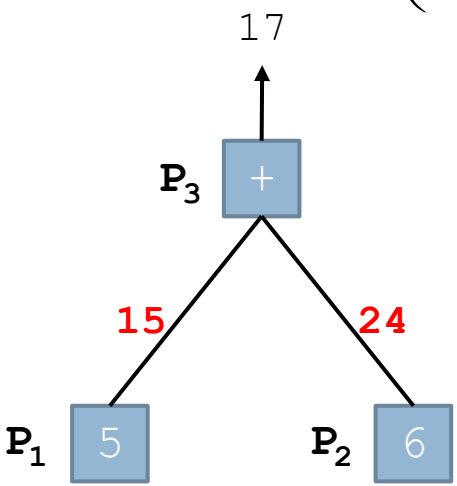
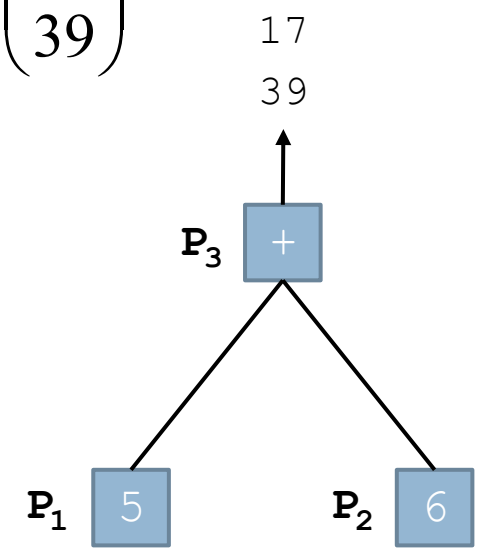
- تحلیل:
  - اولین خروجی پس از $\log n$ مرحله و $m - 1$ خروجی دیگر هر کدام پس از یک مرحله تولید می شوند، بنابراین
    - $t(n) = O(m + \log n - 1)$
  - اگر فرض کنیم $m \leq n$، آنگاه:
    - $t(n) = O(n)$
    - $p(n) = 2n - 1 = O(n)$
    - $c(n) = O(n^2)$

مثال

$$\begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix} \times \begin{pmatrix} 5 \\ 6 \end{pmatrix} = \begin{pmatrix} 17 \\ 39 \end{pmatrix}$$





مثال

$$\begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix} \times \begin{pmatrix} 5 \\ 6 \end{pmatrix} = \begin{pmatrix} 17 \\ 39 \end{pmatrix}$$