Divide & Conquer

สมชาย ประสิทธิ์จูตระกูล
ภาควิชาวิศวกรรมคอมพิวเตอร์
จุฬาลงกรณ์มหาวิทยาลัย

(มีนาคม ๒๕๕๐)

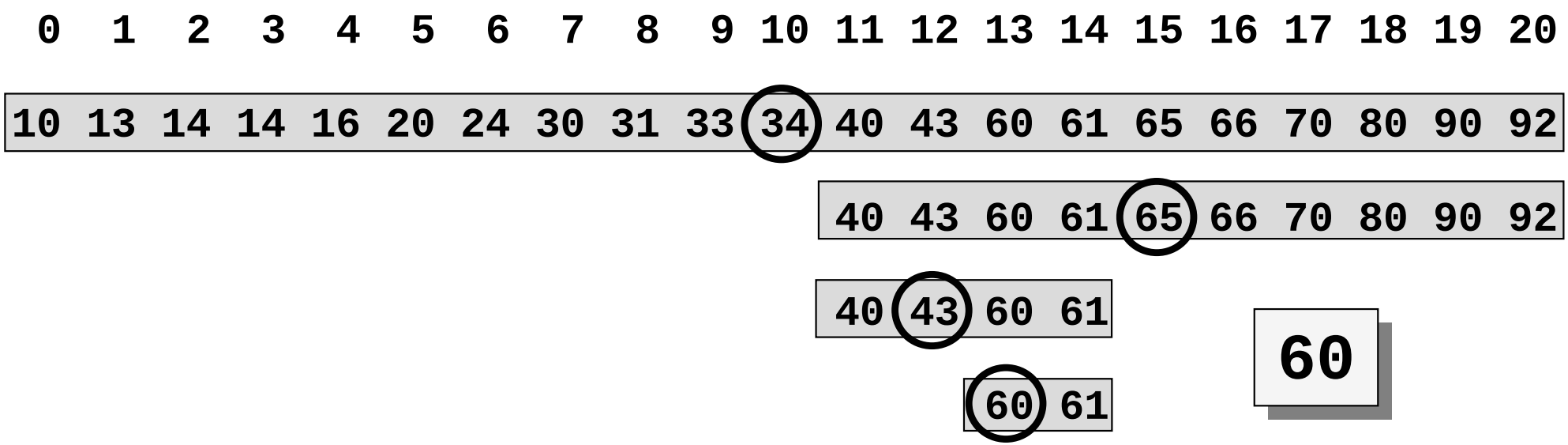
หัวข้อ : Divide & Conquer

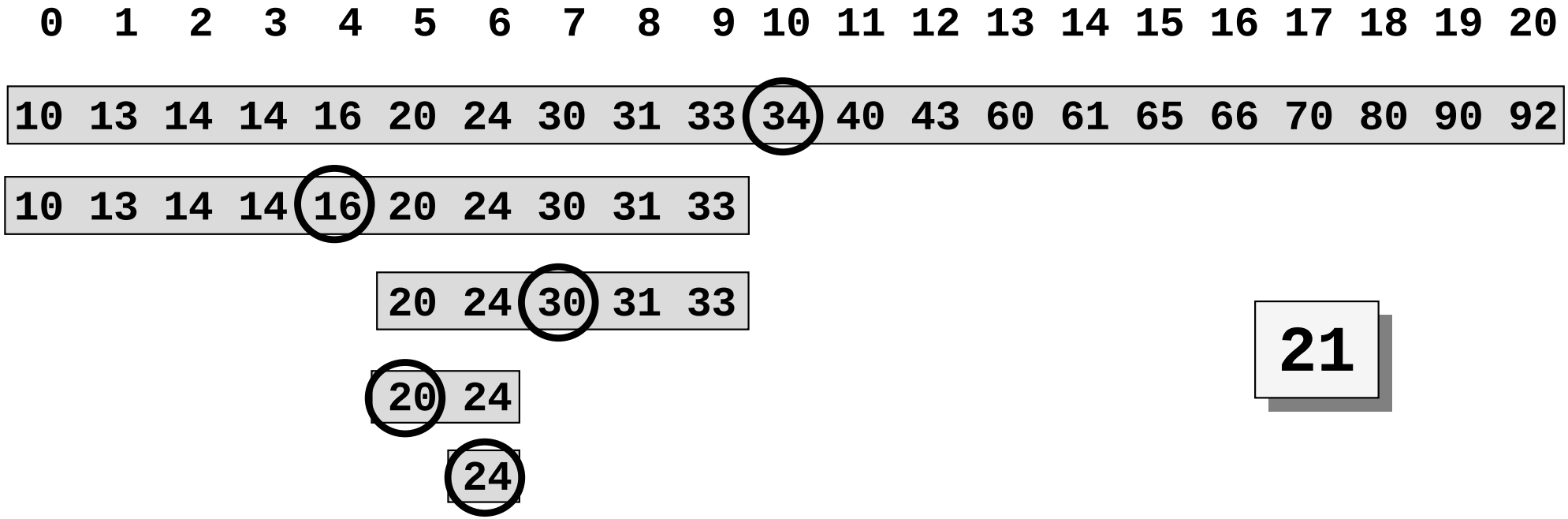
- Mergesort
- Quicksort
- Selection
- Binary search
- Celebrity

Divide and Conquer

```
DQ( P ) {
  if ( P is trivial ) return Solve( P )
  Divide P into P1, P2, …, Pk
  for ( i = 1 to k )
    Si = DQ( Pi )
  S = Combine( S1, S2, …, Sk )
  return S
}
```

Binary Search





Binary Search

```
int binarySearch(int[] a, int key) {
  int low = 0;
  int high = a.length - 1;
  while (low <= high) {

    int mid = (low + high) / 2;

    int midVal = a[mid];
    if (midVal < key)
      low = mid + 1;
    else if (midVal > key)
      high = mid - 1;
    else
      return mid;     // key found
  }
  return -(low + 1);  // key not found.
}
```

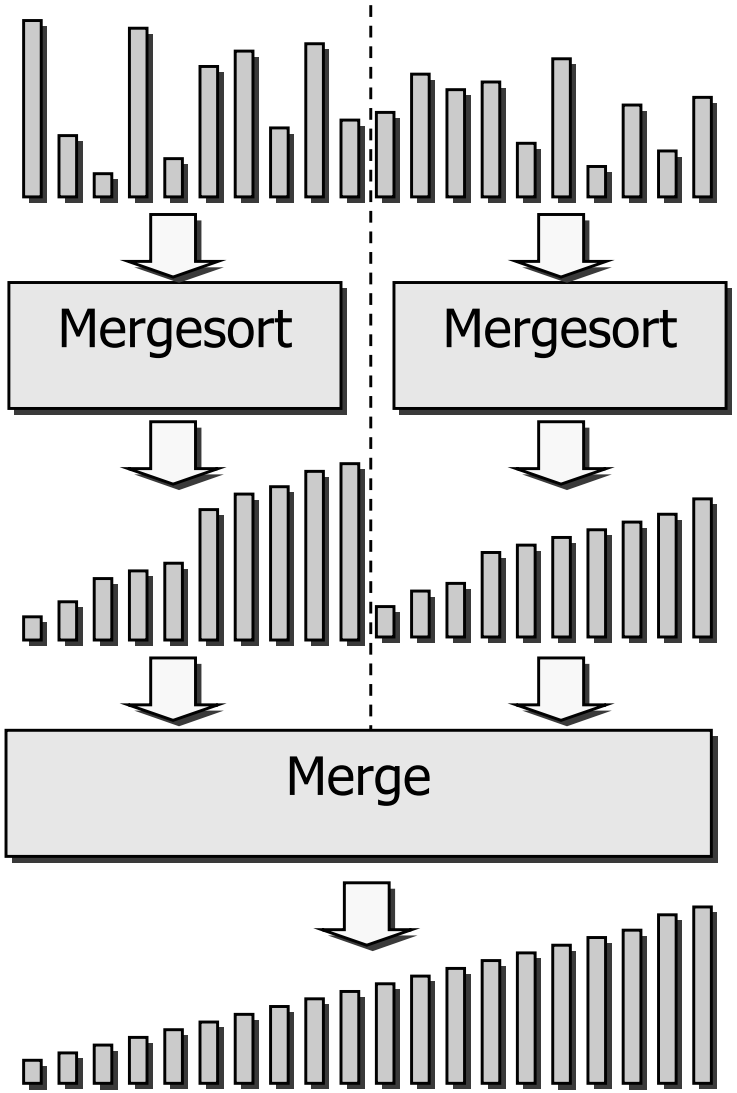
jdk 1.5

Binary Search ที่เขียนกันมักเขียนผิด

- mid = (low + high) / 2; // ผิด !!
- mid = (low + high) >> 1 // ก็ผิด
- int บวกทีละหนึ่งไปเรื่อย ๆ สักวันจะได้ค่าติดลบ !!
- ถ้า low + high มีค่ามาก อาจได้ค่าติดลบ !!
  - เช่น low = high = 2^{30}, low + high = –2147483648
- วิธีแก้
  - mid = low + ((high - low) / 2);
  - mid = (low + high) >>> 1; // Java
  - mid = ((unsigned) (low + high)) >> 1; // C

เหตุการณ์เช่นนี้จะเกิดขึ้นเมื่อเราใช้อาเรย์ขนาดเป็นพันล้าน ซึ่งไม่ค่อยเกิดในโลกของ 32-bit CPU แต่อนาคตเป็นเรื่องไม่แน่.

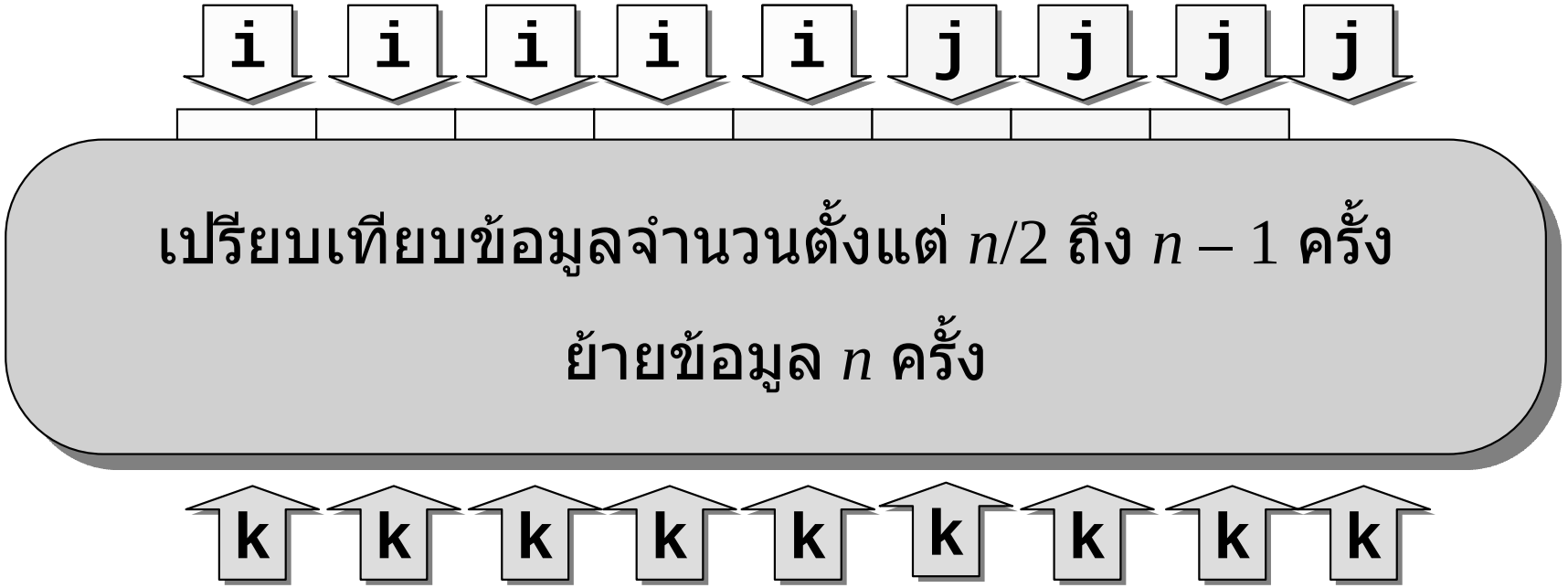
การเรียงลำดับแบบผสาน (Merge Sort)



การเรียงลำดับแบบผสาน : โปรแกรม

```
void mergeSort(int[] d) {
  mSortR(d, 0, d.length-1, (int[]) d.clone());
}

void mSortR(int[] d, int left, int right, int[] t) {
  if (left < right) {
    int m = left + ((right - left) / 2);
    mSortR(t, left, m , d);
    mSortR(t, m + 1, right , d);
    merge(t, left, m, right , d);
    for (int i=left; i<=right; i++) d[i] = t[i];
  }
}
```

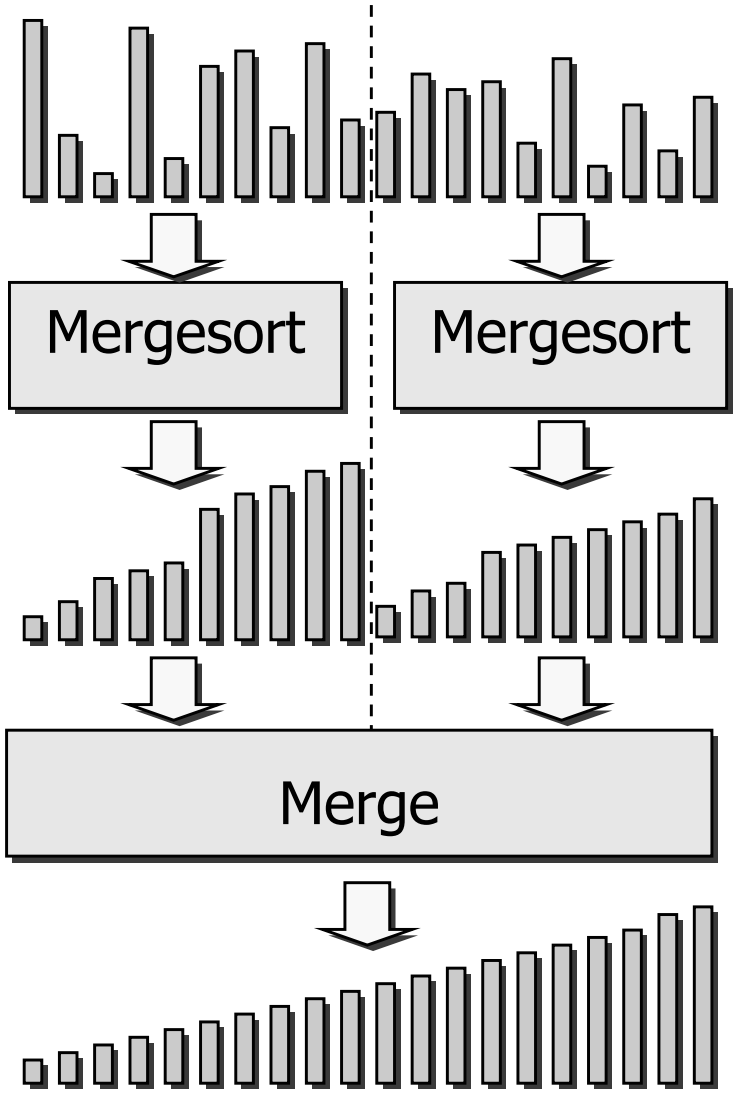
การผสาน (merge)




```
void merge(int[] a, int left, int mid, int right,
           int[] b) {
  int i = left, j = mid+1;
  for (int k = left; k <= right; k++) {
    if (i > mid)   {b[k] = a[j++]; continue;}
    if (j > right) {b[k] = a[i++]; continue;}
    b[k] = (a[i] < a[j]) ? a[i++] : a[j++];
  }
}
```

การเรียงลำดับแบบผสาน : วิเคราะห์

ให้ $c(n)$ คือจำนวนครั้งของการเปรียบเทียบเพื่อเรียงลำดับข้อมูล n ตัว



$$c(n/2) + c(n/2)$$

$$c(n) \geq 2c(n/2) + n/2$$

$$c(n) \leq 2c(n/2) + n - 1$$

$n/2$ ถึง $n - 1$

ขอบเขตบนของจำนวนการเปรียบเทียบ

$$
\begin{aligned}
c(n) &\le 2c(n/2) + n - 1 \\
&\le 2(2c(n/4) + n/2 - 1) + n - 1 \\
&= 4c(n/4) + n - 2 + n - 1 \\
&\le 4(2c(n/8) + n/4 - 1) + n - 2 + n - 1 \\
&= 8c(n/8) + n - 4 + n - 2 + n - 1 \\
&\quad \cdots \\
&\le 2^k c(n/2^k) + n - 2^{k-1} + \cdots + n - 2 + n - 1 \\
&= 2^k c(n/2^k) + nk - (2^k - 1) \\
&= n \log_2 n - n + 1 \\
&= O(n \log n)
\end{aligned}
$$

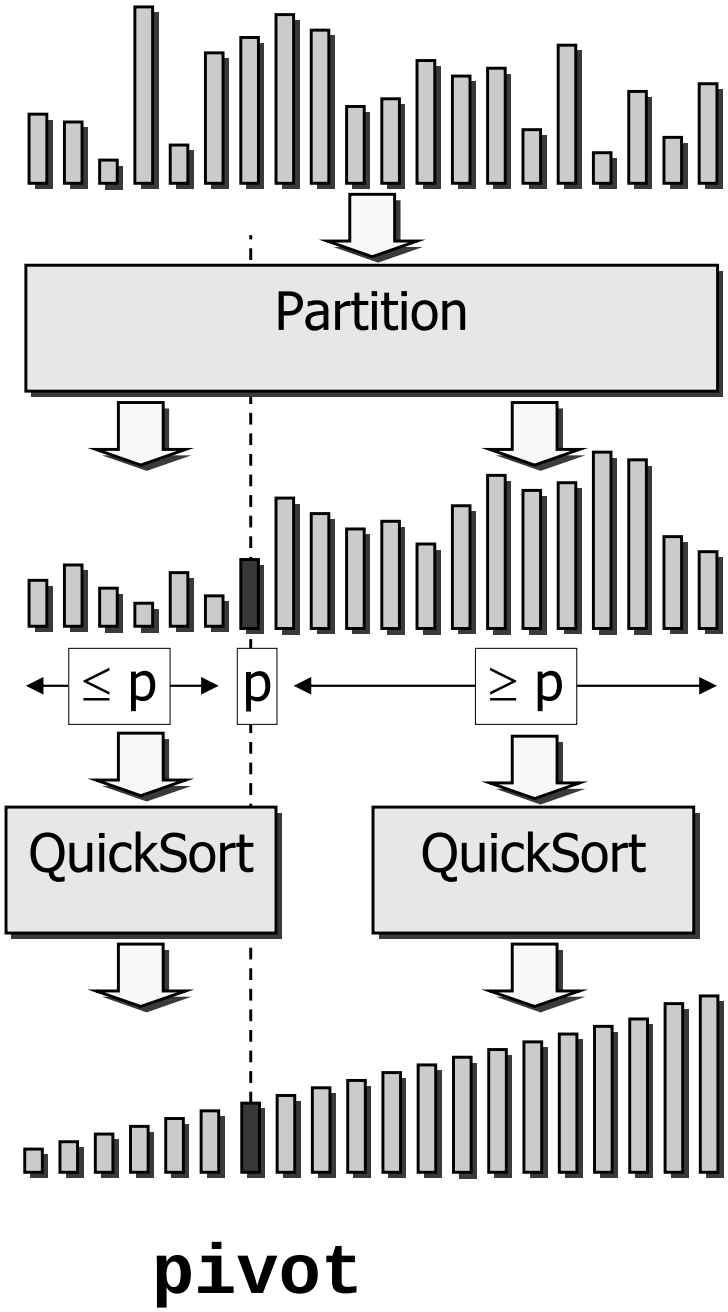
$c(1) = 0$

ให้ $n = 2^k$, $k = \log_2 n$

เวลาการทำงาน

- ขอบเขตบน : $c(n) \leq 2c(n/2) + (n-1) = O(n \log n)$
- ขอบเขตล่าง : $c(n) \geq 2c(n/2) + n/2 = \Omega(n \log n)$
- จำนวนการเปรียบเทียบ = $\Theta(n \log n)$
- จำนวนการย้าย $g(n) = 2g(n/2) + n = \Theta(n \log n)$
- เวลาการทำงาน = $\Theta(n \log n)$

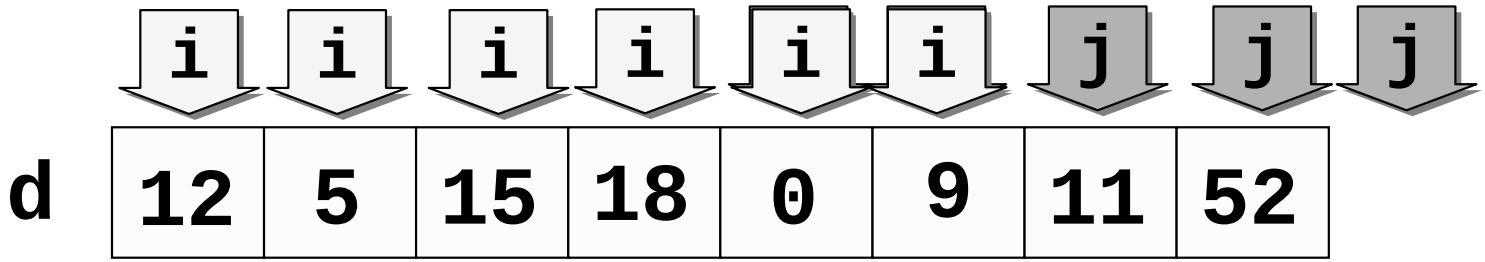
การเรียงลำดับแบบเร็ว (Quick Sort)



การเรียงลำดับแบบเร็ว : โปรแกรม

```
void quickSort(int[] d) {
  qSortR(d, 0, d.length-1);
}
void qSortR(int[] d, int left, int right){
  if (left < right) {
    int j = partition(d, left, right);
    qSortR(d, left, j - 1);
    qSortR(d, j + 1, right);
  }
}
```

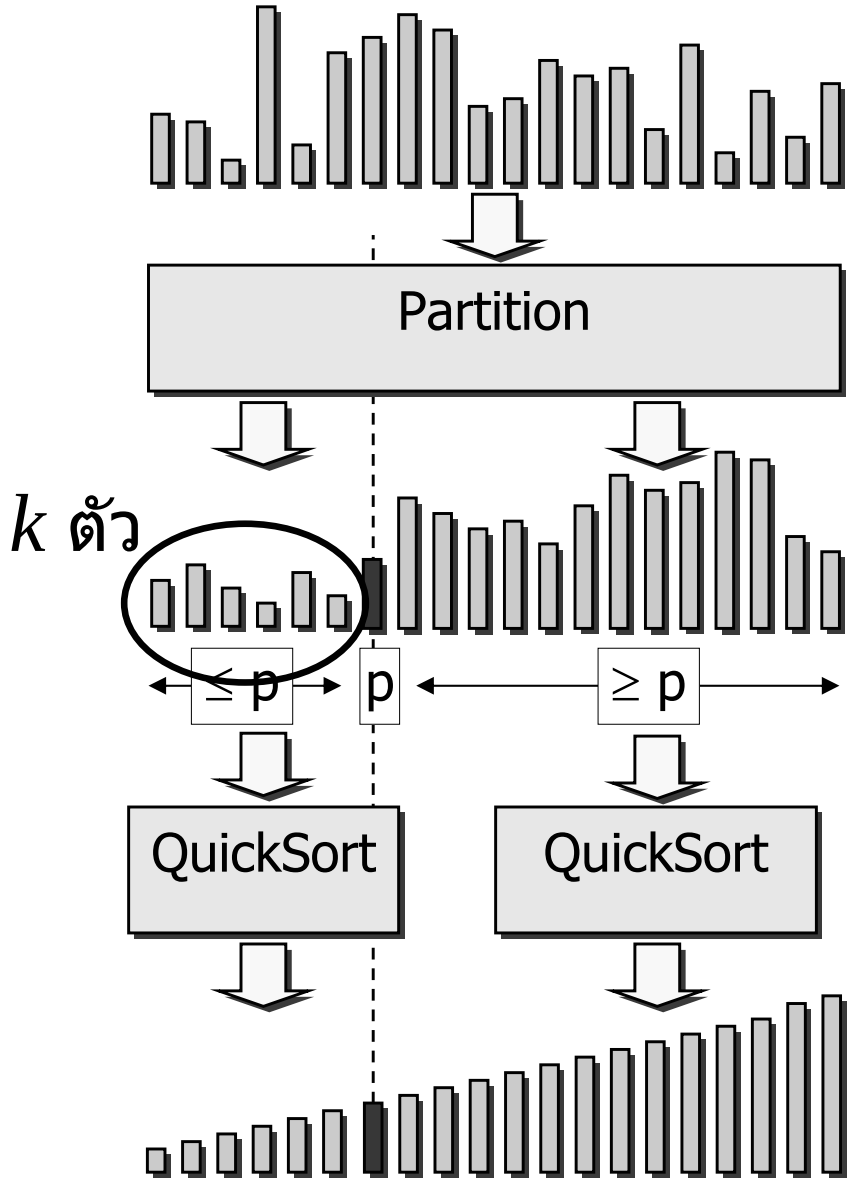
การแบ่งส่วน (partition)




```
int partition(int[] d, int left, int right) {
  int p = d[left];
  int i = left, j = right + 1;
  while (i < j) {
    while (d[--j] > p) ;
    while (d[++i] < p) if (i == right) break;
    if (i < j) swap(d, i, j);
  }
  swap(d, left, j);
  return j;
}
```

การเรียงลำดับแบบเร็ว : วิเคราะห์

ให้ $c(n)$ คือจำนวนครั้งของการเปรียบเทียบเพื่อเรียงลำดับข้อมูล n ตัว



$n-1$

$$c(n) = c(k) + c(n-k-1) + n - 1$$

$c(k) + c(n-k-1)$

จำนวนการเปรียบเทียบ : กรณีเร็วสุด

$$c(n) = c(k) + c(n-k-1) + n - 1$$

$$c_{min}(n) = c_{min}(\lfloor n/2 \rfloor) + c_{min}(n - \lfloor n/2 \rfloor - 1) + n - 1$$

$$\leq 2c_{min}(n/2) + n - 1$$

$$= n \log_2 n - n + 1$$

$$= O(n \log n)$$

จำนวนการเปรียบเทียบ : กรณีช้าสุด

$$c(n) = c(k) + c(n - k - 1) + n - 1$$

$$c_{max}(n) = c\,(0) + c_{max}(n - 1) + n - 1$$

$$= c_{max}(n - 1) + n - 1$$

$$= c_{max}(n - 2) + n - 2 + n - 1$$

$$\ldots$$

$$= c_{max}(1) + 1 + 2 + \ldots + n - 2 + n - 1$$

$$= n(n - 1)/2$$

$$= \Theta(n^2)$$

จำนวนการเปรียบเทียบ : กรณีเฉลี่ย

$$c(n) = c(k) + c(n-k-1) + n - 1$$

$$c_{\text{avg}}(n) = \frac{1}{n}\sum_{k=0}^{n-1}\left(c_{\text{avg}}(k) + c_{\text{avg}}(n-k-1)\right) + (n-1)$$

$$= \frac{2}{n}\sum_{k=0}^{n-1} c_{\text{avg}}(k) + (n-1)$$

$$nc_{\text{avg}}(n) = 2\sum_{k=0}^{n-1} c_{\text{avg}}(k) + n(n-1)$$

$$(n-1)c_{\text{avg}}(n-1) = 2\sum_{k=0}^{n-2} c_{\text{avg}}(k) + (n-1)(n-2)$$

$$nc_{\text{avg}}(n) = (n+1)c_{\text{avg}}(n-1) + 2(n-1)$$

จำนวนการเปรียบเทียบ : กรณีเฉลี่ย

$$nc_{\text{avg}}(n) = (n+1)c_{\text{avg}}(n-1) + 2(n-1)$$

$$\frac{c_{\text{avg}}(n)}{n+1} = \frac{c_{\text{avg}}(n-1)}{n} + \frac{2(n-1)}{n(n+1)}$$

$$= \frac{c_{\text{avg}}(1)}{2} + 2\sum_{i=2}^{n} \frac{(i-1)}{i(i+1)}$$

$$\approx 2\sum_{i=2}^{n} \frac{1}{(i+2)}$$

$$= 2(\ln n + \mathrm{O}(1))$$

$$c_{\text{avg}}(n) = 2n \ln n + \mathrm{O}(n)$$

$$\approx 1.39n \log_2 n + \mathrm{O}(n)$$

$$= \mathrm{O}(n \log n)$$

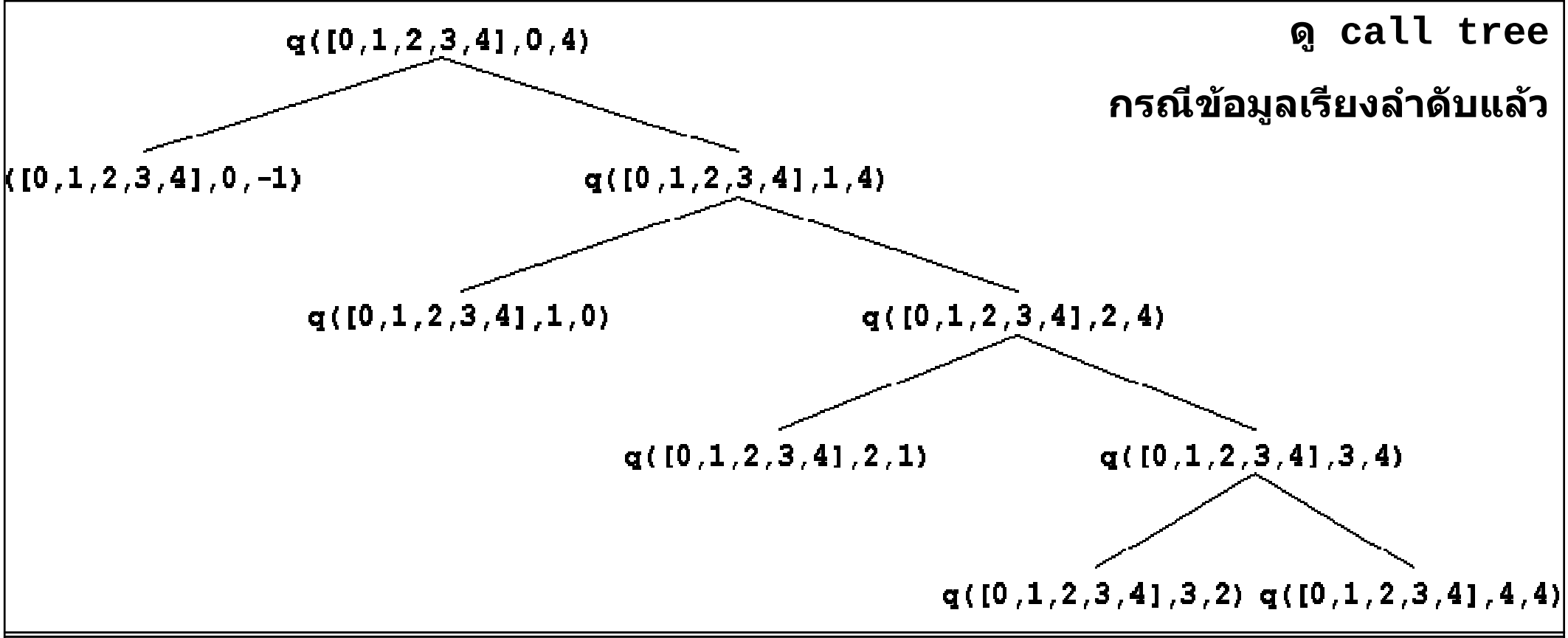
การเลือกตัวหลัก (pivot)

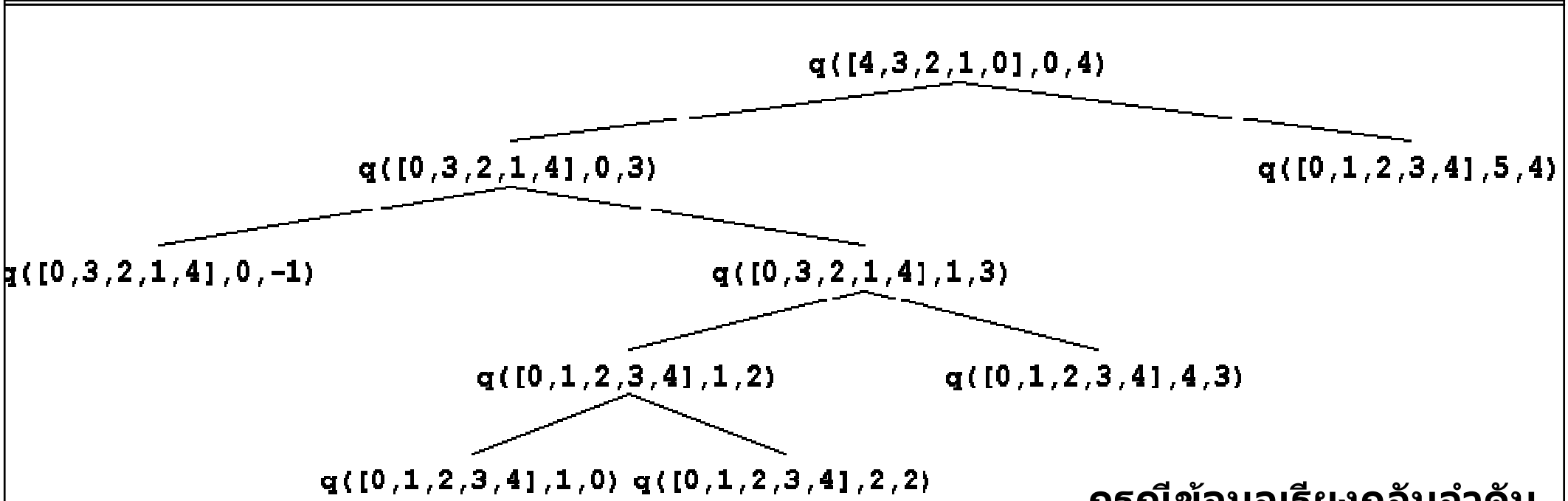
- ที่ผ่านมา : เลือกตัวซ้ายสุดเป็นตัวหลัก
- ถ้าข้อมูลเริ่มต้นเรียงลำดับอยู่แล้ว
  - หลังการแบ่งส่วน ชุดซ้ายมี 0 ตัวเสมอ
  - เกิดกรณีช้าสุด : $\Theta(n^2)$
  - ถ้าใช้โปรแกรมที่เขียนแบบเวียนเกิด
  - เกิดการเรียกเวียนเกิดซ้อน ๆ กันจำนวน n – 1 ครั้ง
  - มีโอกาสเกิด stack overflow
- สุ่มเลือกตัวหลัก
  - โอกาสเกิดกรณีช้าสุดมีน้อยมาก ๆ

```
int partition(int[] d, int left, int right) {
  int m = left + (int)(Math.random()*(right–left+1));
  swap(d, left, m);
  int p = d[left];
  ...
```

ดู call tree

กรณีข้อมูลเรียงลำดับแล้ว





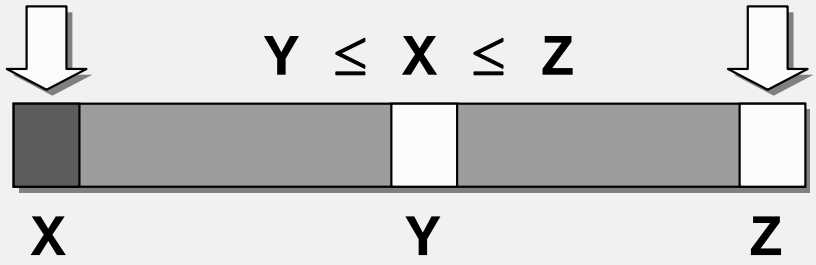
กรณีข้อมูลเรียงกลับลำดับ

การเลือกมัธยฐานสามเป็นตัวหลัก

- ใช้ตัวหลักที่ได้มาจากมัธยฐานของตัวซ้าย ขวา และกลางของข้อมูล

```
int partition(int[] d, int left, int right) {
  int c = low + (right - left)/2;
  if (d[left]  < d[c]) swap(d, left, c);
  if (d[right] < d[c]) swap(d, c, right);
  if (d[right] < d[left]) swap(d, left, right);
  int p = d[left];
  int i = left, j = right + 1;
  while (i < j) {
    while (d[--j] > p) ;
    while (d[++i] < p) if (i==right) break;
    if (i < j) swap(d, i, j);
  }
  swap(d, left, j);
  return j;
}
```

ลองหา worst case input



Tuned Quicksort

- ใช้ insertion sort เมื่อ n < 7
- ถ้า 7 ≤ n < 40
  - เลือก pivot จาก median ของตัวซ้าย กลาง ขวา
- ถ้า n > 40
  - เลือก pivot จาก median of median of three ของข้อมูล 9 ตัว
- three-way partition (<p)*,(=p)*,(>p)*

Jon L. Bentley and M. Douglas McIlroy's "Engineering a Sort Function", Software-Practice and Experience, Vol. 23(11) P. 1249-1265 (November 1993).

Tuned Mergesort

- ใช้ insertion sort เมื่อ n < 7

```
void mSortR(int[] d, int left, int right, int[] t) {
  if (right-left < 6) {
    insertionSort(d, left, right);
  } else {
    int m = left + ((right - left) / 2);
    mSortR(t, left, m, d);
    mSortR(t, m + 1, right, d);
    if (d[m]<d[m+1])
      System.arraycopy(t, left, d, left, right-left+1);
    else
      merge(t, left, m, right, d);
  }
}
```

This is an optimization that results in faster sorts for nearly ordered lists.

เปรียบเทียบเวลาเรียงลำดับ

ข้อมูลเริ่มต้นเรียงลำดับ (เวลาเป็น ms)

n	เชลล์	ฮีป	ผสาน	เร็ว	เร็ว (สุ่ม)	เร็ว (M3)
1000	**0.20**	0.96	0.38	11.66	0.48	0.29
10000	**2.96**	16.74	4.77	-	5.94	4.38
100000	64.67	168.72	58.99	-	67.60	**46.87**
1000000	1153.70	2005.57	724.59	-	787.78	**503.23**

เปรียบเทียบเวลาเรียงลำดับ

ข้อมูลเริ่มต้นเรียงกลับลำดับ (เวลาเป็น ms)

n	เชลล์	ฮีป	ผสาน	เร็ว	เร็ว (สุ่ม)	เร็ว (M3)
1000	0.40	0.77	0.39	10.51	0.49	**0.30**
10000	6.45	11.51	9.86	-	6.23	**5.95**
100000	113.26	154.61	60.88	-	71.68	**48.66**
1000000	1692.68	1997.14	745.68	-	809.86	**541.56**

เปรียบเทียบเวลาเรียงลำดับ

ข้อมูลเริ่มต้นแบบสุ่ม (เวลาเป็น ms)

n	เชลล์	ฮีป	ผสาน	เร็ว	เร็ว (สุ่ม)	เร็ว (M3)
1000	0.97	0.96	0.56	0.58	0.94	**0.52**
10000	12.82	15.53	10.27	**6.88**	15.37	8.89
100000	199.27	224.27	111.66	106.92	121.83	**102.17**
1000000	3143.03	4303.59	1682.07	**1591.25**	1810.50	1600.21

การหลีกเลี่ยง stack overflow

```
void qSortR(int[] d, int left, int right){
  if (left < right) {
    int j = partition(d, left, right);
    qSortR(d, left, j-1);
    qSortR(d, j+1, right);
  }
}
```

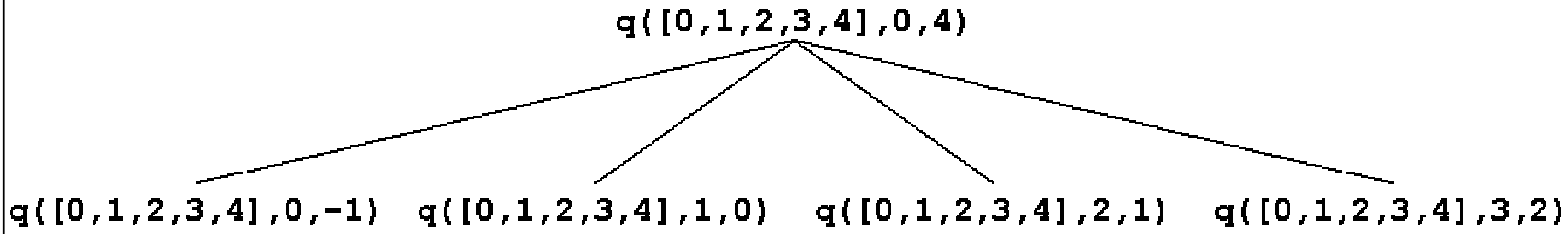
```
while (left < right) {
  int j = partition(d, left, right);
  qSortR(d, left, j-1);
  left = j+1;
}
```

```
while (left < right) {
  int j = partition(d, left, right);
  qSortR(d, j+1, right);
  right = j-1;
}
```

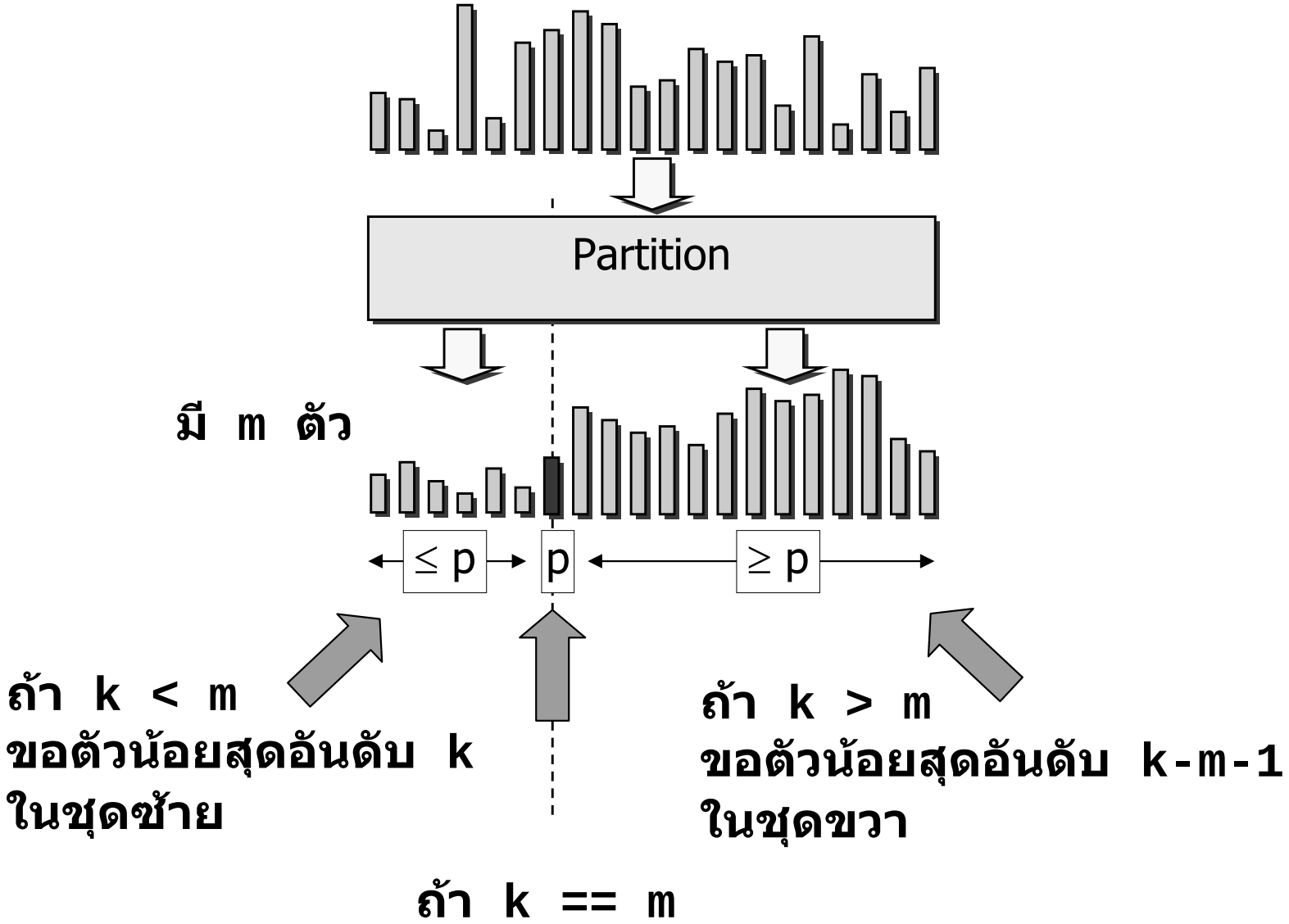
การหลีกเลี่ยง stack overflow

```
void qSortR(int[] d, int left, int right){
  while (left < right) {
    int j = partition(d, left, right);
    if (j - left < right - j) {
      qSortR(d, left, j - 1);
      left = j + 1;
    } else {
      qSortR(d, j + 1, right);
      right = j - 1;
    }
  }
}
```

ลองคิดดู : แย่สุดๆ เรียกซ้อนๆกันกี่ครั้ง ?



Selection : ขอตัวน้อยสุดอันดับ k



Selection

```
int select(int[] d, int k) {
  return selectR(d, 0, d.length-1, k);
}
int selectR(int[] d, int left, int right, int k) {
  if (left == right) return d[left];
  int j = partition(d, left, right);
  int m = j - left;
  if (k == m) return d[left + m];  // d[j]
  if (k < m)
    return selectR(d, left, j-1, k);
  else
    return selectR(d, j+1, right, k-m-1);
}
```

ตัวน้อยสุดอันดับ k จาก d[left] ถึง d[right]

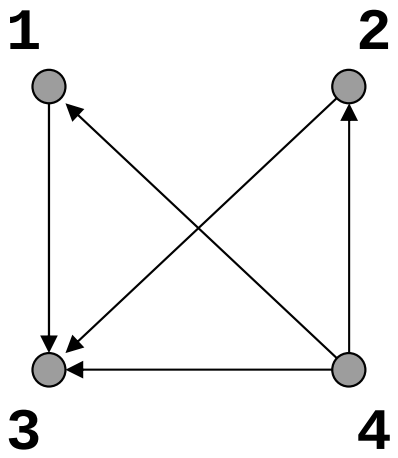
Selection

```
int select(int[] d, int k) {
  return selectR(d, 0, d.length-1, k);
}
int selectR(int[] d, int left, int right, int k) {
  if (left == right) return d[left];
  int j = partition(d, left, right);

  if (k == j) return d[k];
  if (k < j)
    return selectR(d, left, j-1, k);
  else
    return selectR(d, j+1, right, k);
}
```

ตัวน้อยสุดอันดับ k จาก d[0] ถึง d[d.length-1]

Celebrity Problem

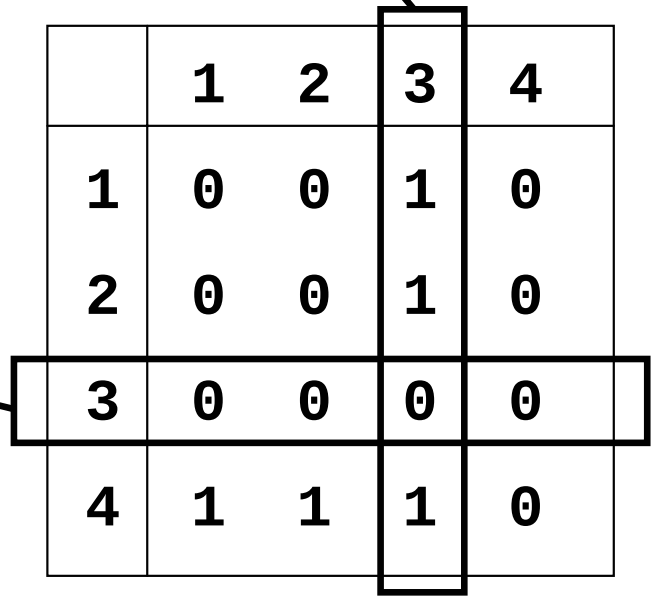
- อยากทราบว่าใครคือ "ดาวเด่น" ของงานสังคมที่มีผู้ร่วมงาน n คน
- "ดาวเด่น" คือผู้ซึ่งทุก ๆ คนในงานรู้จัก แต่เจ้าตัวไม่รู้จักใครเลย



	1	2	3	4
1	0	0	1	0
2	0	0	1	0
3	0	0	0	0
4	1	1	1	0

ลุย

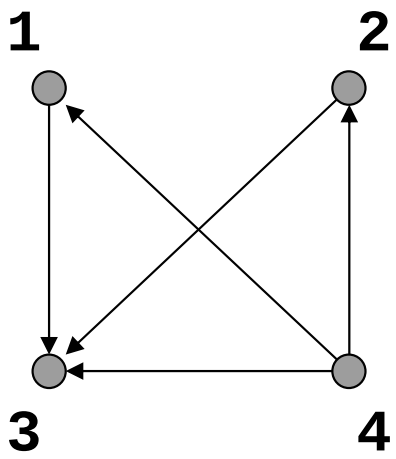
```
Celebrity ( a[1..n, 1..n] ) {
  for i = 1 to n {
    c1 = 0
    for k = 1 to n
      c1 = c1 + a[k,i]
    c2 = 0
    for k = 1 to n
      c2 = c2 + a[i,k]
    if (c1 == n-1 AND c2 == 0) return i
  }
  return 0
}
```

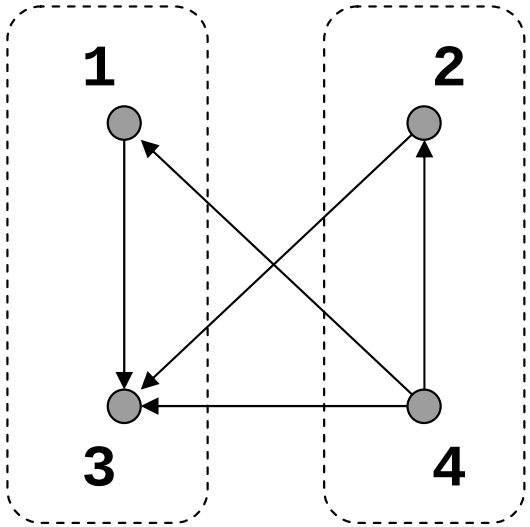


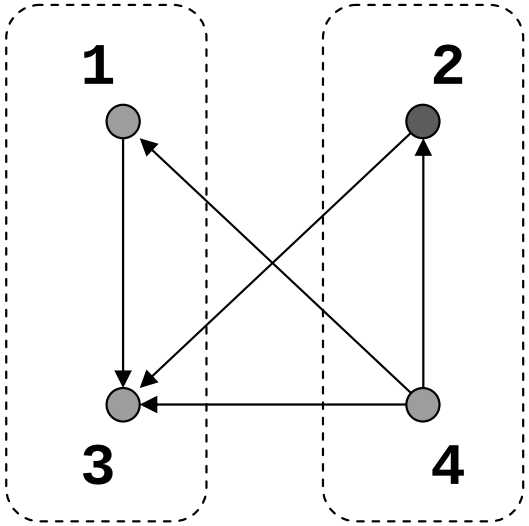
	1	2	3	4
1	0	0	1	0
2	0	0	1	0
3	0	0	0	0
4	1	1	1	0

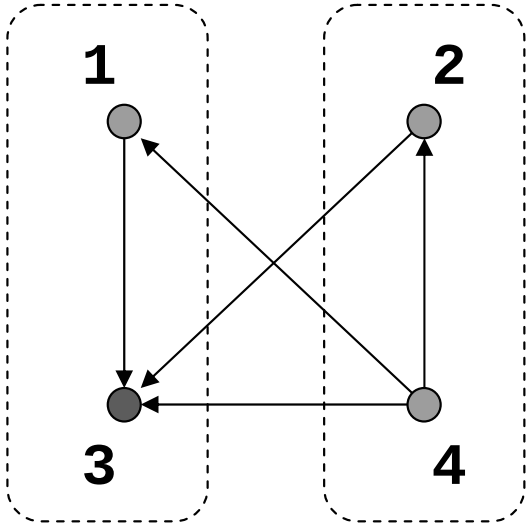
$O(n^2)$

Divide & Conquer





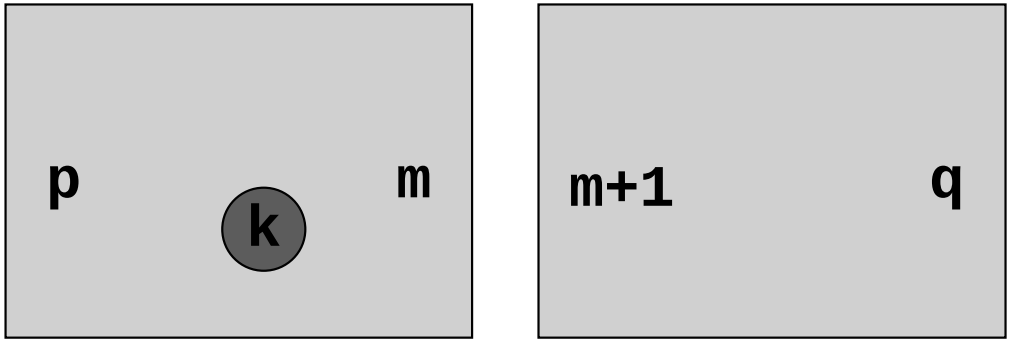




Divide & Conquer (50 : 50)

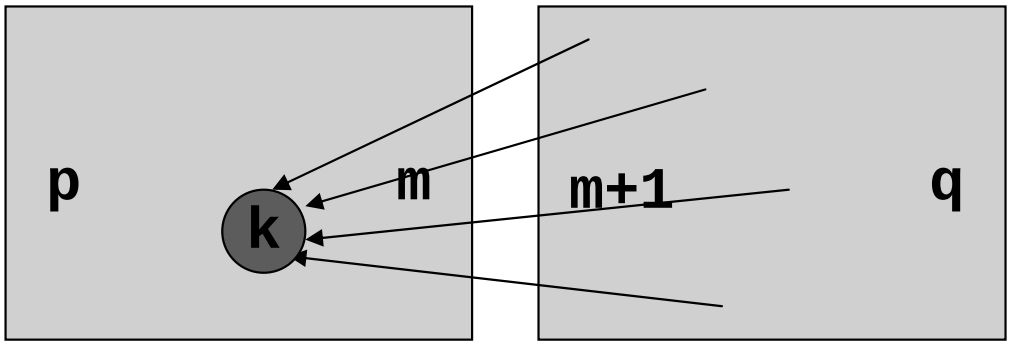

```
Celebrity_DQ( a[1..n, 1..n], p, q ) {
```

Divide & Conquer (50 : 50)



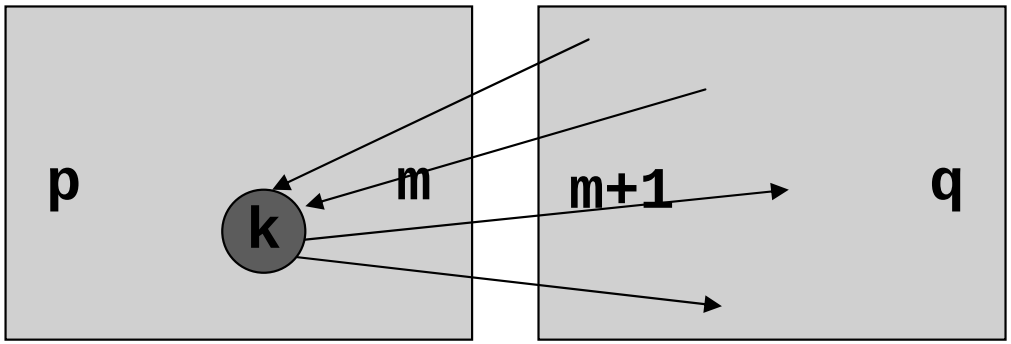

```
Celebrity_DQ( a[1..n, 1..n], p, q ) {
    if ( p == q ) return p
    m = (p+q)/2
    k = Celebrity_DQ( a, p, m )
```

Divide & Conquer (50 : 50)



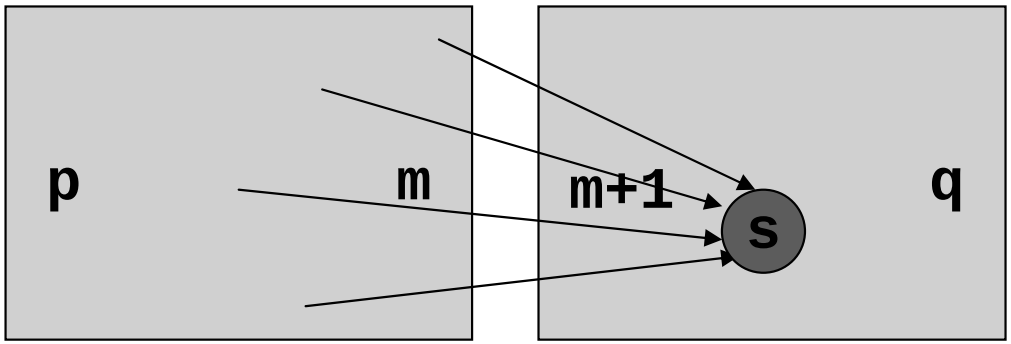

```
Celebrity_DQ( a[1..n, 1..n], p, q ) {
    if ( p == q ) return p
    m = (p+q)/2
    k = Celebrity_DQ( a, p, m )
    k = Celebrity_Check( a, k, m+1, q )
    if ( k > 0 ) return k
```

Divide & Conquer (50 : 50)




```
Celebrity_DQ( a[1..n, 1..n], p, q ) {
    if ( p == q ) return p
    m = (p+q)/2
    k = Celebrity_DQ( a, p, m )
    k = Celebrity_Check( a, k, m+1, q )
    if ( k > 0 ) return k
```

Divide & Conquer (50 : 50)



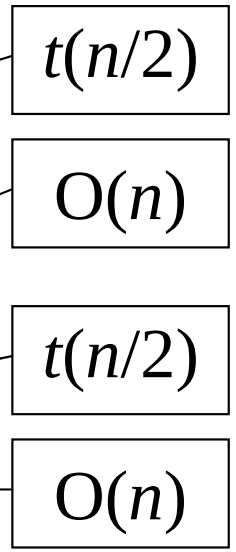

```
Celebrity_DQ( a[1..n, 1..n], p, q ) {

   if ( p == q ) return p
   m = (p+q)/2
   k = Celebrity_DQ( a, p, m )
   k = Celebrity_Check( a, k, m+1, q )
   if ( k > 0 ) return k

   s = Celebrity_DQ( a, m+1, q )
   s = Celebrity_Check( a, s, p, m )
   return s

}
```

Divide & Conquer (50 : 50)

$$t(n) \leq 2t(n/2) + \mathrm{O}(n)$$

$$t(n) = \mathrm{O}(n \log n)$$

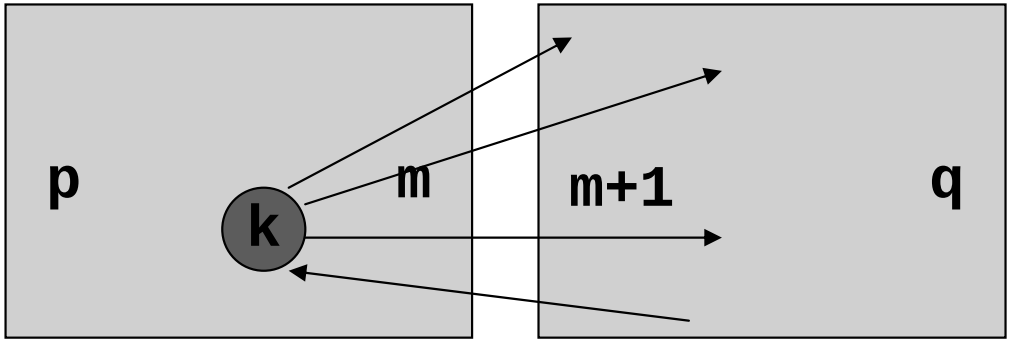
```
Celebrity_DQ( a[1..n, 1..n], p, q ) {
    if ( p == q ) return p
    m = (p+q)/2
    k = Celebrity_DQ( a, p, m )
    k = Celebrity_Check( a, k, m+1, q )
    if ( k > 0 ) return k

    s = Celebrity_DQ( a, m+1, q )
    s = Celebrity_Check( a, s, p, m )
    return s

}
```



Divide & Conquer (50 : 50)

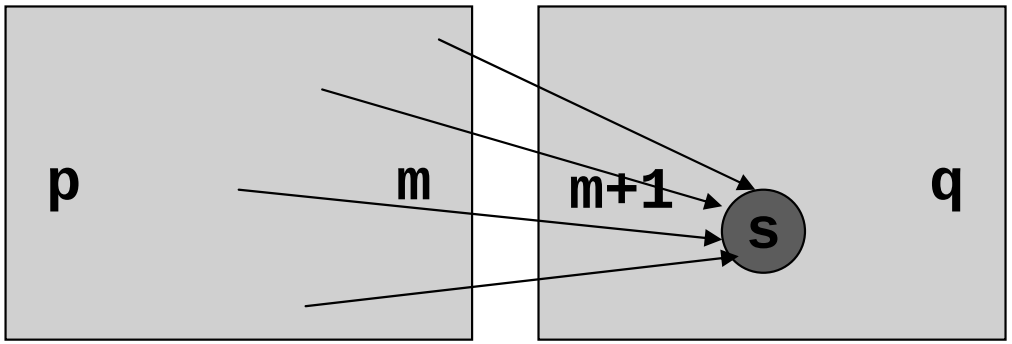
0	0	1	0	0	1	1	1
1	0	1	1	1	0	1	0
0	0	0	0	0	1	1	1
1	0	1	0	1	1	1	0
1	1	0	0	0	1	1	1
1	0	0	0	0	0	1	1
0	0	0	0	0	0	0	0
1	0	1	0	0	1	1	0

Divide & Conquer (50 : 50)



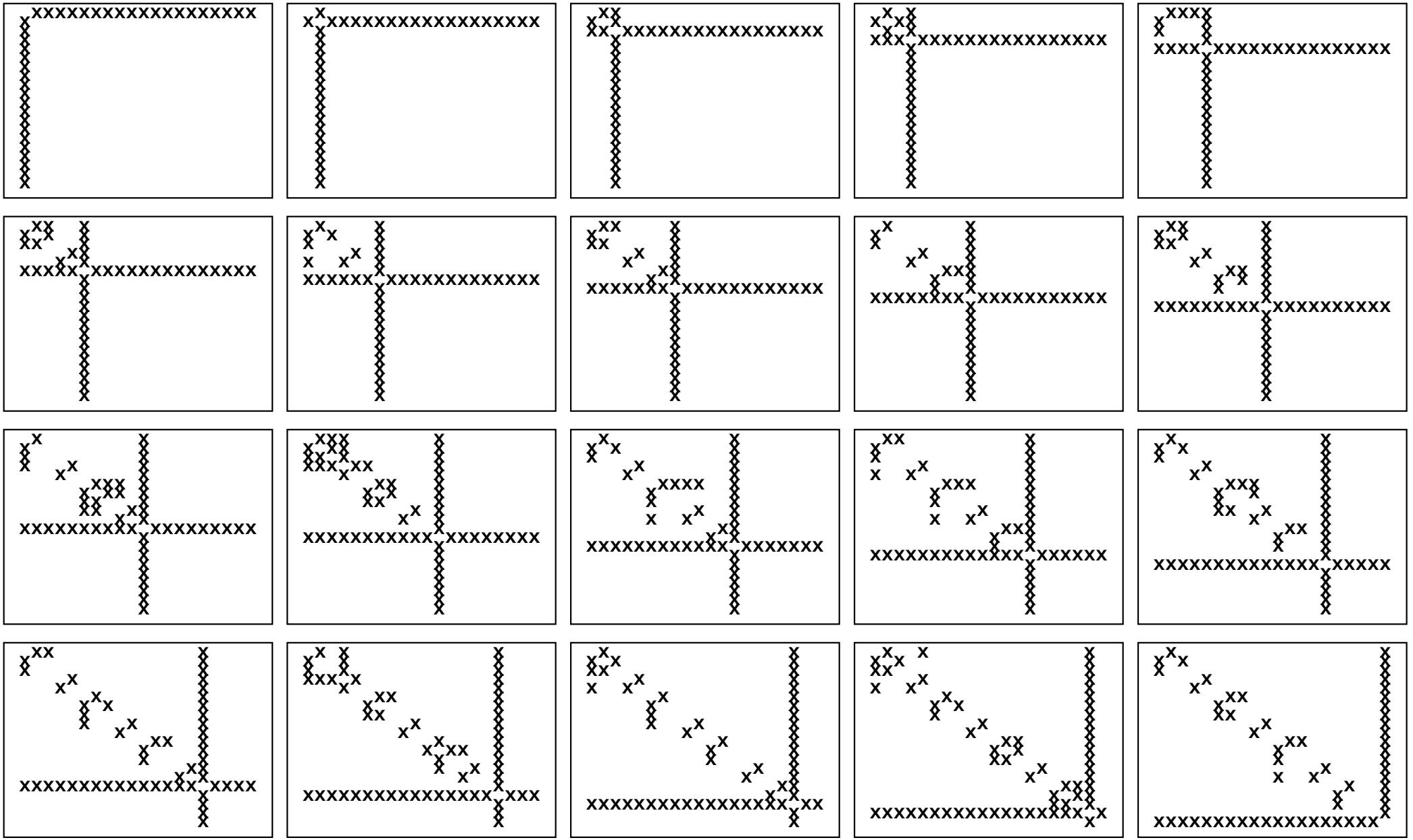
0	0	1	0	0	1	1	1
1	0	1	1	1	0	1	0
0	0	0	0	0	1	1	1
1	0	1	0	1	1	1	0
1	1	0	0	0	1	1	1
1	0	0	0	0	0	1	1
0	0	0	0	0	0	0	0
1	0	1	0	0	1	1	0

Divide & Conquer (50 : 50)



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

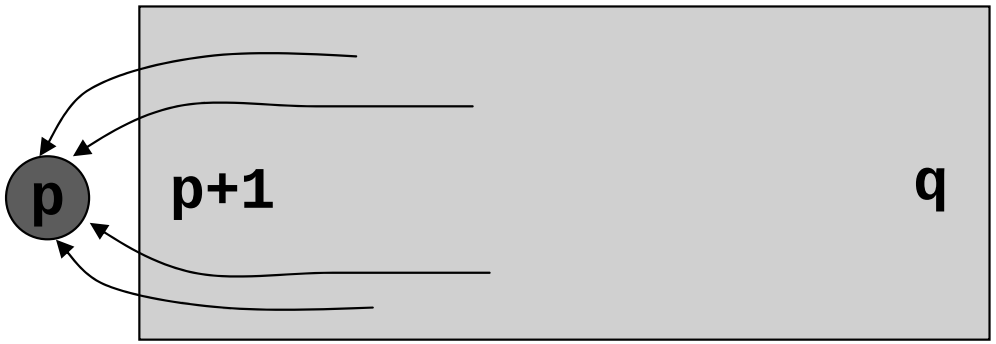
การทดลอง : เมทริกซ์ขนาด 20x20



Divide & Conquer (1 : n – 1)

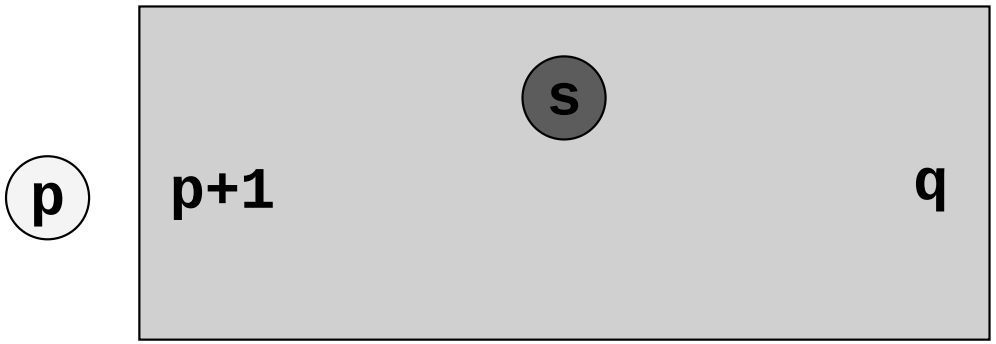

```
Celebrity_DQ( a[1..n, 1..n], p, q ) {
```

Divide & Conquer (1 : n – 1)



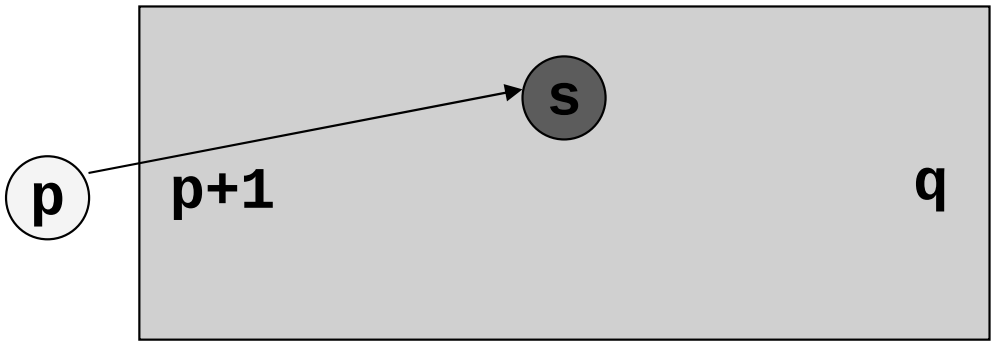

```
Celebrity_DQ( a[1..n, 1..n], p, q ) {

   if ( p == q ) return p

   k = Celebrity_Check( a, p, p+1, q )
   if ( k > 0 ) return k
```

Divide & Conquer (1 : n – 1)




```
Celebrity_DQ( a[1..n, 1..n], p, q ) {

   if ( p == q ) return p

   k = Celebrity_Check( a, p, p+1, q )
   if ( k > 0 ) return k

   s = Celebrity_DQ( a, p+1, q )



}
```

Divide & Conquer (1 : n – 1)




```
Celebrity_DQ( a[1..n, 1..n], p, q ) {

   if ( p == q ) return p

   k = Celebrity_Check( a, p, p+1, q )
   if ( k > 0 ) return k

   s = Celebrity_DQ( a, p+1, q )
   s = Celebrity_Check( a, s, p, p )
   return s

}
```

Divide & Conquer (1 : n – 1)

$$t(n) \leq t(n-1) + O(n)$$

$$t(n) = O(n^2)$$

```
Celebrity_DQ( a[1..n, 1..n], p, q ) {
   if ( p == q ) return p
   k = Celebrity_Check( a, p, p+1, q )
   if ( k > 0 ) return k
   s = Celebrity_DQ( a, p+1, q )
   s = Celebrity_Check( a, s, p, p )
   return s
}
```

$O(n)$

$t(n\text{-}1)$

$\Theta(1)$

Divide & Conquer (1 : n – 1)

$$t(n) \leq t(n-1) + \mathrm{O}(n)$$

$$t(n) = \mathrm{O}(n^2)$$

```
Celebrity_DQ( a[1..n, 1..n], p, q ) {
   if ( p == q ) return p
   k = Celebrity_Check( a, p, p+1, q )
   if ( k > 0 ) return k
   s = Celebrity_DQ( a, p+1, q )
   s = Celebrity_Check( a, s, p, p )
   return s
}
```

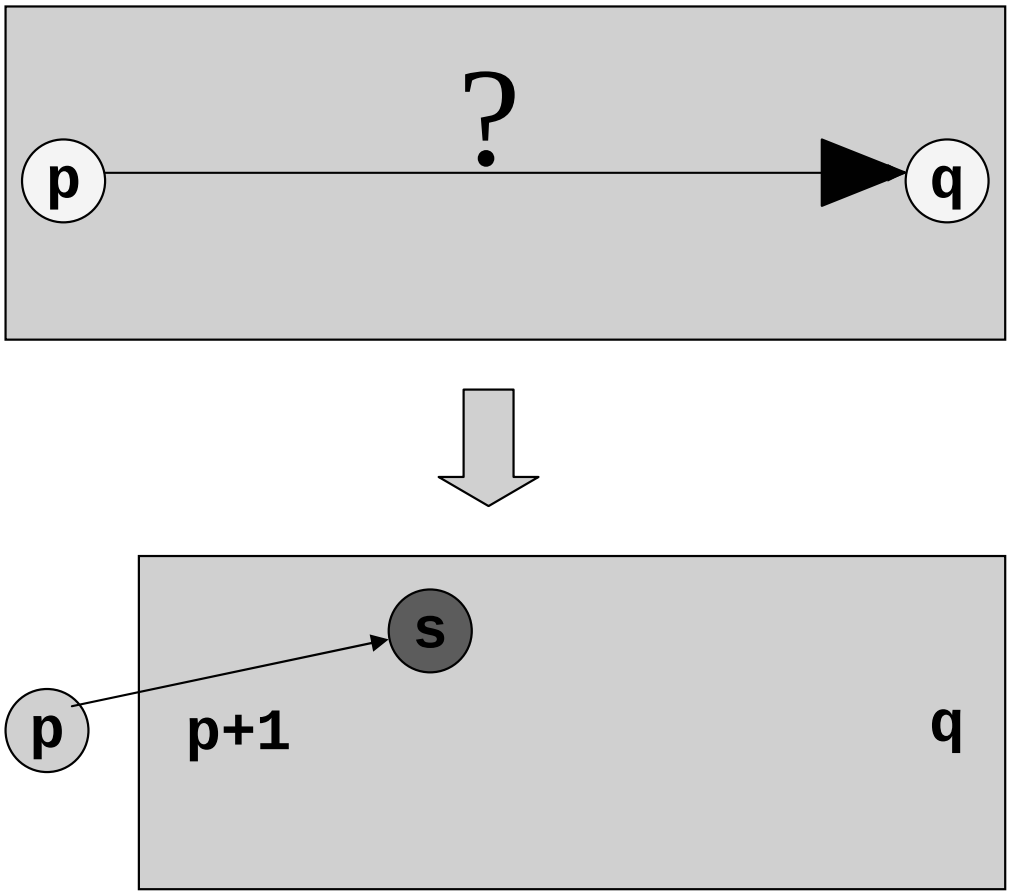
Decrease & Conquer

$$t(n) = t(n-1) + \Theta(1)$$

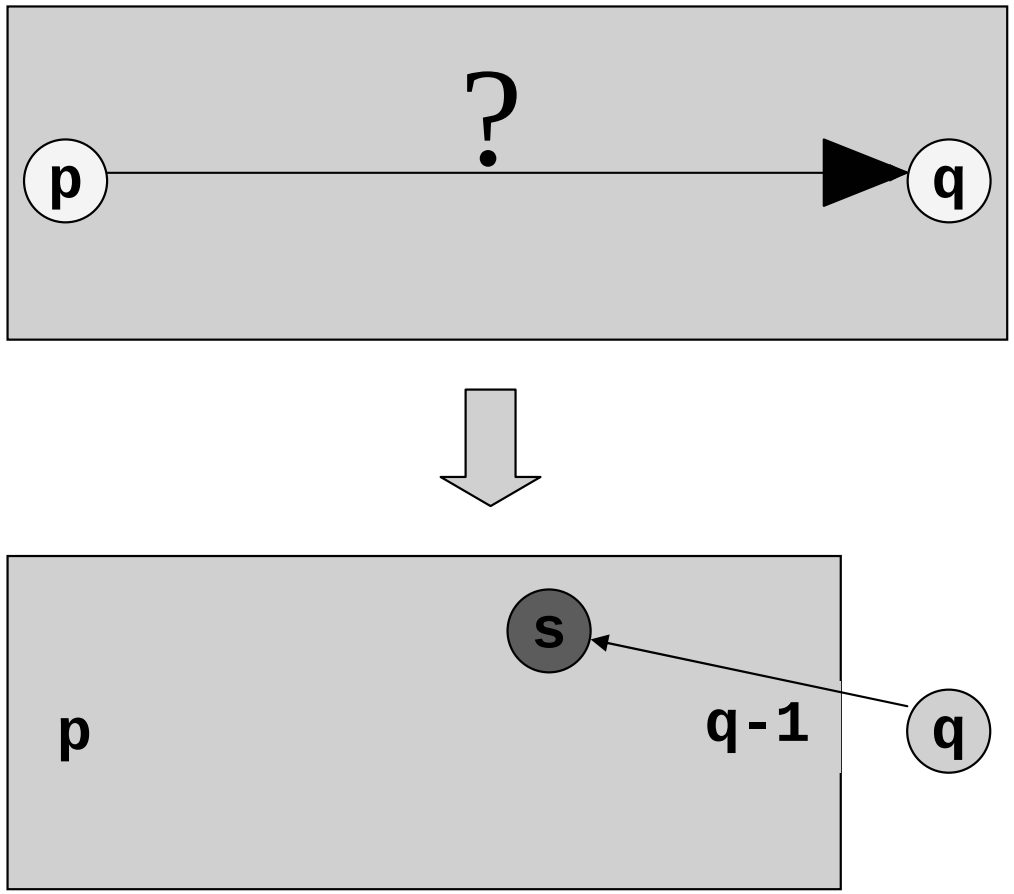
$$t(n) = \Theta(n)$$

```
Celebrity_DQ( a[1..n, 1..n], p, q ) {
   if ( p == q ) return p
   k = Celebrity_Check( a, p, p+1, q )
   if ( k > 0 ) return k
   s = Celebrity_DQ( a, p+1, q )
   s = Celebrity_Check( a, s, p, p )
   return s
}
```

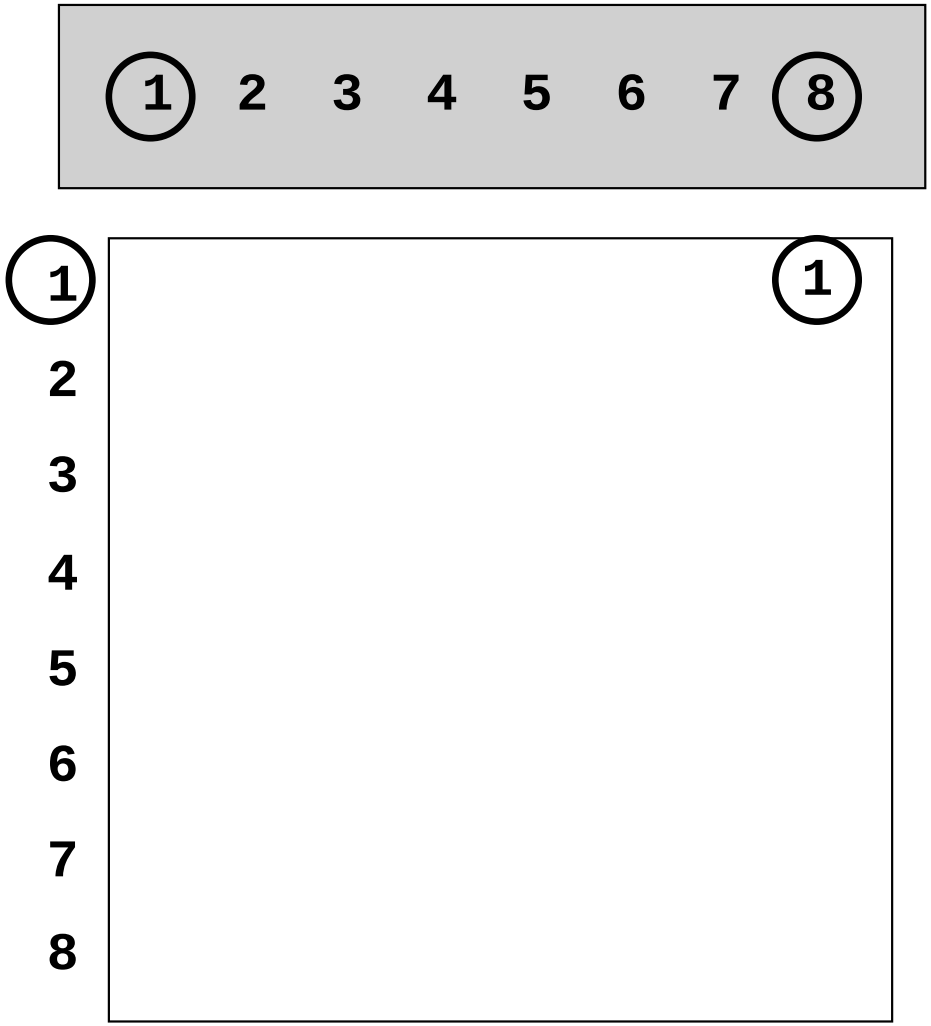
Decrease & Conquer



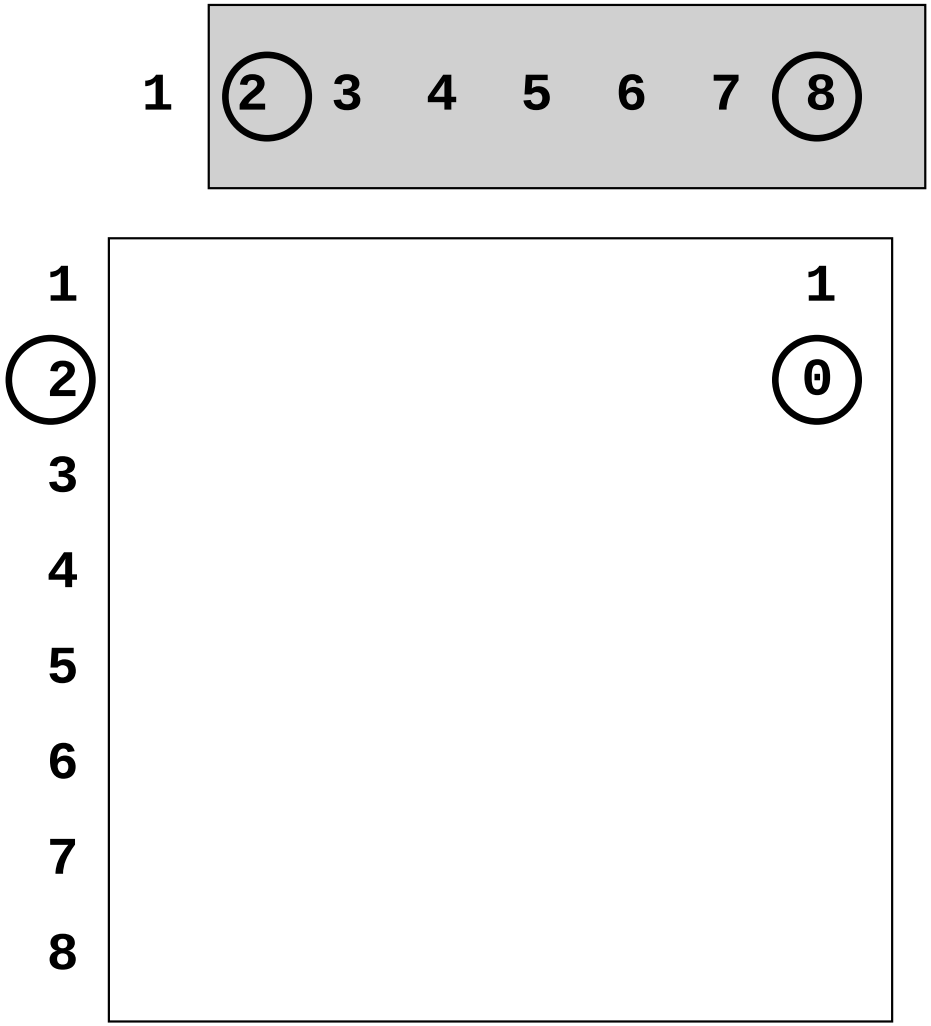
Decrease & Conquer



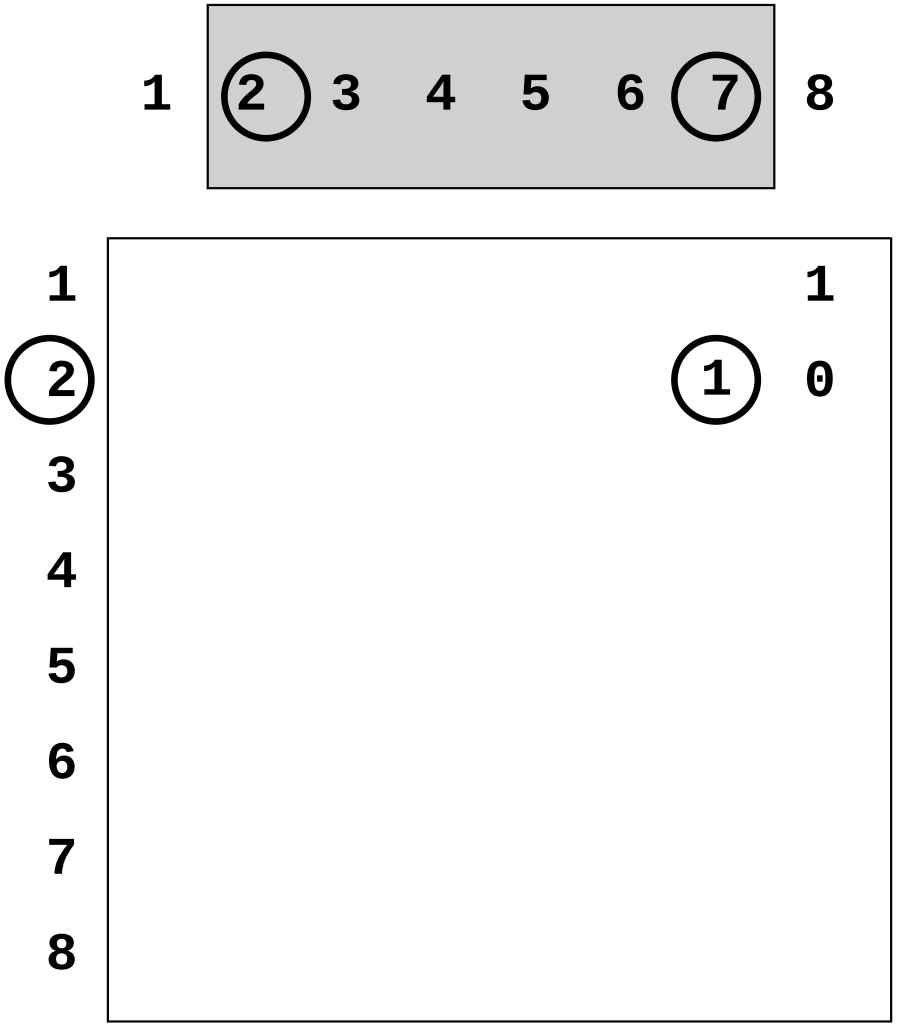
Decrease & Conquer



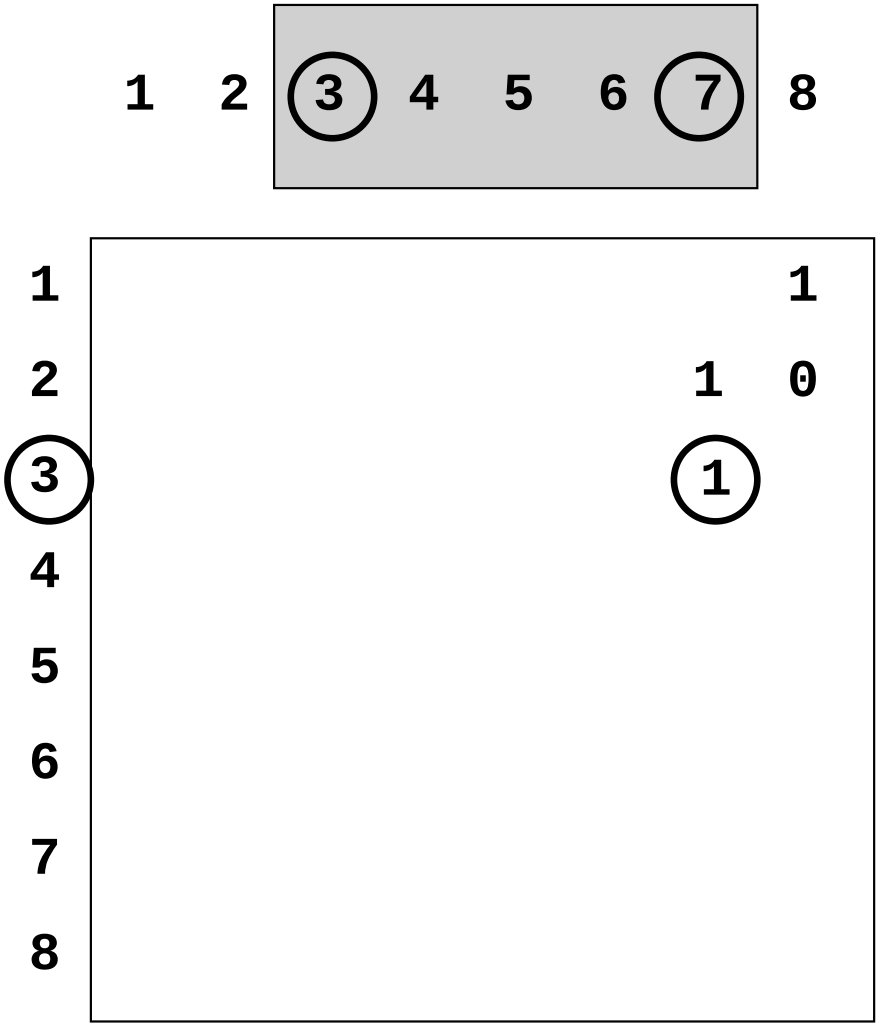
Decrease & Conquer



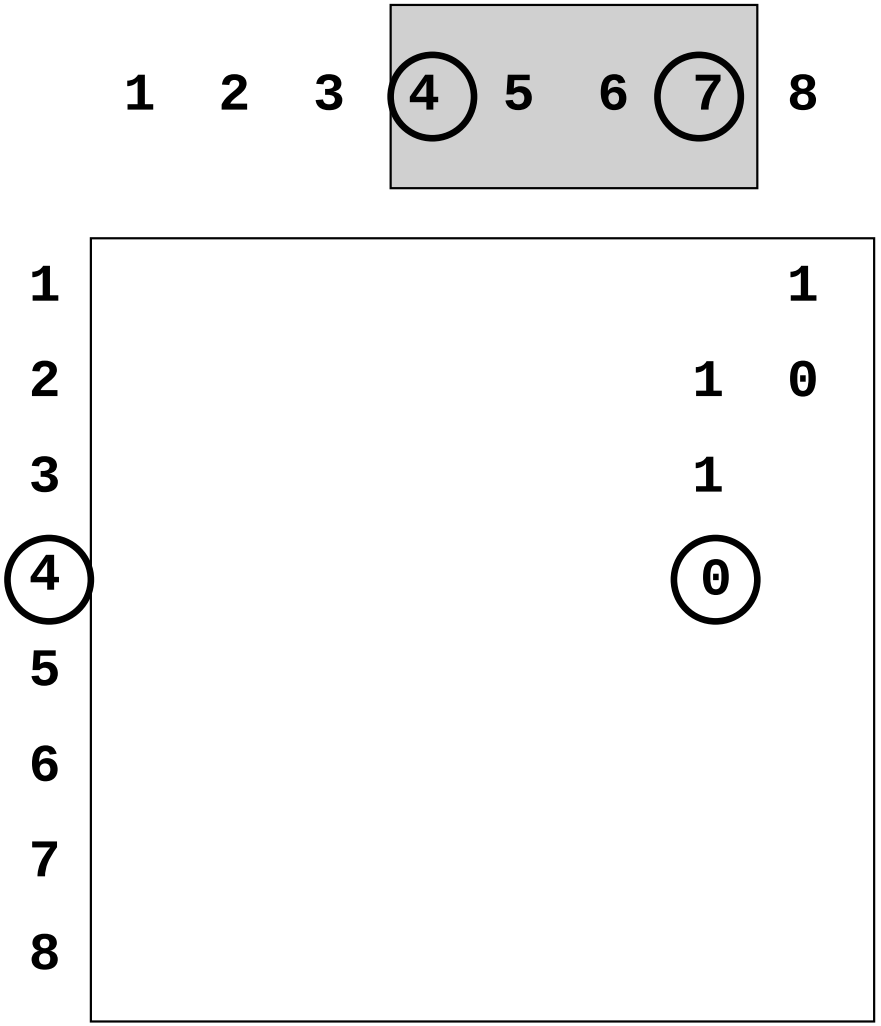
Decrease & Conquer



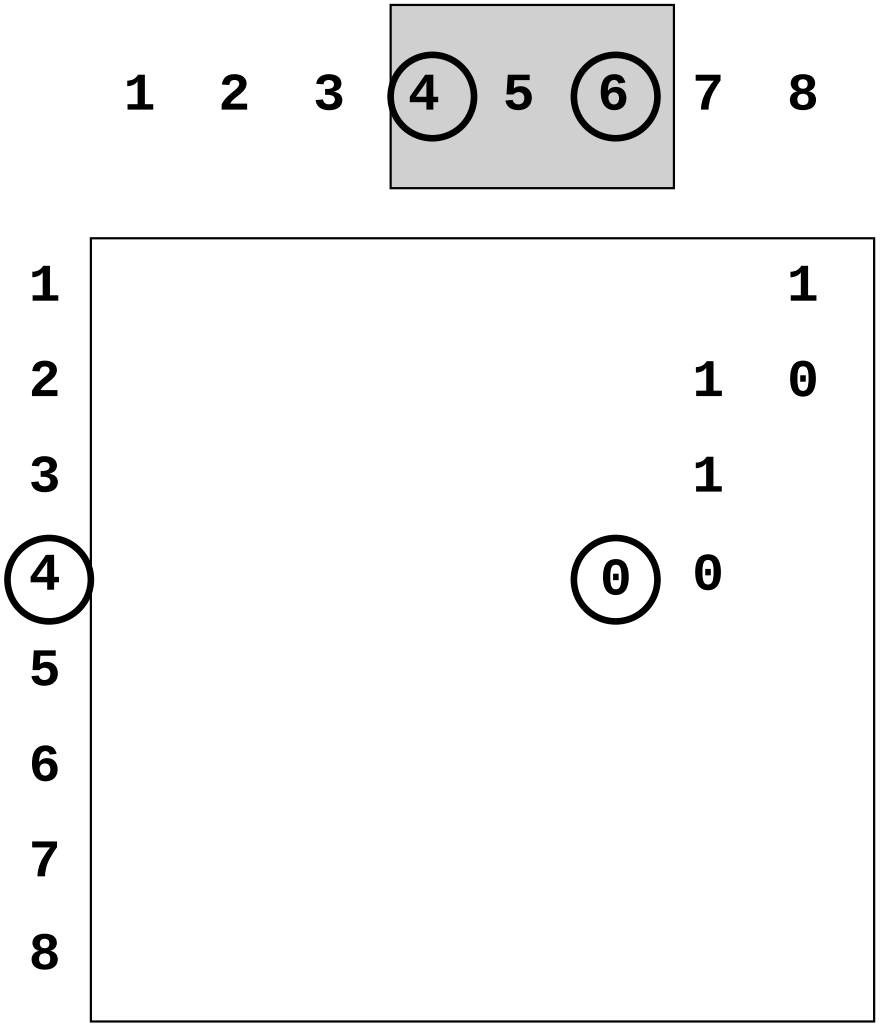
Decrease & Conquer



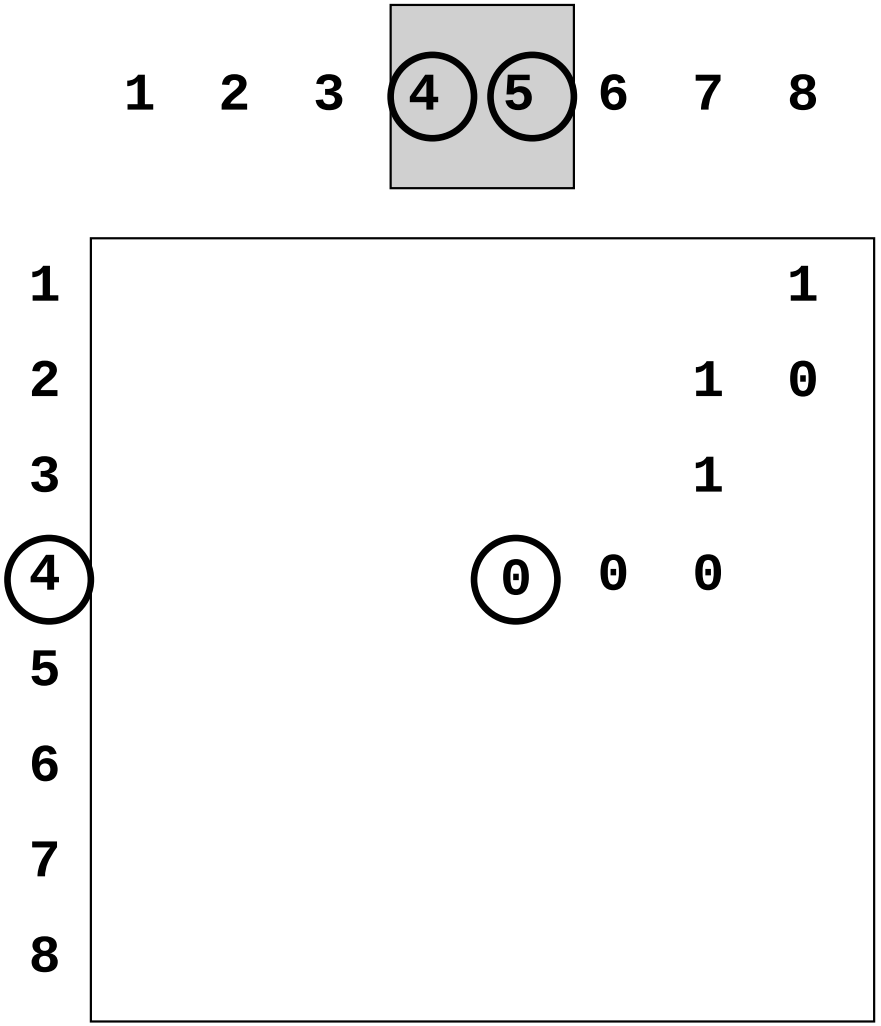
Decrease & Conquer



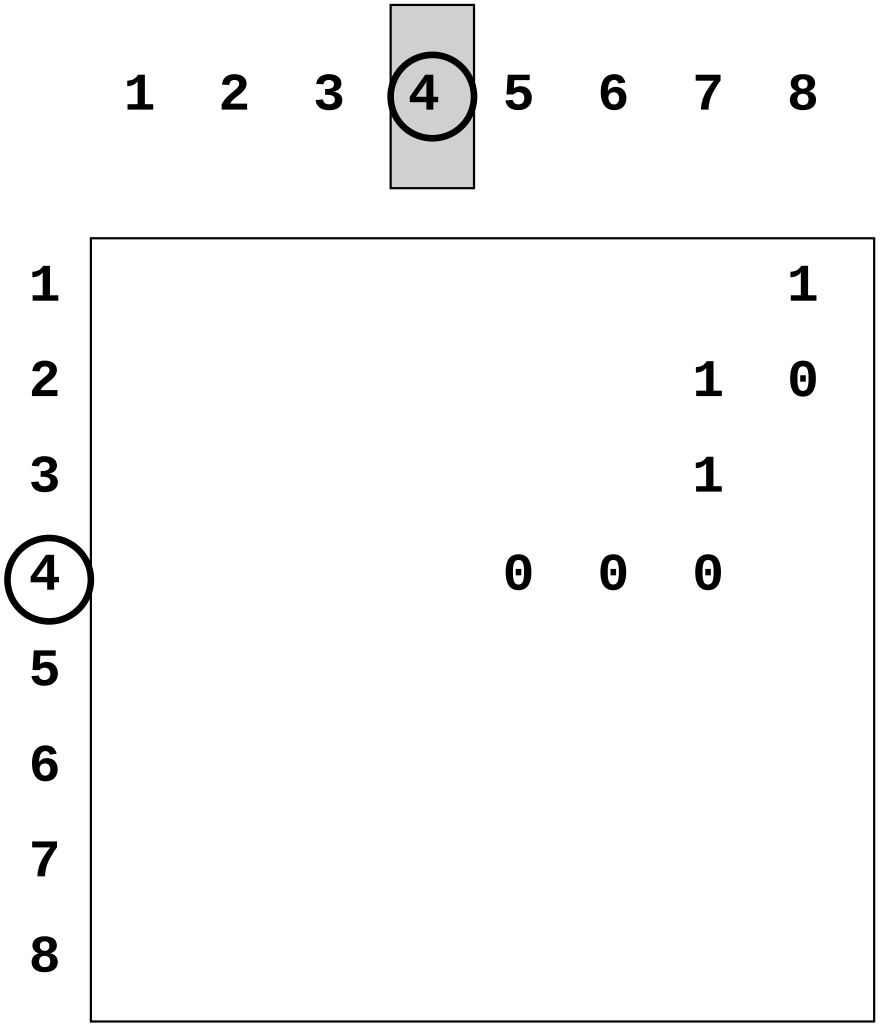
Decrease & Conquer



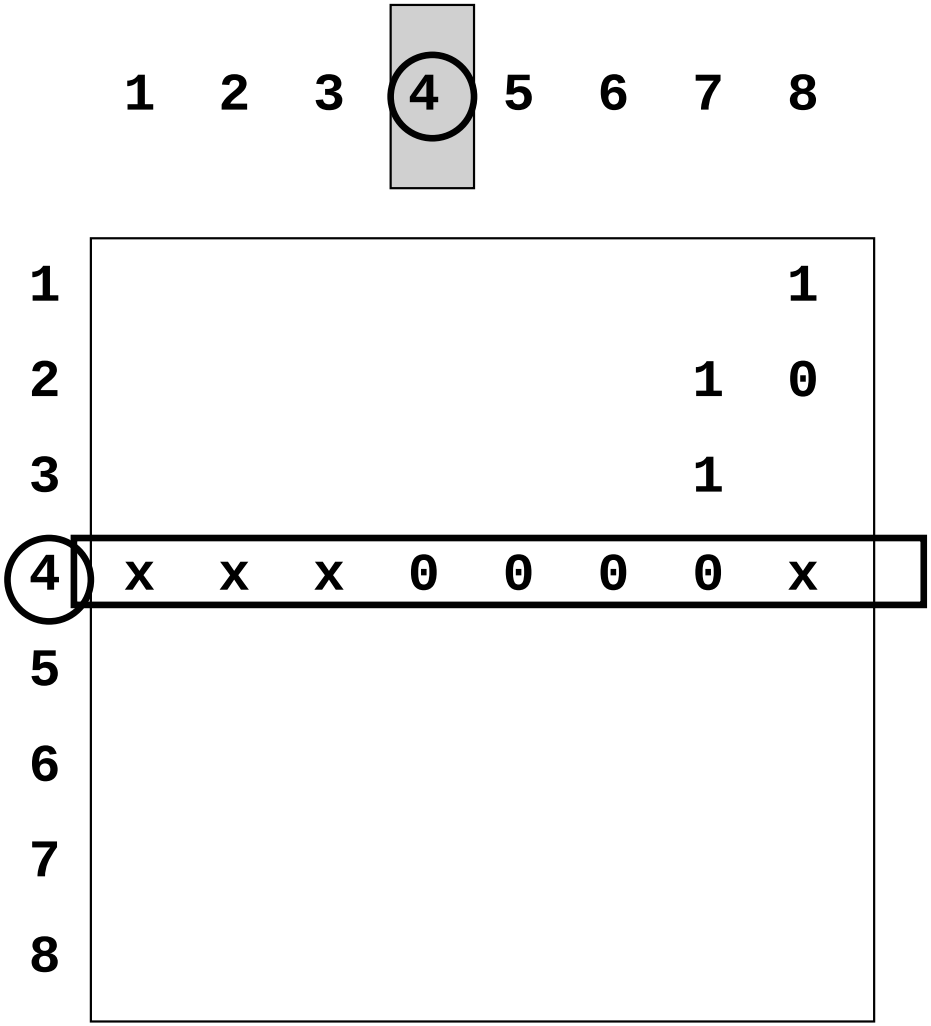
Decrease & Conquer



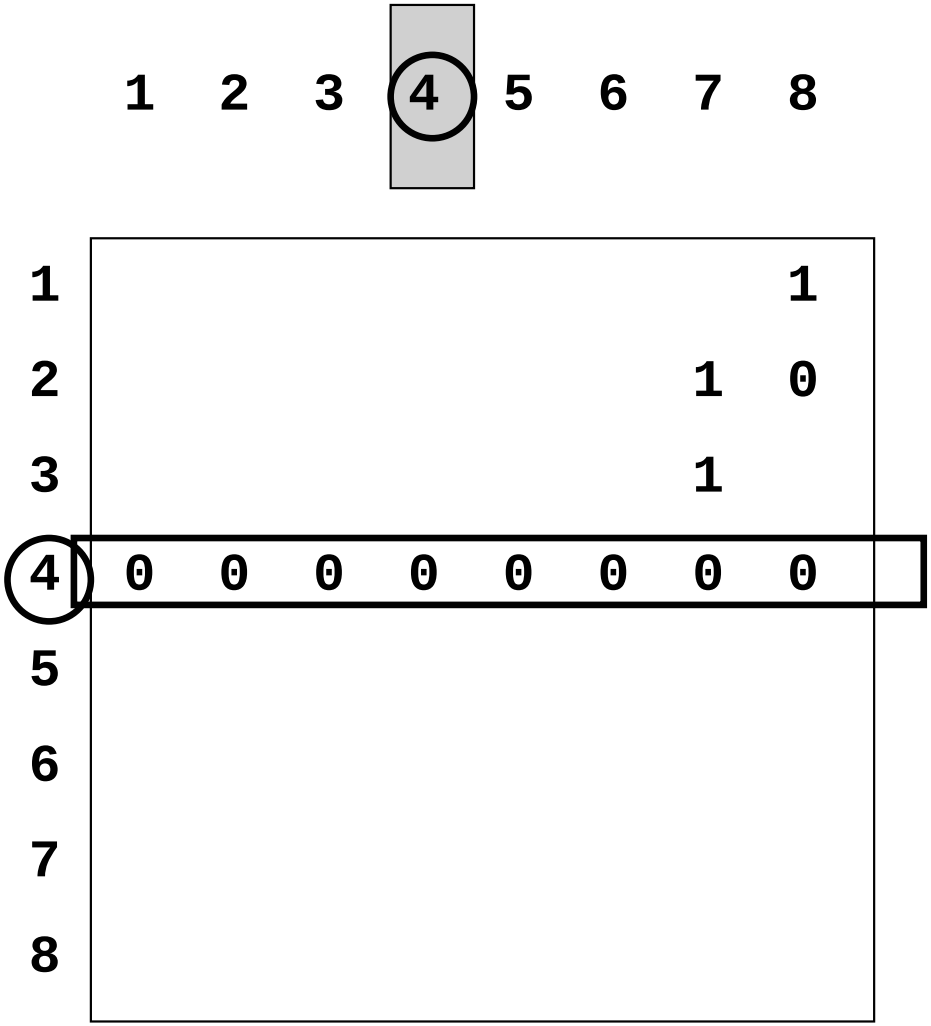
Decrease & Conquer



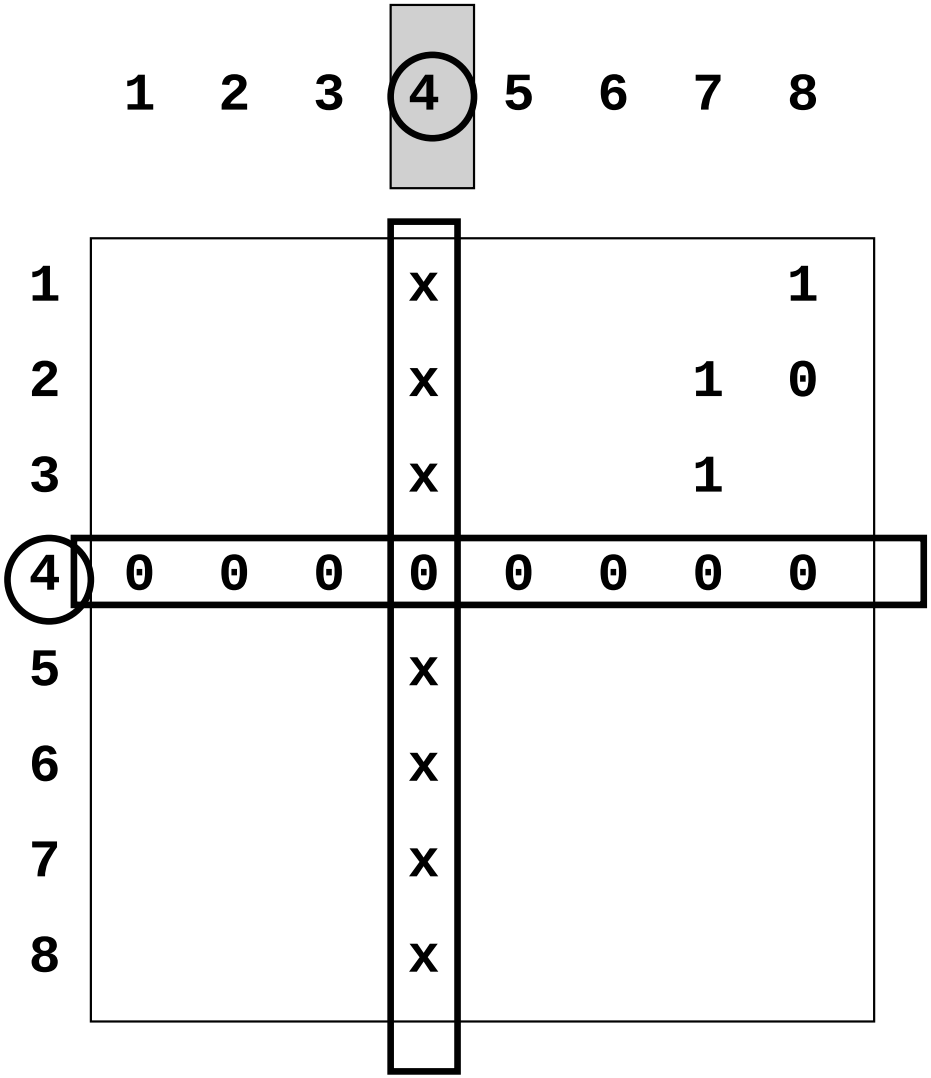
Decrease & Conquer



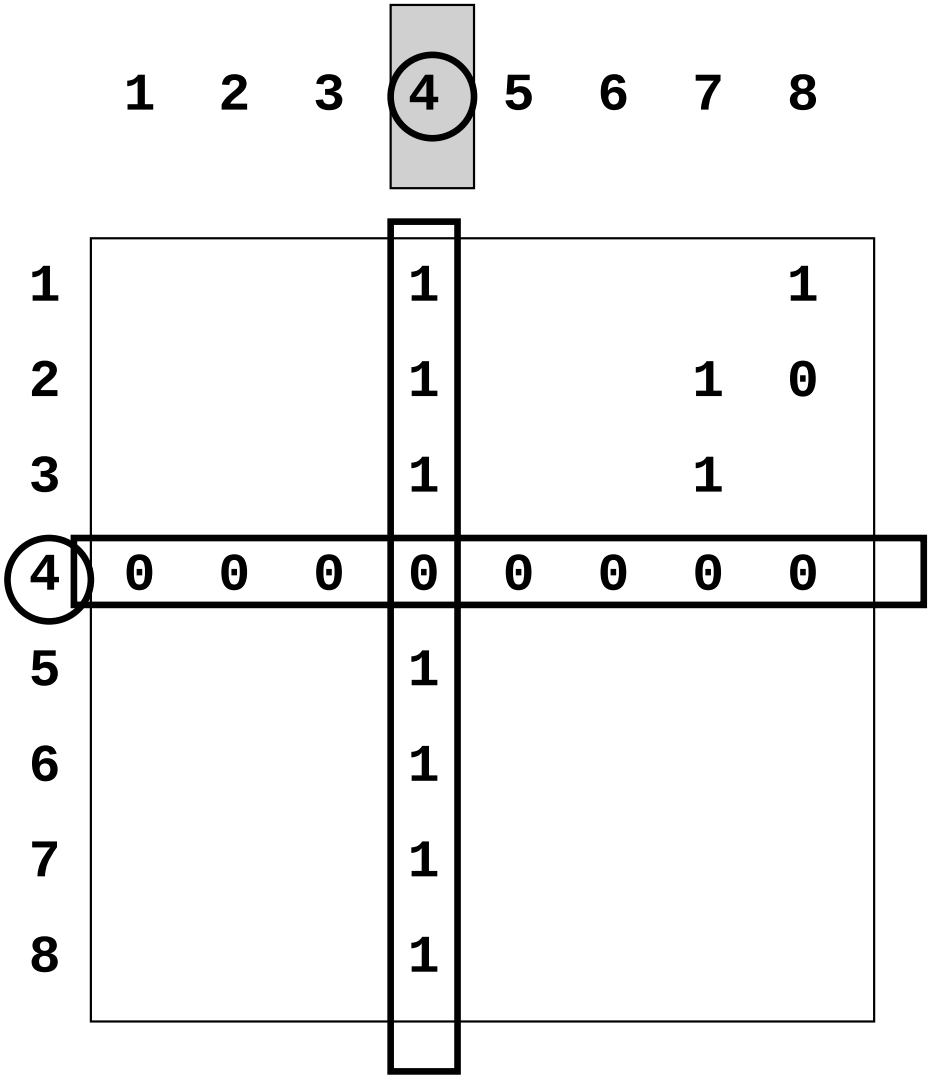
Decrease & Conquer



Decrease & Conquer



Decrease & Conquer



Decrease & Conquer

```
Celebrity_DQ( a[1..n, 1..n], p, q ) {

   if ( p == q ) return p

   if (a[p,q] == 1) {
     s = Celebrity_DQ( a, p+1, q )
     s = Celebrity_Check( a, s, p, p )
   } else {
     s = Celebrity_DQ( a, p, q-1 )
     s = Celebrity_Check( a, s, q, q )
   }

   return s
}
```

t(n-1)

Θ(1)

t(n-1)

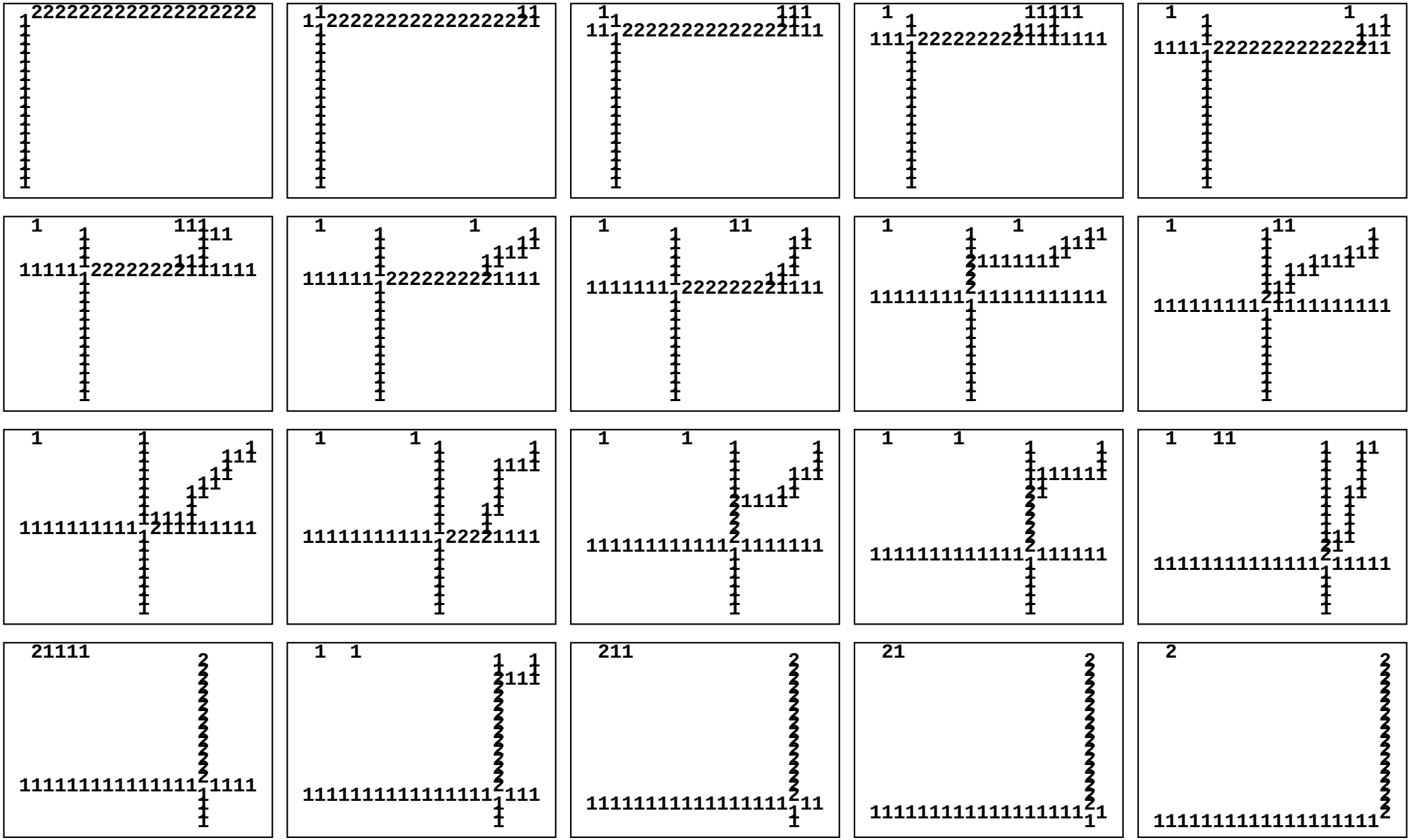
Θ(1)

$$t(n) = t(n-1) + \Theta(1)$$

$$t(n) = \Theta(n)$$

จำนวน probes = p(n) = p(n-1)+3 = 3(n-1)

การทดลอง : เมทริกซ์ขนาด 20x20



จำนวน probes เท่ากันหมด = 3(20-1)=57