فصل ۵
جستجوی موازی

سید ناصر رضوی

rezavi@iust.ac.ir

۱۳۹۰

معرفی مسأله جستجو

- **مسأله جستجو**:
  - **ورودی**:
    - دنباله ای از n عدد صحیح و عدد صحیح x

$$\mathbf{S} = [s_1, s_2, \ldots, s_n]$$

  - **خروجی**:
    - اندیس عنصر x در لیست S (در صورت وجود)

$$s_k = x$$

الگوریتم جستجوی ترتیبی

O(n)

```
procedure SEQUENTIAL SEARCH (S, x, k)

Step 1: (1.1) i ← 1
        (1.2) k ← 0

Step 2: while( i ≤ n and k = 0) do
           if (s_i = x) then k ← i end if
           i ← i + 1
        end while
```

فهرست مطالب

- جستجوی موازی بر روی یک آرایه مرتب
  - مدل EREW
  - مدل CREW
  - مدل CRCW

- جستجو بر روی یک آرایه نامرتب
  - جستجو روی مدل SIMD با اتصال درختی
  - جستجو روی مدل SIMD با اتصال توری شکل

جستجو در یک آرایه مرتب: مدل EREW

- فرض: همه عناصر S متمایز هستند.
  - در ادامه با استفاده از مدل CRCW این فرض را حذف خواهیم نمود.

- الگوریتم جستجو بر روی EREW:
  - انتشار مقدار x در بین پردازنده ها: O(log N)
  - تقسیم S به N قسمت مساوی و توزیع قسمتها در بین پردازنده ها
  - اجرای همزمان جستجوی دودویی به وسیله پردازنده ها بر روی قسمتهای مربوطه: O(log(n/N))

- زمان اجرا:
- O(log N) + O(log(n/N)) = O(log n)

جستجو در یک آرایه مرتب: مدل CREW

- فرض: همه عناصر S متمایز هستند.
  - در ادامه با استفاده از مدل CRCW این فرض را حذف خواهیم نمود.

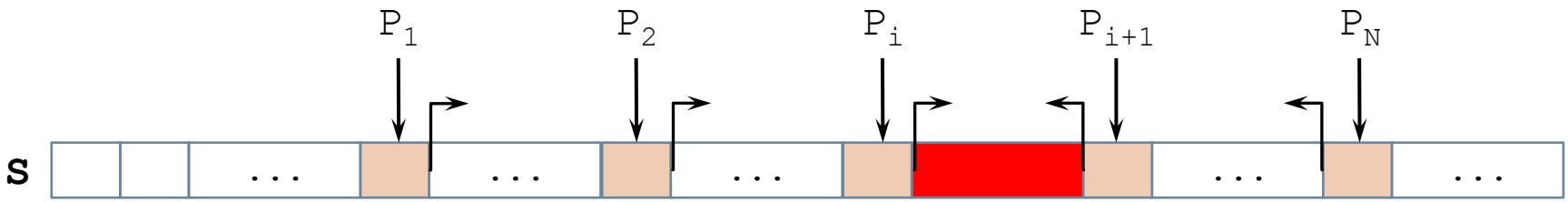
- الگوریتم جستجو بر روی CREW:
  - تقسیم S به N قسمت مساوی و توزیع قسمتها در بین پردازنده ها
  - اجرای همزمان جستجوی دودویی به وسیله پردازنده ها بر روی قسمتهای مربوطه: O(log(n/N))

- زمان اجرا:

- O(log(n/N))

یک الگوریتم بهتر برای مدل CREW

- ایده: تعمیم جستجوی دودویی به جستجوی $(N + 1)$ تایی



جستجوی یک لیست مرتب: مدل CREW


```
procedure CREW SEARCH(S, x, k)

Step 1: {Initialize indices of sequence to be searched}
        (1.1) q ← 1
        (1.2) r ← n

Step 2: {Initialize result and maximum number of stages}
        (2.1) k ← 0
        (2.2) g ← ⌈log(n+1)/log(N+1)⌉

Step 3: while ( q ≤ r and k = 0) do
        (3.1) j0 ← q - 1
        (3.2) for i = 1 to N do in parallel
                  (i) ji ← (q - 1) + i(N + 1)^(g - 1)
```

جستجوی یک لیست مرتب: مدل CREW

```
(ii)  if j_i ≤ r
      then if s[j_i] = x
           then k ← j_i
           else if s[j_i] > x
                then c_i ← left
                else c_i ← right
                end if
           end if
      else (a) j_i ← r + 1
           (b) c_i ← left
      end if
(iii) if c_i ≠ c_{i-1} then (a) q ← j_{i-1} + 1
                            (b) r ← j_i - 1
(iv)  if (i = N and c_i ≠ c_{i+1}) then q ← j_i + 1
end for
(3.3) g ← g - 1
end while
```

تحلیل

- زمان اجرا:

$$t(n) = O(\log_{N+1}(n+1))$$

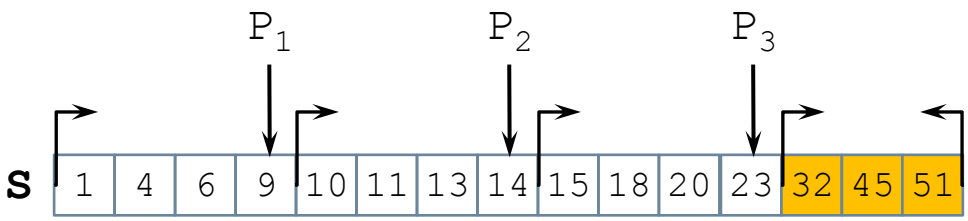
- هزینه:

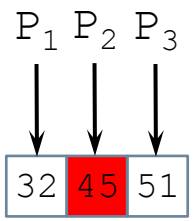
$$c(n) = O(N \log_{N+1}(n+1))$$

- در نتیجه این الگوریتم هزینه-بهینه نیست.

مثال: جستجوی موفق (N = 3)

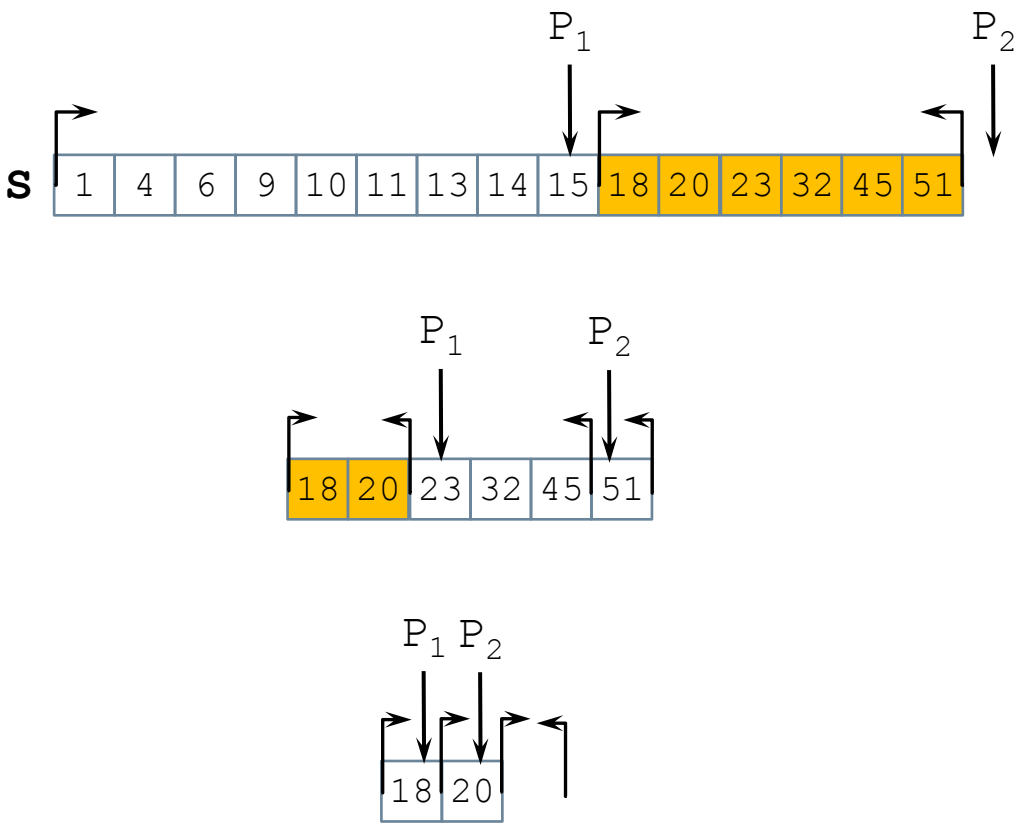
x = 45





مثال: جستجوی ناموفق (N = 2)

x = 21



جستجوی یک لیست مرتب: مدل CRCW

- اگر عناصر S متمایز نباشند:
  - ممکن است چند پردازنده به طور همزمان x را یافته و بخواهند مقدار k را تغییر دهند.
  - در نتیجه باید از مدل CRCW استفاده نماییم.
  - به منظور رفع درگیری مینیمم مقادیر پردازنده ها را در k می نویسیم.
    - هزینه: O(log N)

- تحلیل:
- **t(n)= O(log(n+1)/log(N+1)) + O(log N)**
  - به ازای n = N داریم: t(n) = O(log N) (مانند جستجوی دودویی)

جستجو در یک دنباله نامرتب: الگوریتم عمومی


```
procedure SM SEARCH (S, x, k)

Step 1: for i = 1 to N do in parallel
            read x
        end for

Step 2: for i = 1 to N do in parallel
            (2.1) S_i ← {s_(i-1)(n/N)+1, s_(i-1)(n/N)+2, …, s_i(n/N)}
            (2.2) SEQUENTIAL SEARCH(S_i, x, k_i)
        end for

Step 3: for i = 1 to N do in parallel
            if k_i > 0 then k ← k_i end if
        end for
```

پیاده سازی روی مدل EREW

- مرحله ۱) با استفاده از الگوریتم انتشار
- O(log N)
- مرحله ۲) جستجوی ترتیبی
- O(n/N)
- مرحله ۳) با استفاده از رویه STORE و یک مکانیزم مناسب برای رفع تصادم میان نوشتن های همزمان
- O(log N)

- تحلیل:
- t(n) = O(log N) + O(n/N)
- c(n)= O(N log N) + O(n)
  - این الگوریتم هزینه-بهینه نیست.

پیاده سازی روی مدل ERCW

- مرحله ۱ و ۲ مانند مدل EREW
- مرحله ۳ در زمان ثابت

- <u>تحلیل</u>:
  - همانند مدل EREW

پیاده سازی روی مدل CREW

- مرحله ۱ در زمان ثابت
- مرحله ۲ و ۳ همانند EREW

- تحلیل:
  - همانند مدل EREW

پیاده سازی روی مدل CRCW

- مرحله ۱ و ۳ در زمان ثابت
- مرحله ۲ همانند EREW

- تحلیل:

- $t(n) = O(n/N)$
- $c(n) = N * O(n/N) = O(n)$

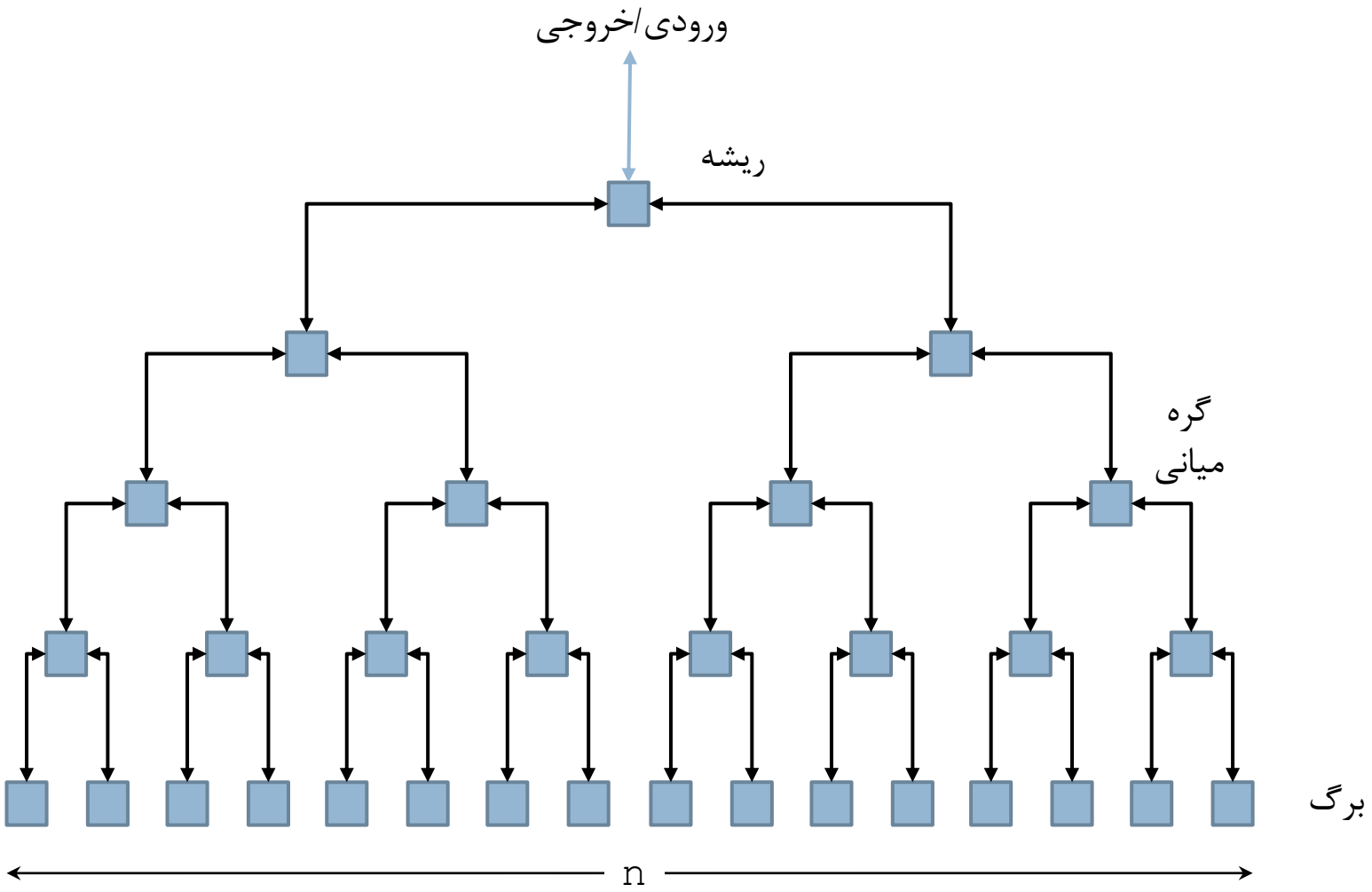
جستجو روی ساختار درختی

- استفاده از یک کامپیوتر SIMD با اتصال درختی شامل n گره برگ برای جستجو در یک فایل حاوی n رکورد

- ریشه درخت:
  - دریافت ورودی و ارسال آن به فرزندان
  - تولید خروجی بر اساس مقادیر دریافتی از فرزندان

- نقش گره های میانی همانند گره ریشه است.

جستجو روی ساختار درختی

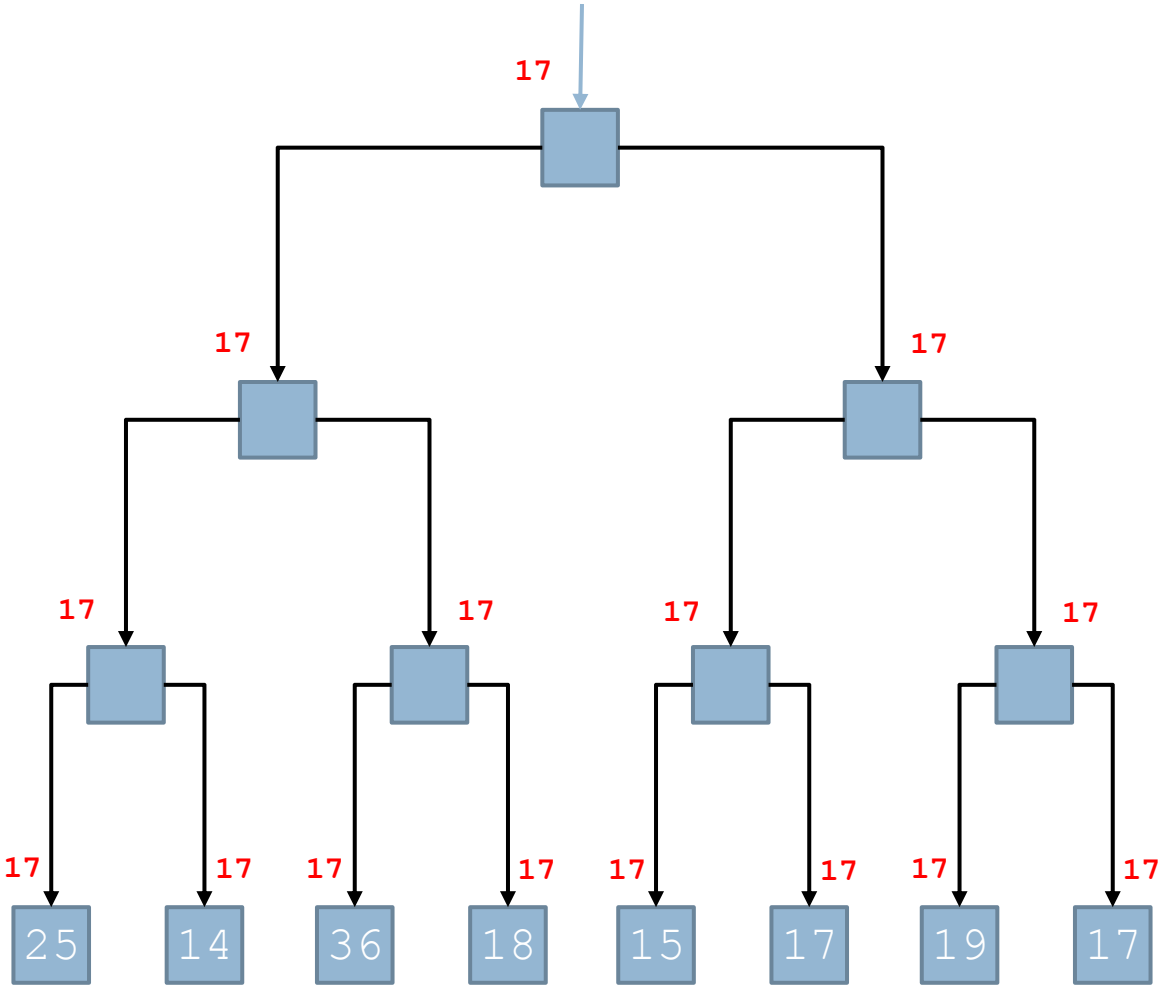


جستجو روی ساختار درختی

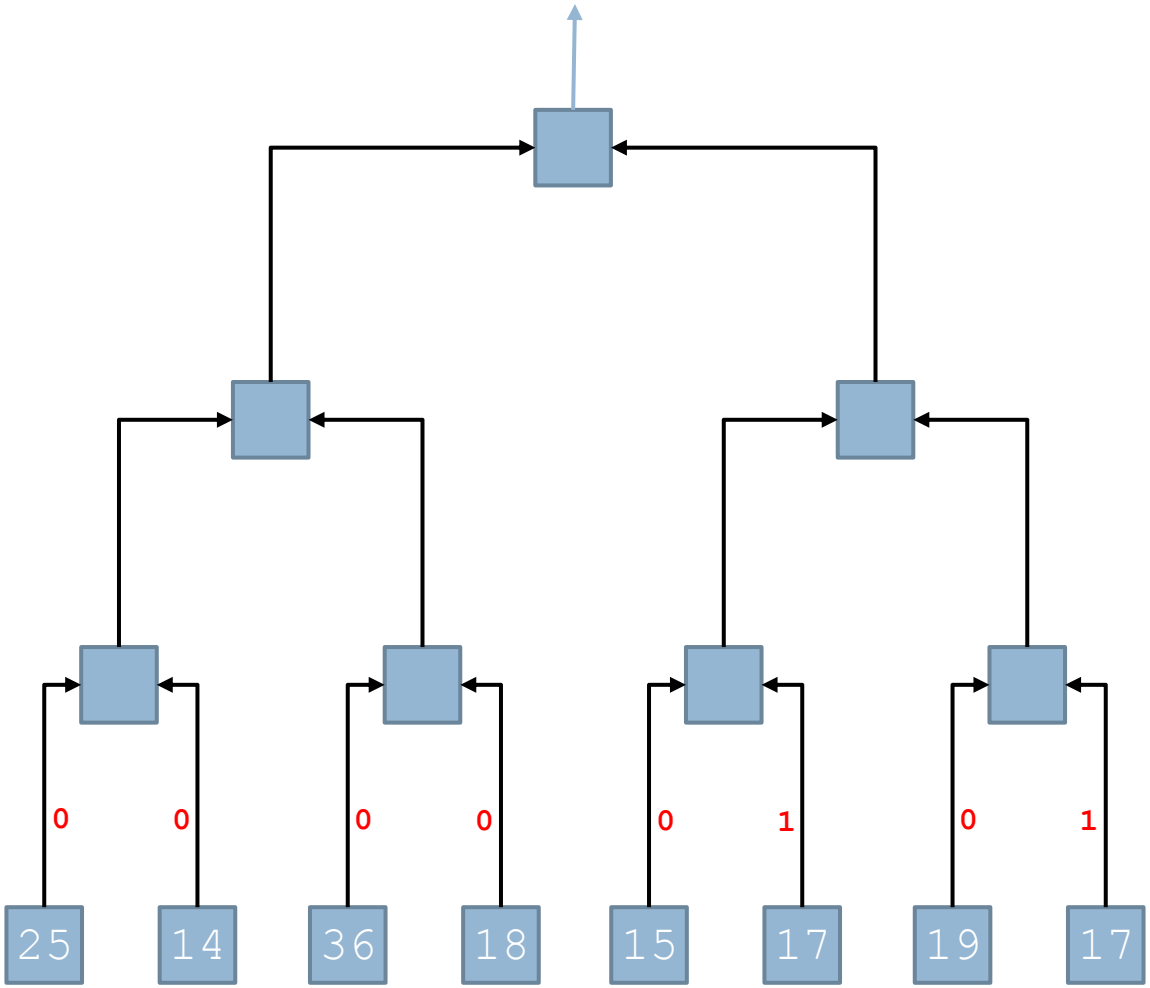
- مراحل جستجو:
  - مرحله ۱) خواندن x به وسیله گره ریشه و انتشار آن به سمت پایین تا رسیدن به برگ های درخت
  - مرحله ۲) مقایسه همزمان مقدار هر برگ با مقدار x.
    - تولید ۱ در صورت مساوی بودن مقادیر
    - تولید ۰ در صورت عدم تساوی
  - مرحله ۳) محاسبه OR گره های برگ در گره والد و تکرار این عمل در سطوح بالاتر درخت تا رسیدن به گره ریشه

- زمان اجرا:
  - $t(n) = O(\log N) + O(1) + O(\log N) = O(\log N)$

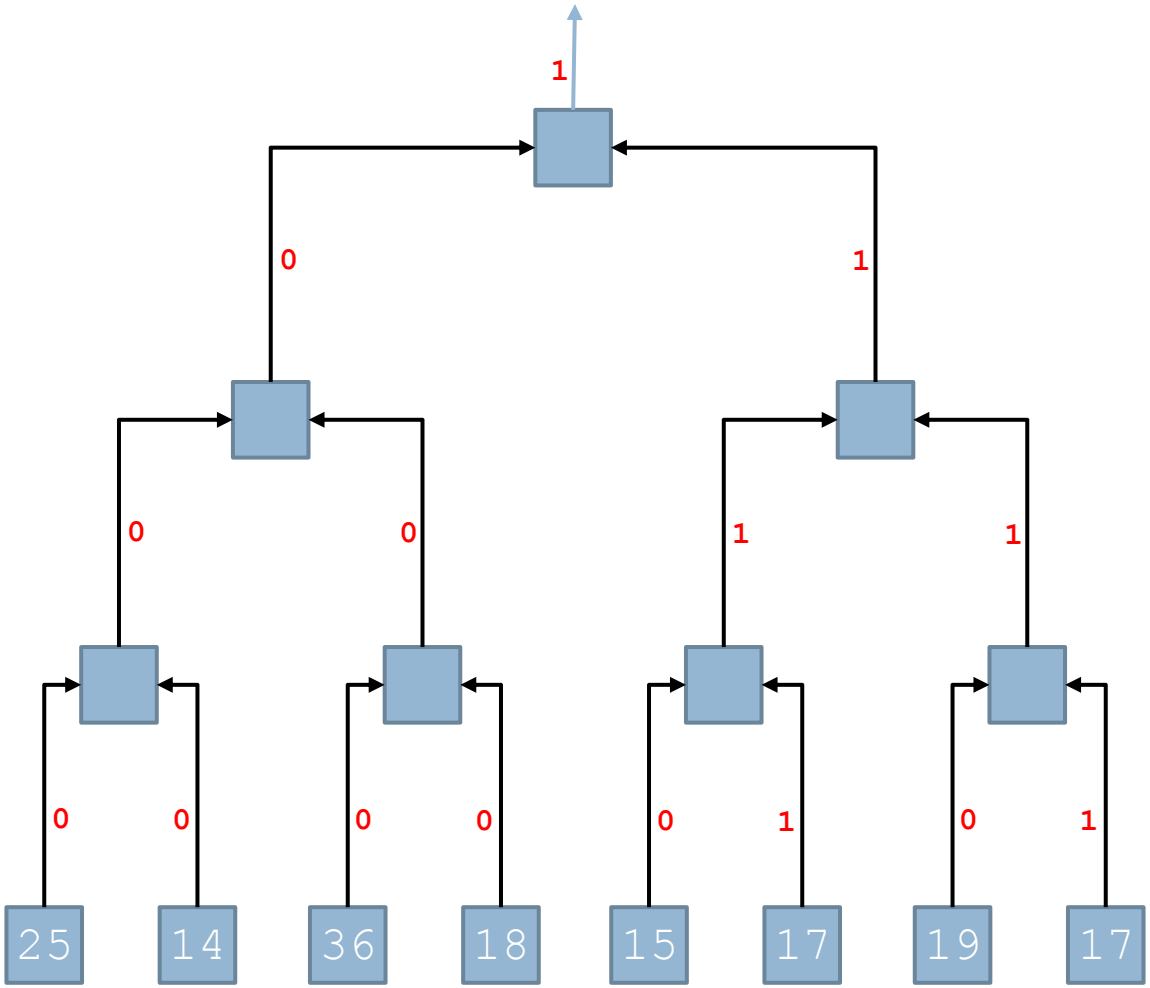
مثال: مرحله ۱



مثال: مرحله ۲



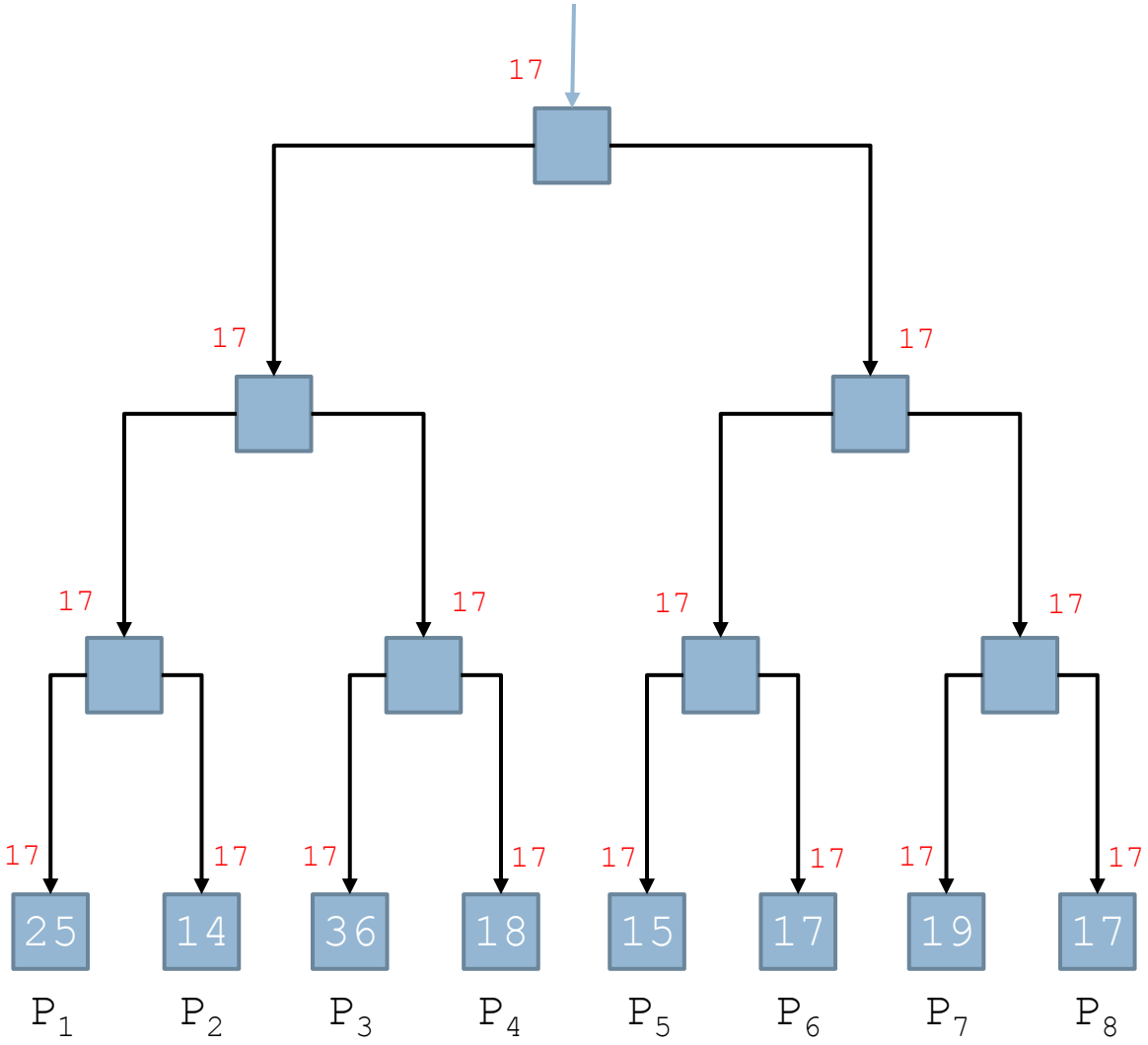
مثال: مرحله ۳



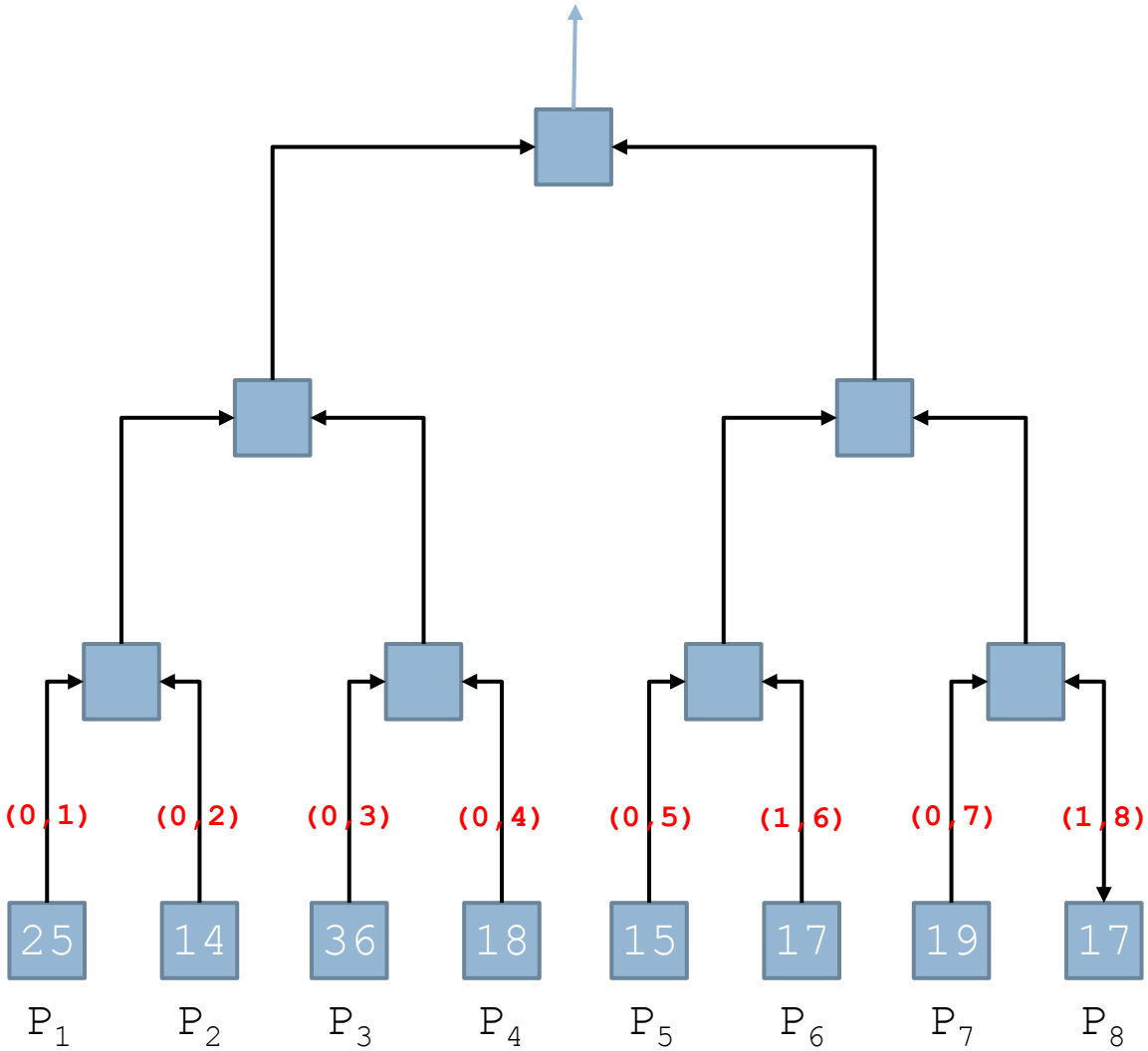
اجرای q پرس و جو

- استفاده از روش pipeline
  - پردازش پرس و جوی اول:
    - O(log n)
  - پردازش پرس و جوی دوم:
    - O(log n) + 1
  - ...
  - پردازش پرس و جوی q اُم:
    - O(log n) + (q - 1)

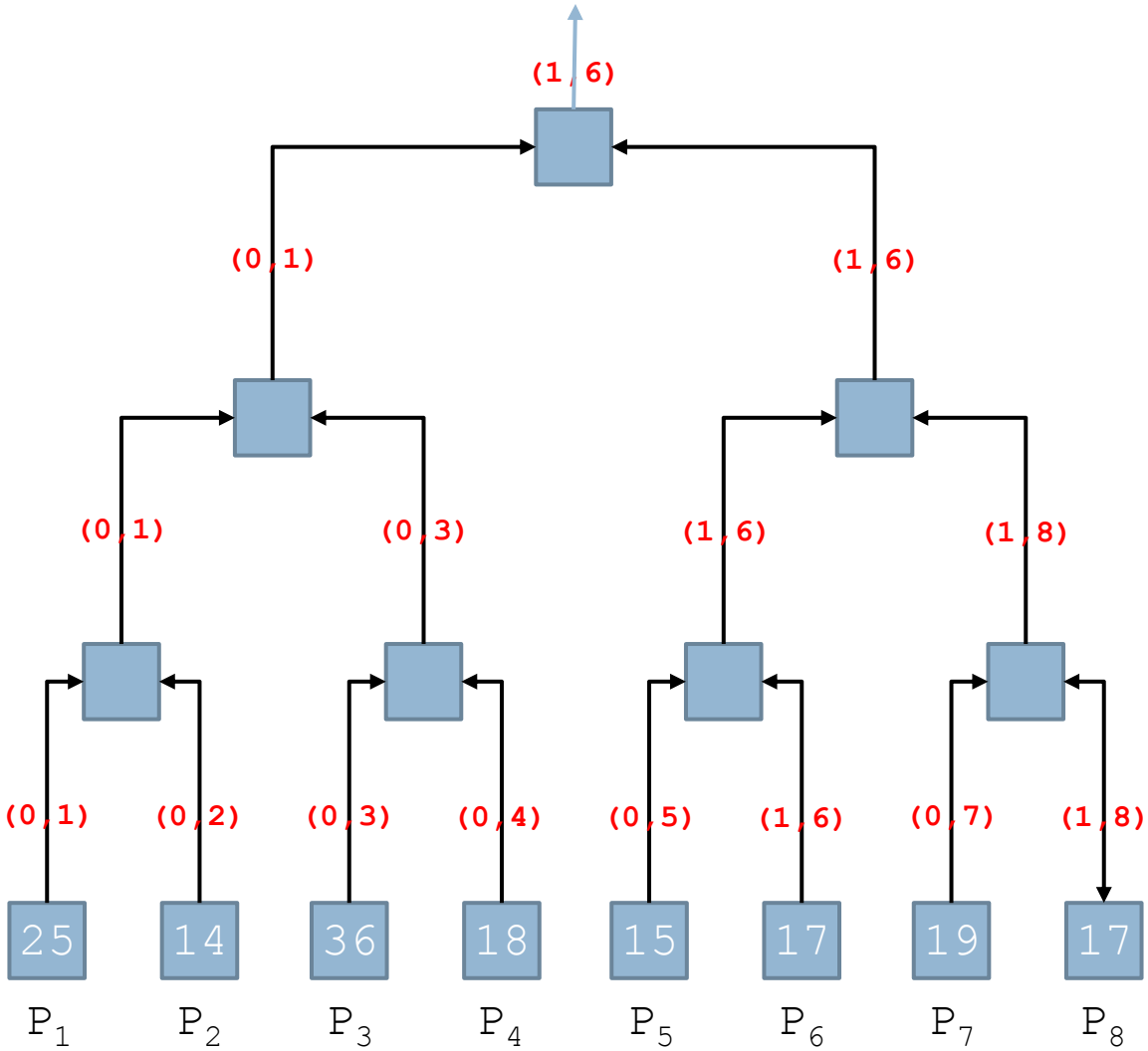
اصلاح اول: برگرداندن اندیس x: مرحله ۱



برگرداندن اندیس x: مرحله ۲



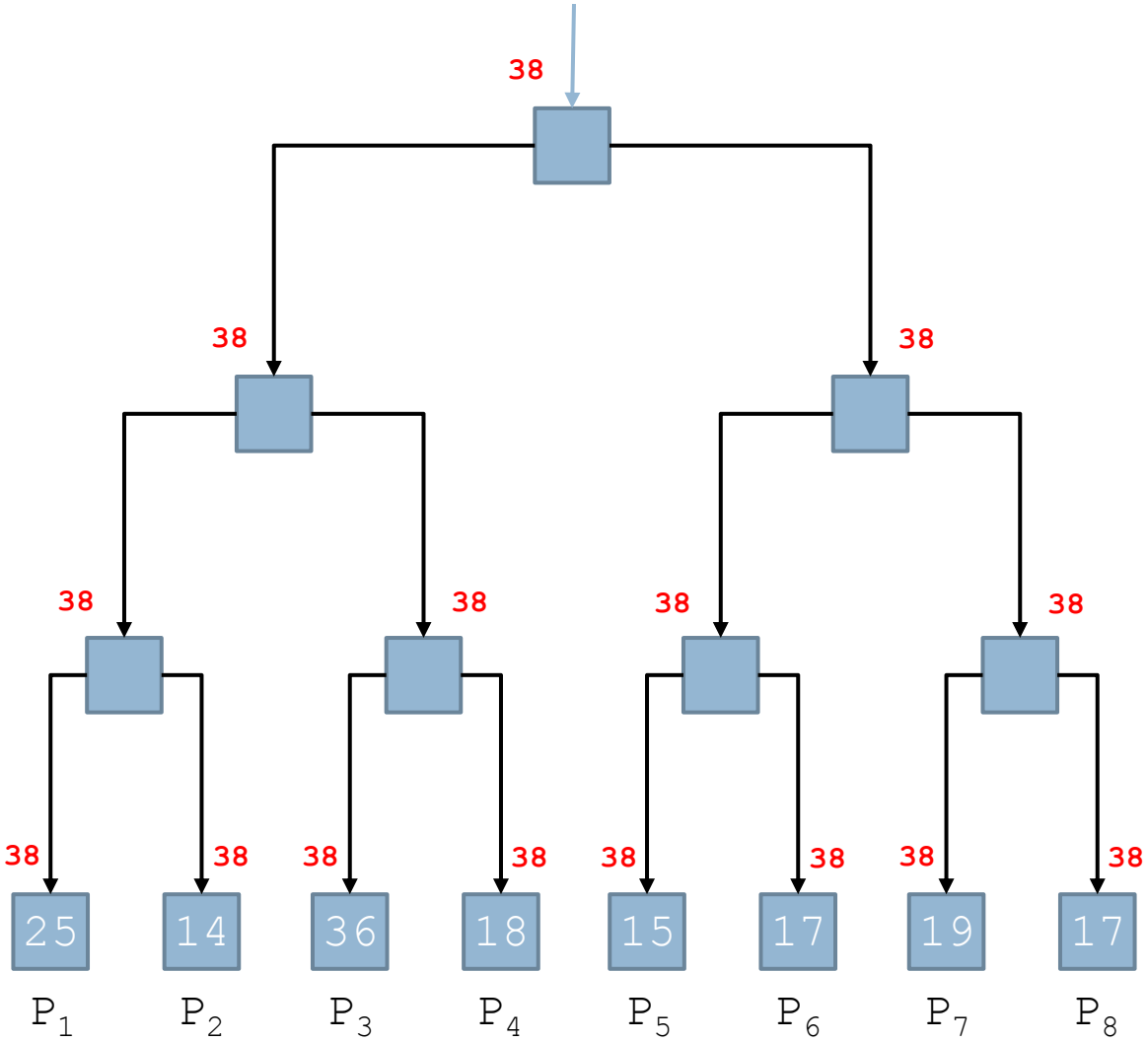
برگرداندن اندیس x: مرحله ۳



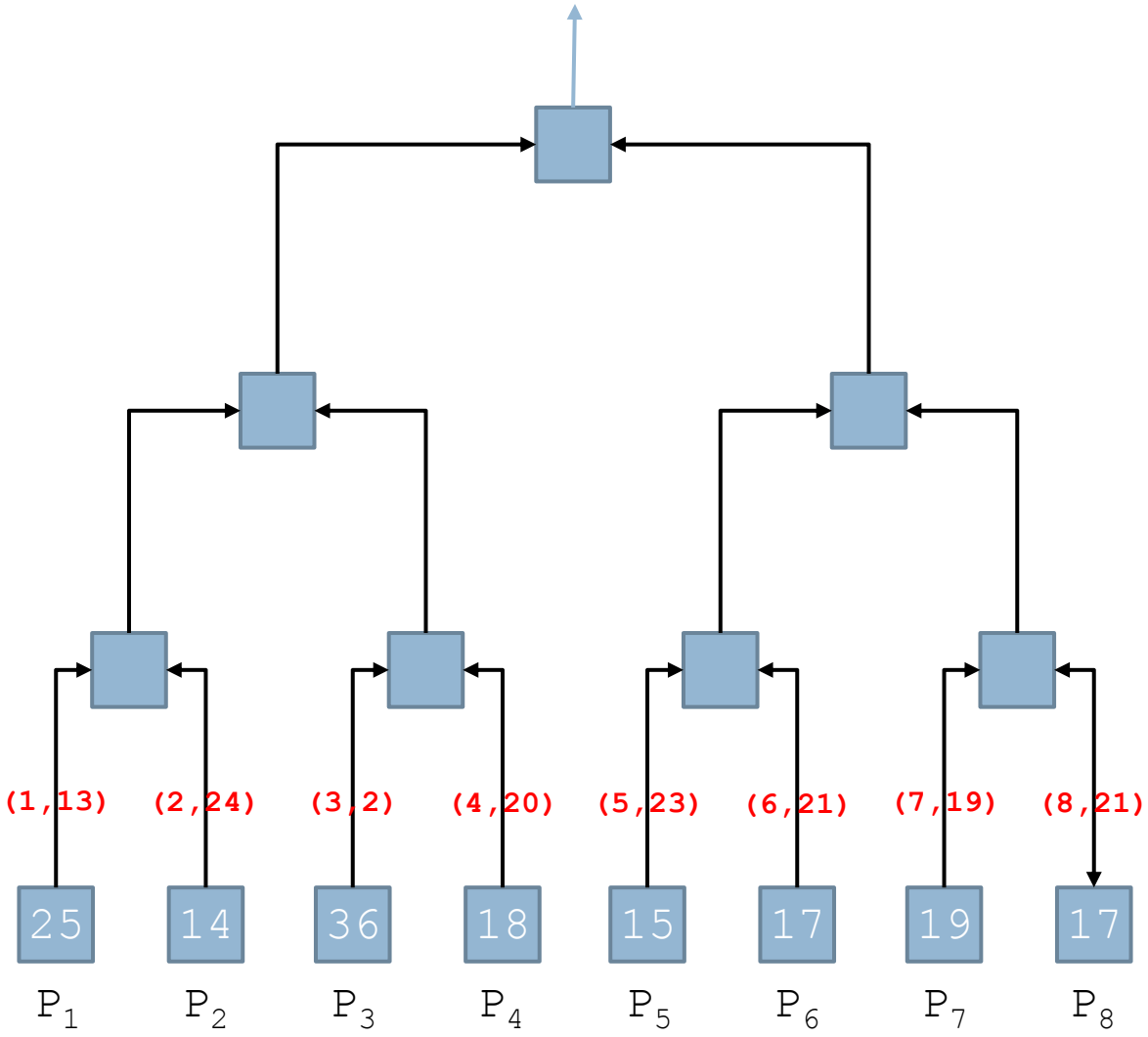
اصلاح دوم: برگرداندن تعداد x ها

- برگردندان تعداد عناصر برابر با x:
  - همانند قبل است با این تفاوت که گره های میانی به جای محاسبه OR فرزندان خود، **مجموع** مقدار آن ها را برمی گردانند.

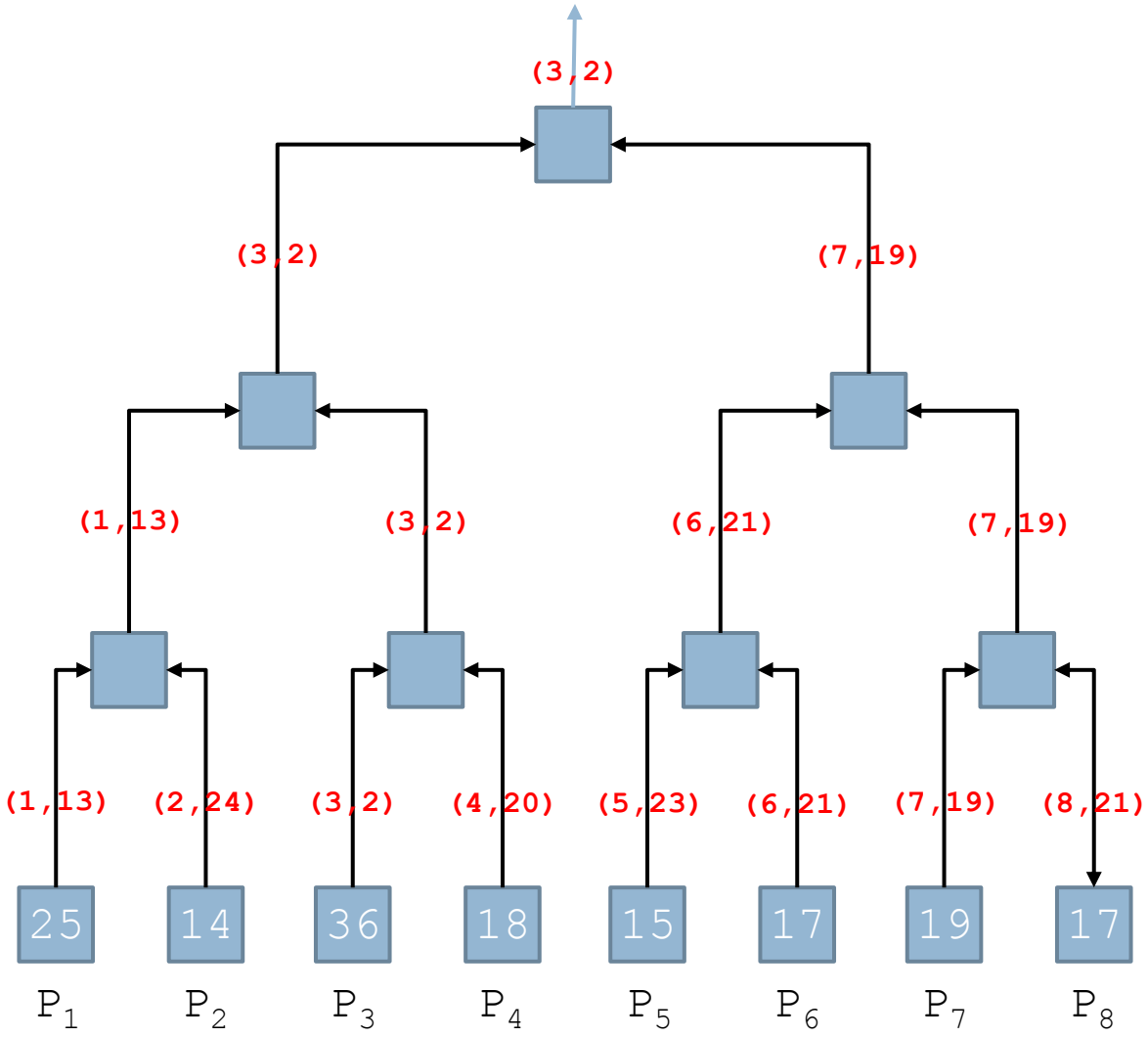
اصلاح سوم: برگرداندن نزدیک ترین مقدار به x
مرحله ۱



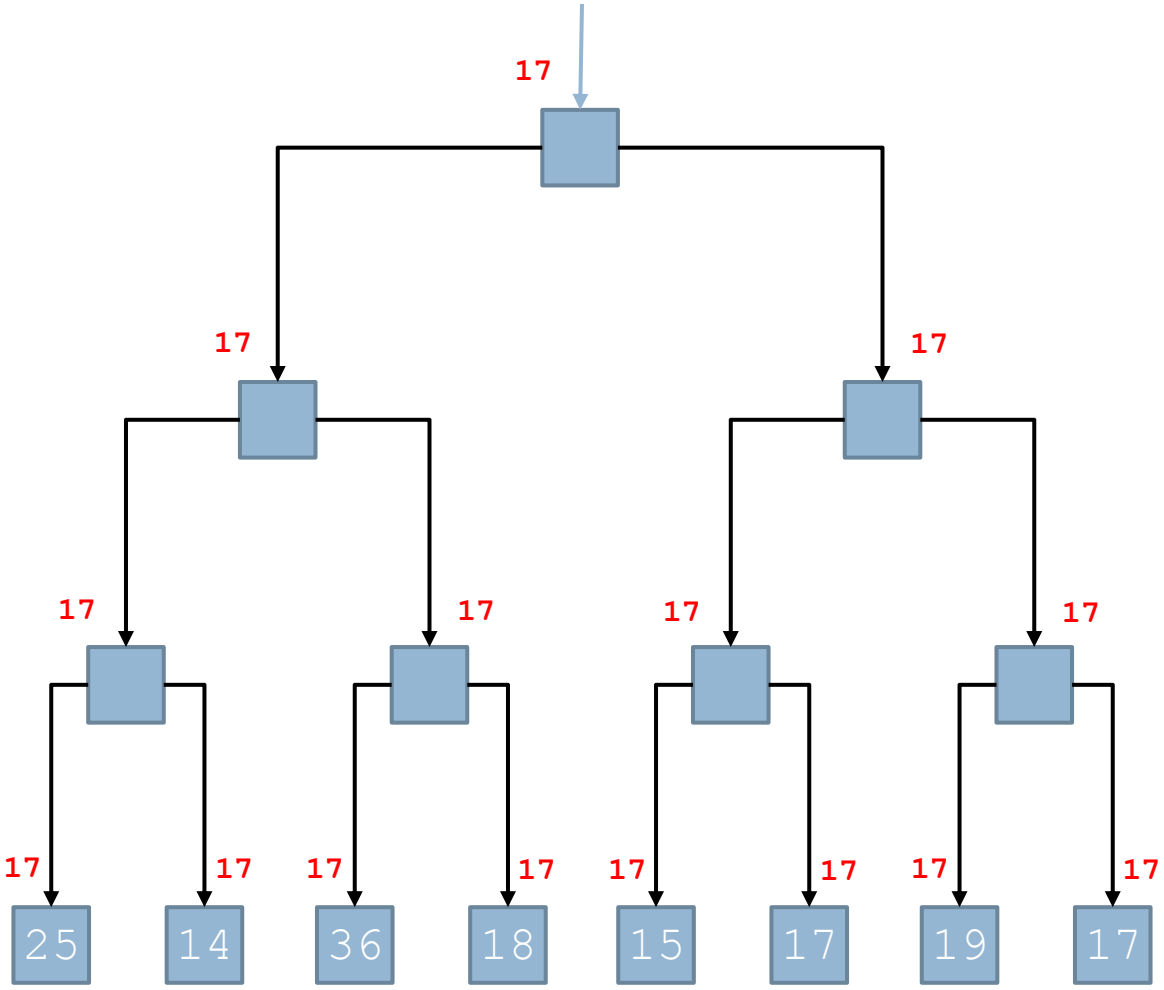
اصلاح سوم: برگرداندن نزدیک ترین مقدار به x
مرحله ۲



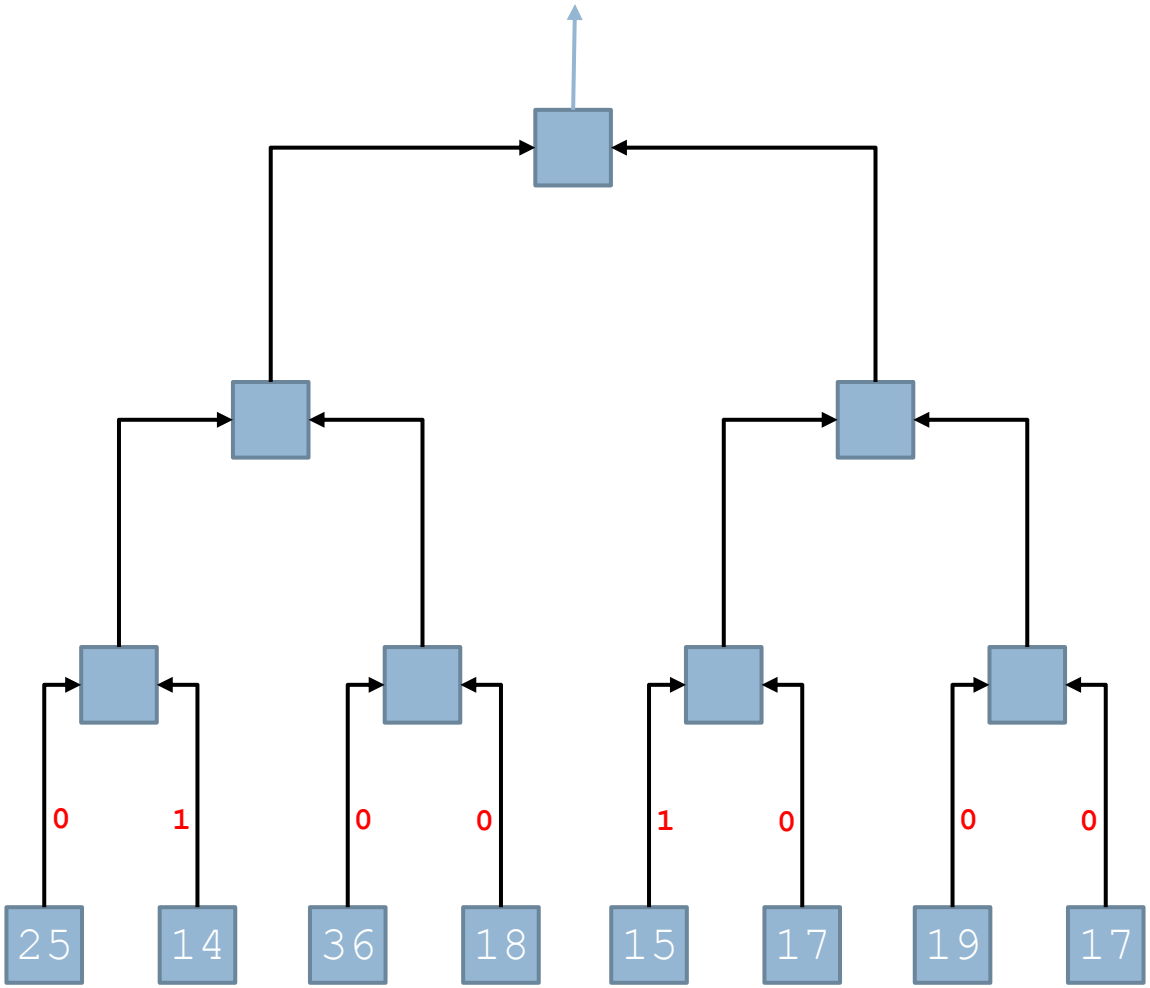
اصلاح سوم: برگرداندن نزدیک ترین مقدار به x
مرحله ۳



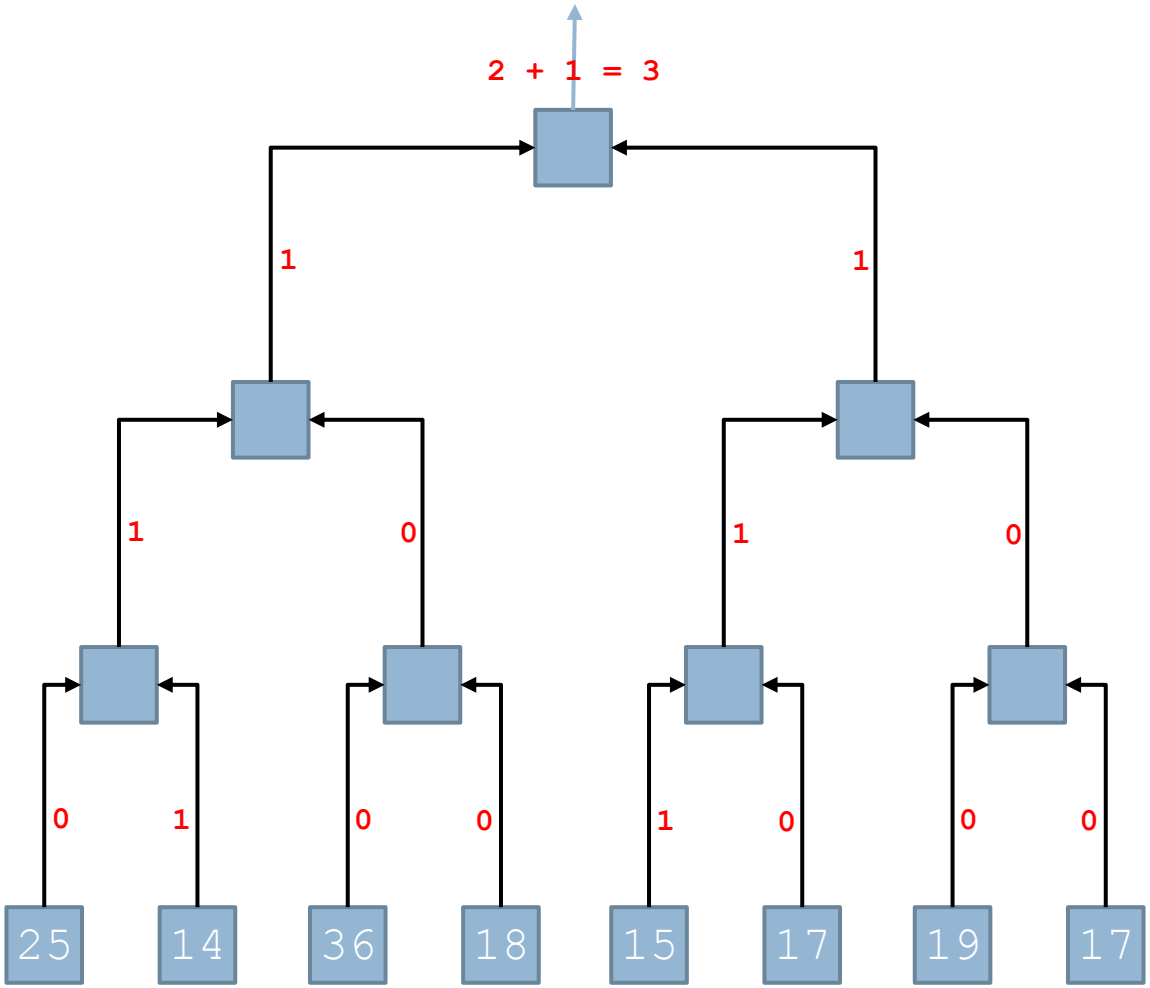
اصلاح چهارم) برگرداندن مرتبه x: مرحله ۱



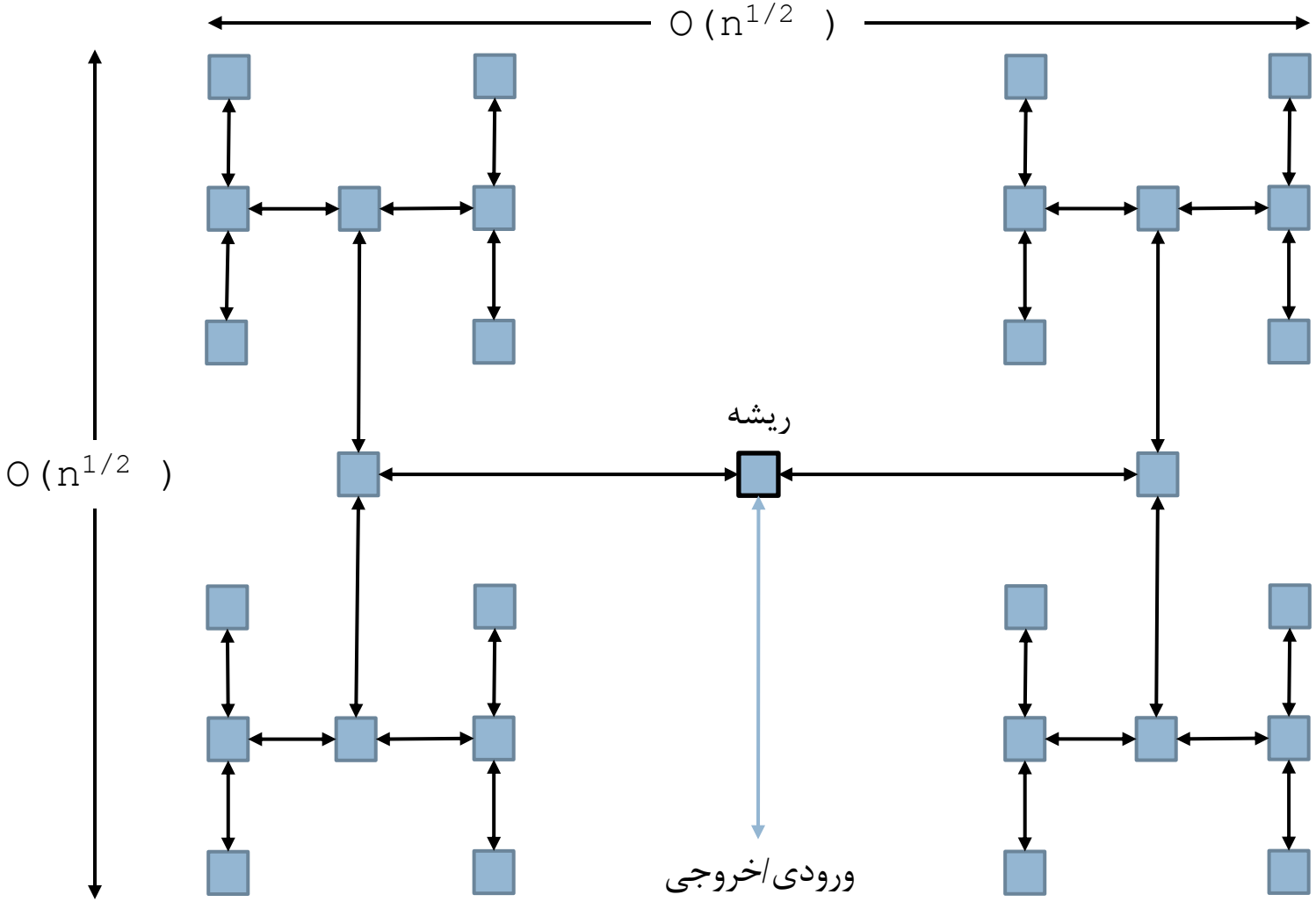
اصلاح چهارم) برگرداندن مرتبه x: مرحله ۲



اصلاح چهارم) برگرداندن مرتبه x: مرحله ۳

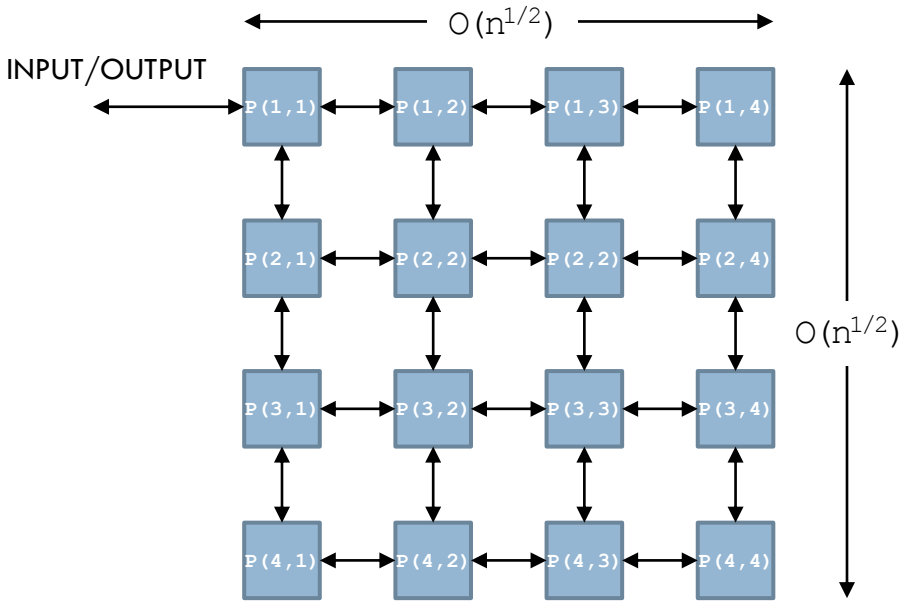


جستجو روی ساختار درختی: پیاده سازی سخت افزاری



جستجو روی ساختار توری شکل

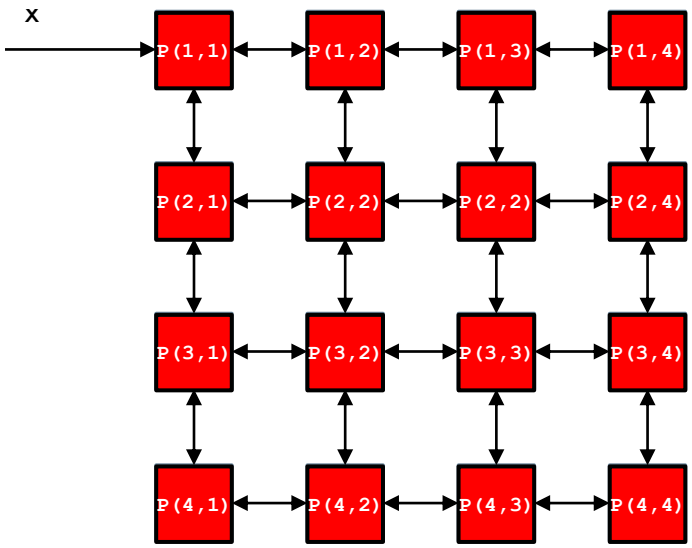
- ساختار توری شکل:
  - یک آرایه دو بعدی از پردازنده ها



- ویژگی ها:
  - طول سیم ها مستقل از اندازه ورودی و ثابت است.
  - مساحت اشغال شده به وسیله پردازنده ها برابر با $O(n)$ است.
  - زمان اجرا به ازای هر عمل پرس و جو یا نگهداری برابر با $O(n^{1/2})$ است.

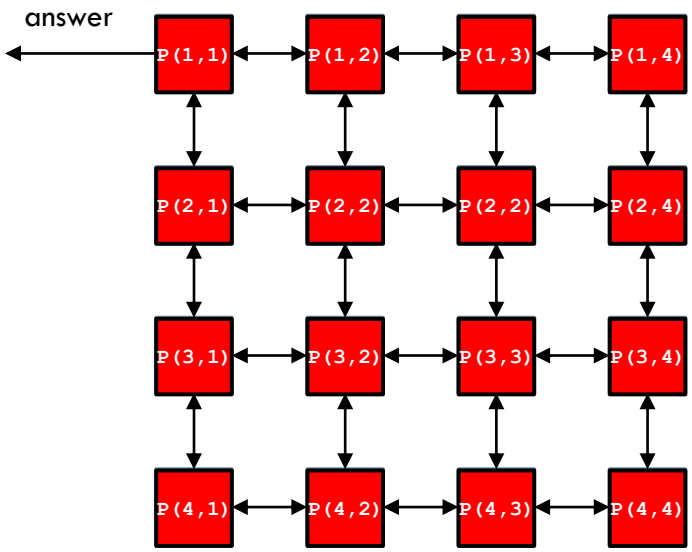
عمل پرس و جو

- مراحل:
  - باز کردن : انتشار x از چپ به راست و از بالا به پایین



عمل پرس و جو

- مراحل:
  - تا زدن: انتشار بیت خروجی از راست به چپ و از پایین به بالا



الگوریتم جستجو روی ساختار توری شکل


```
procedure MESH-SEARCH(S, x, answer)
Step 1: { P(1,1) READS THE INPUT }
 O(1)   if x = s_{1,1} then b_{1,1} ← 1
                       else b_{1,1} ← 0

Step 2: { UNFOLDING }
 O(n^{1/2})  for i = 1 to n^{1/2} - 1 do
        (2.1) for j = 1 to i do in parallel
              (i) P(j, i) transmits (b_{j,i}, x) to P(j, i + 1)
             (ii) if (b_{j,i} = 1 or x = s_{j,i+1}) then b_{j,i+1} ← 1
                                                    else b_{j,i+1} ← 0
                  end if
              end for
```

الگوریتم جستجو روی ساختار توری شکل


```
(2.2) for j = 1 to i + 1 do in parallel
        (i)  P(i, j) transmits (b_{i,j}, x) to P(i + 1, j)
       (ii)  if (b_{i,j} = 1 or x = s_{i+1,j}) then b_{i+1,j} ← 1
                                                 else b_{i+1,j} ← 0
             end if
      end for
end for.

Step 3: { FOLDING }
O(n^{1/2})  for i = n^{1/2} downto 2 do
(3.1) for j = 1 to i do in parallel
            P(j, i) transmits b_{j,i} to P(j, i - 1)
      end for
```

الگوریتم جستجو روی ساختار توری شکل


```
(3.2)  for j = 1 to i - 1 do in parallel
           b_{j,i-1} ← b_{j,i}
       end for
(3.3)  if (b_{i,i-1} = 1 or b_{i,i} = 1) then b_{i,i-1} ← 1
                                         else b_{i,i-1} ← 0
       end if
(3.4)  for j = 1 to i - 1 do in parallel
           P(i, j) transmits b_{i,j} to P(i - 1, j)
       end for
(3.5)  for j = 1 to i - 2 do in parallel
           b_{i-1, j} ← b_{i,j}
       end for
(3.6)  if (b_{i-1,i-1} = 1 or b_{i,i-1} = 1) then b_{i-1,i-1} ← 1
                                             else b_{i-1,i-1} ← 0
       end if
end for.
```

الگوریتم جستجو روی ساختار توری شکل


```
Step 4: { P(1,1) PRODUCES THE OUTPUT }
O(1)    if b_{1,1} = 1 then answer ← yes
                       else answer ← no
        end if
```

تحلیل

- زمان اجرا: $O(n^{1/2})$
  - مرحله ۱ و مرحله ۴: $O(1)$
  - مرحله ۲ و مرحله ۳: $O(n^{1/2})$

مثال: باز کردن

1	2	3	4
8	7	6	5
9	10	11	12
16	15	14	13

15			

15	15		

15	15		
15	15		

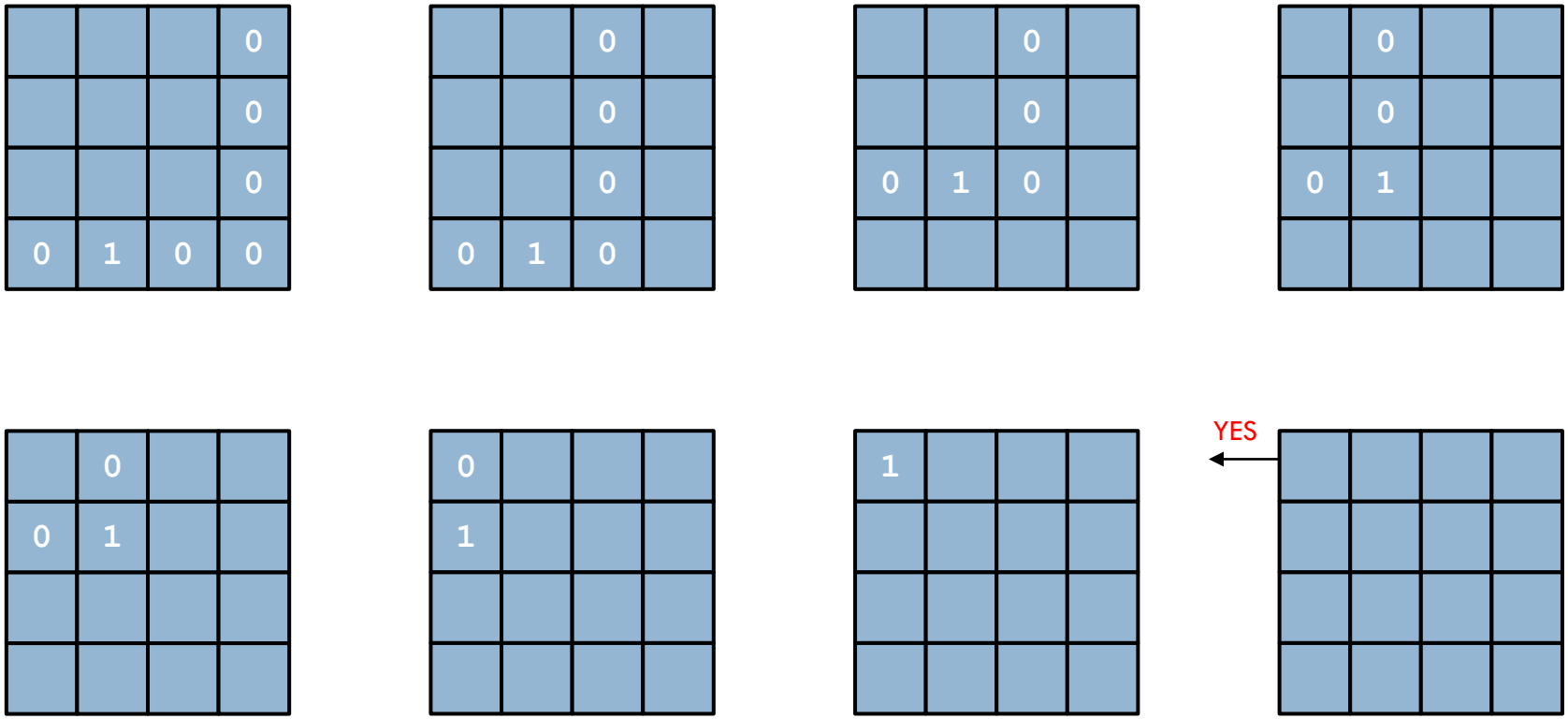
15	15	15	
15	15	15	

15	15	15	
15	15	15	
15	15	15	

15	15	15	15
15	15	15	15
15	15	15	15

15	15	15	15
15	15	15	15
15	15	15	15
15	15	15	15

مثال: باز کردن


0
0
0 1
0
0
0
0 1 0
0
0
0
0 1 0
0
0
0
0 1 0
0
0
0
0 1
0
0
0
0 1
0
0
0
0 1

عملیات نگهداری

- عمل درج:
  - هر پردازنده در بالاترین سطر شبکه، تعداد پردازنده های اشغال نشده در ستون خود را نگهداری می کند.
  - در هنگام درج یک عنصر جدید، این عنصر در طول سطر اول منتشر می شود تا به اولین ستون شامل حداقل یک پردازشگر اشغال نشده برسد.
  - سپس عنصر جدید در طول این ستون منتشر شده تا به اولین پردازنده اشغال نشده برسد.
  - تعداد پردازنده های اشغال نشده در ستون مزبور یک واحد کاهش می یابد.

- عمل حذف:
  - ابتدا عنصر مورد نظر مکان یابی می شود.
  - سپس وضعیت پردازنده حاوی این عنصر به «اشغال نشده» تغییر می کند.
  - تعداد پردازنده های اشغال نشده در این ستون یک واحد افزایش می یابد.

- عمل به روز رسانی:
  - مکان یابی و سپس انجام تغییرات