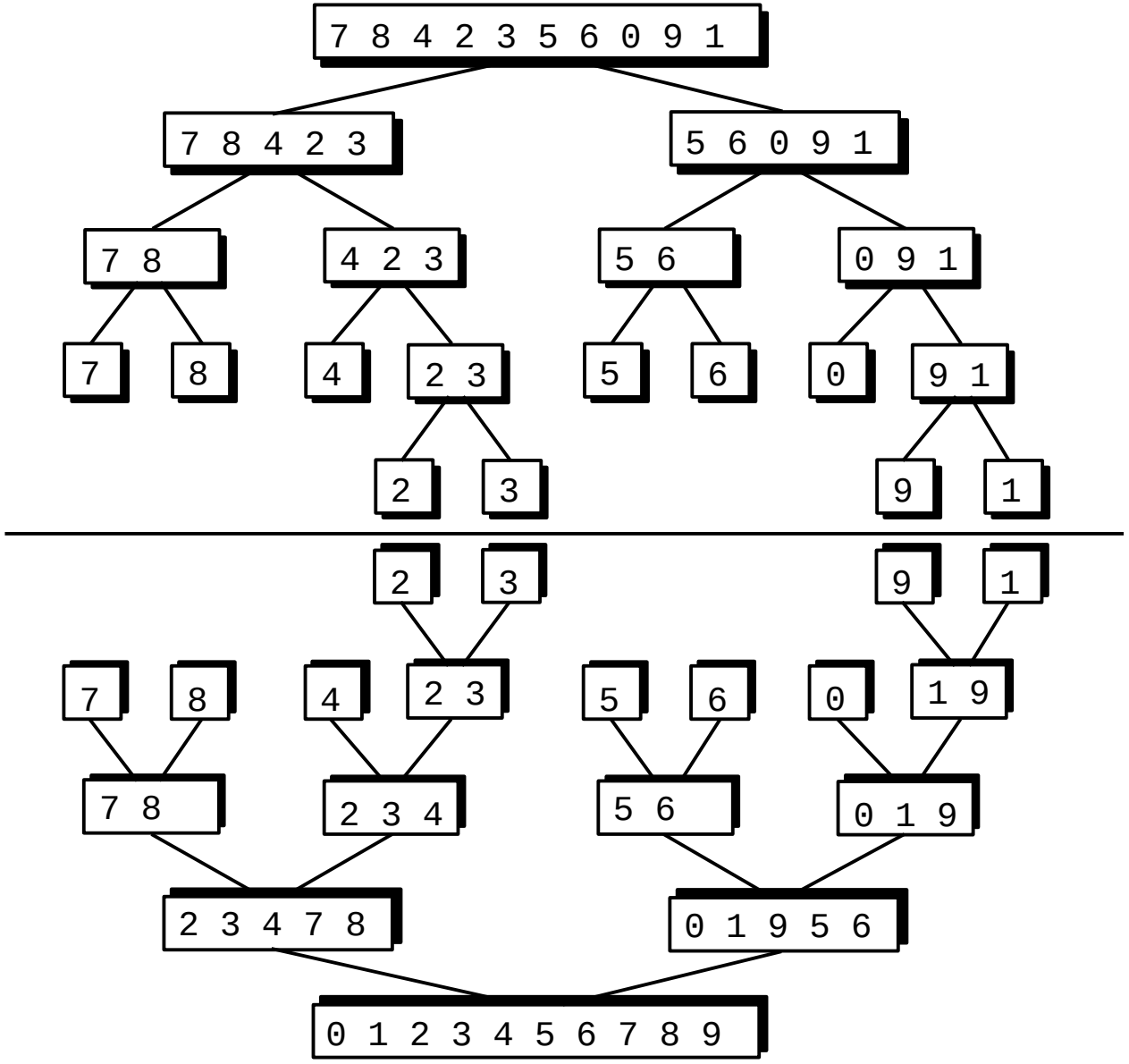
Divide and Conquer

```
Sort( list )
{
  if ( List has length greater than 1 ) {
    partition the list into lowlist and highlist;
    Sort( lowlist );
    Sort( highlist );
    list = Combine( lowlist, highlist );
  }
}
```

MergeSort

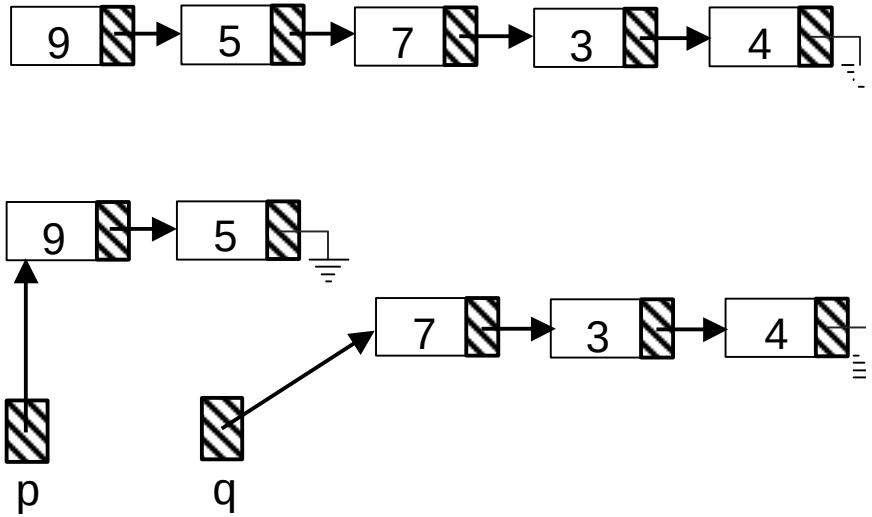
QuickSort

Mergesort

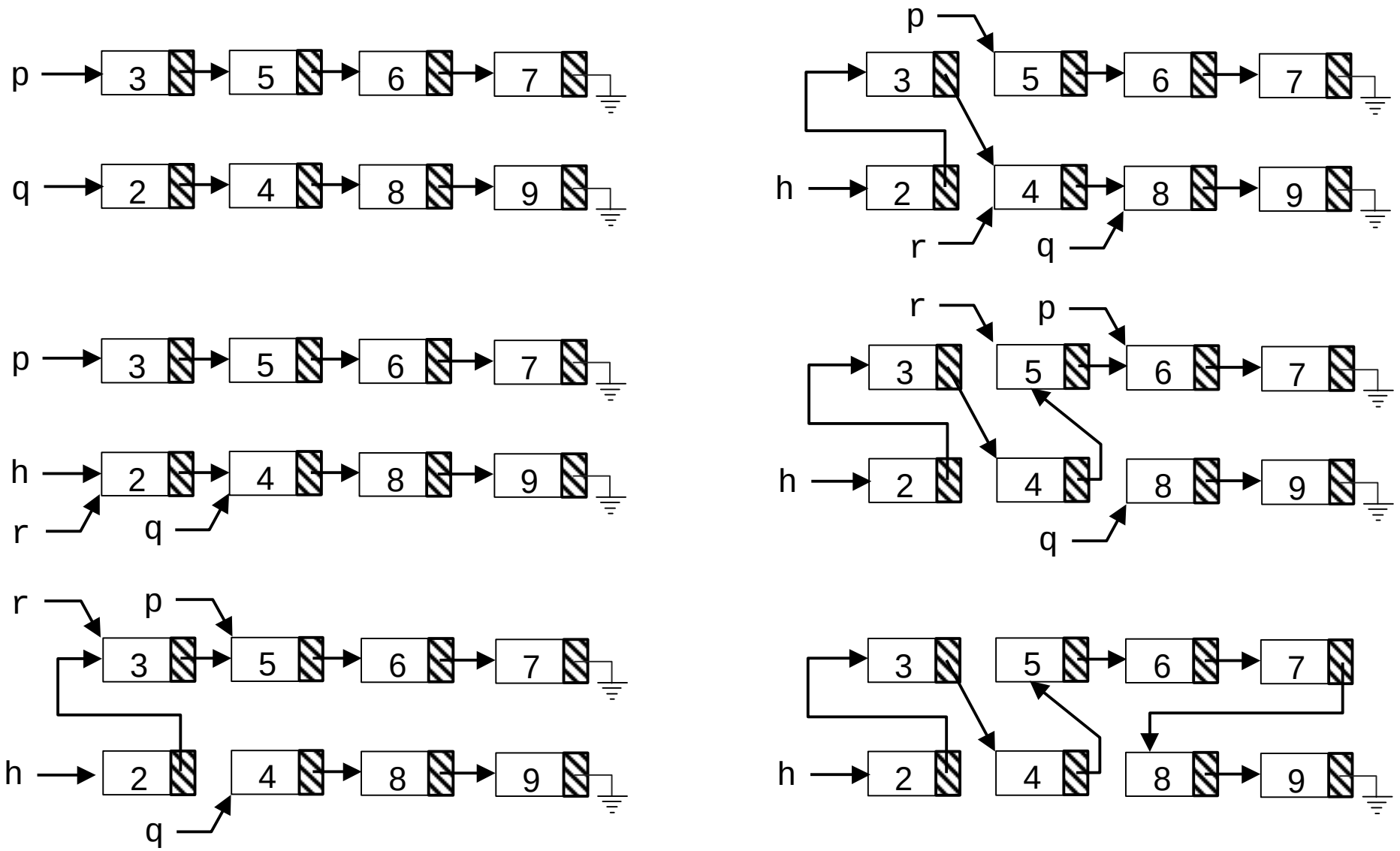


Mergesort : Dividing

```
ListType *Mergesort( ListType *p )
{
  ListType   *q, *head;
  head = p;
  if ( p && p->next ) {
    q = Divide( p );
    p = Mergesort( p );   q = Mergesort( q );
    head  = Merge( p, q );
  }
  return( head );
}
ListType *Divide( ListType *p )
{
  ListType      *q, *r;
  q = p; r = p->next->next;
  while ( r ) {
    q = q->next;   r = r->next;
    if ( r )  r = r->next;
  }
  r = q->next;   q->next = NULL;
```



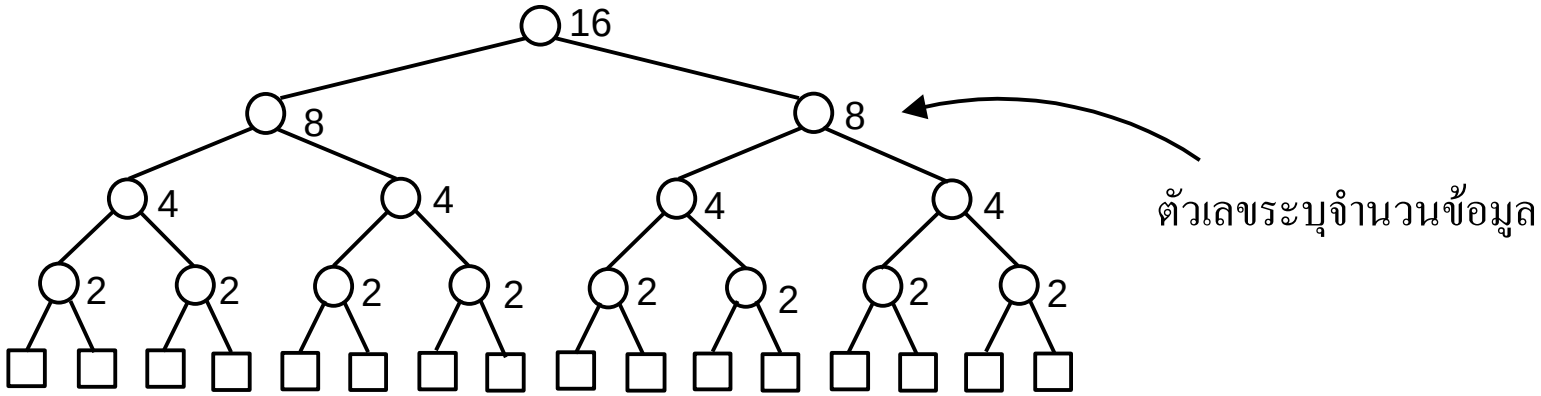
Mergesort : Merging



Merge ง่ายหรือไม่ถ้าเก็บข้อมูลใน array ?

Analysis of Mergesort

- การผสานสองรายการเข้าด้วยกัน โดยที่แต่ละรายการมี $n/2$ ข้อมูล
  จะใช้การเปรียบเทียบอย่างมากจำนวน n ครั้ง (ความจริงอย่างมาก n-1)
- ในแต่ละระดับ มีการเปรียบเทียบข้อมูล ระหว่างการผสานข้อมูลทั้งหมด ไม่มากกว่า n ครั้ง



- เนื่องจากต้นไม้มีความสูง $\lceil \log n \rceil$
- ดังนั้น ต้องทำการเปรียบเทียบข้อมูลทั้งหมดไม่มากกว่า $n \lceil \log n \rceil$ ครั้ง

Analysis of Mergesort

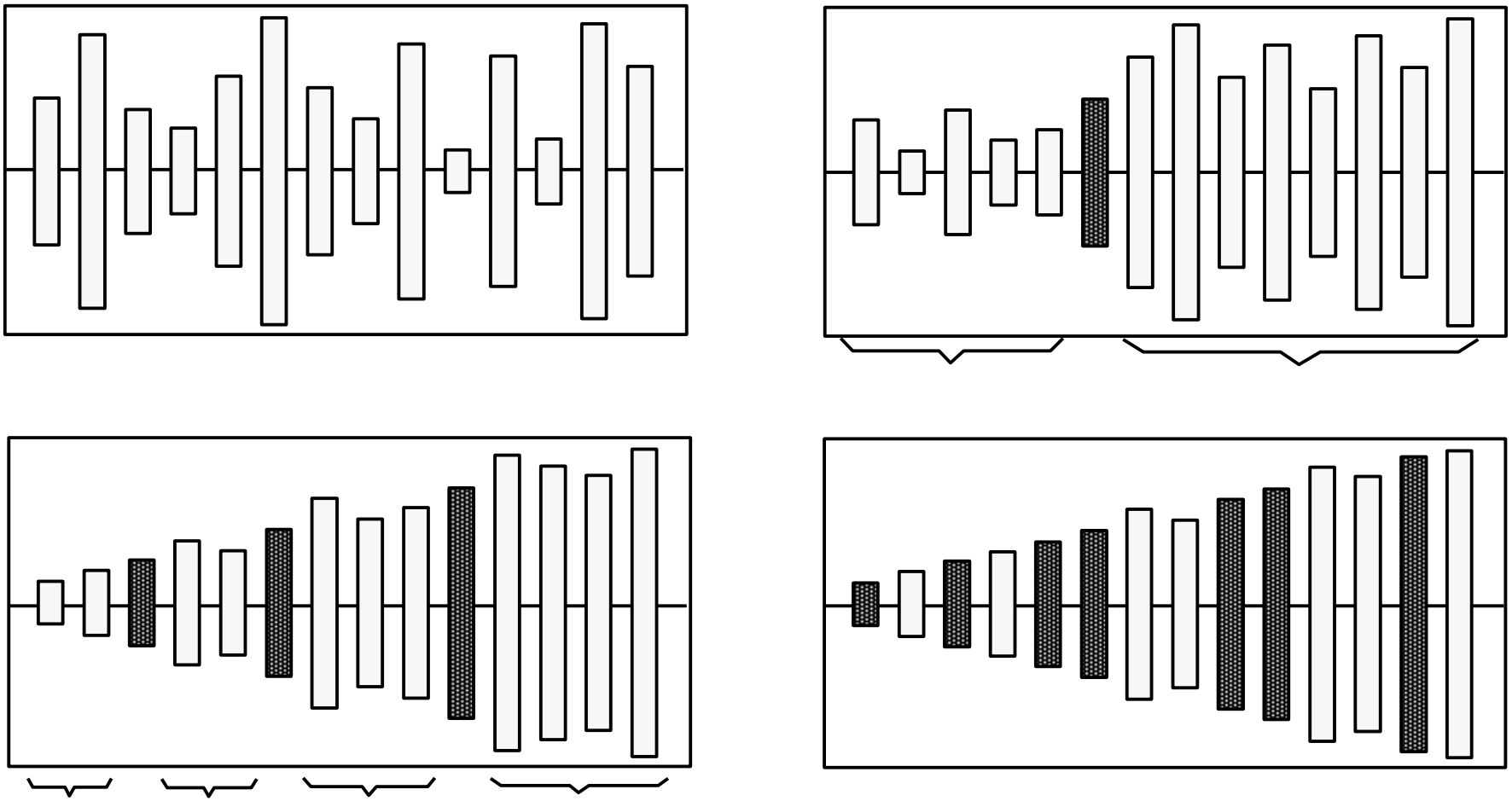
- ความจริงแล้ว การผสานสองรายการเข้าด้วยกัน โดยที่แต่ละรายการมี $n/2$ ข้อมูล จะใช้การเปรียบเทียบอย่างมากจำนวน n-1 ครั้ง
- ดังนั้นในระดับที่ k จะมีการเปรียบเทียบข้อมูล ระหว่างการผสานข้อมูลทั้งหมด $n - n/2^{k+1}$ เพราะฉะนั้นจำนวนการเปรียบเทียบของการเรียงลำดับทั้งหมด

$$= \sum_{k=0}^{\log_2 n - 1} \left(n - \frac{n}{2^{k+1}} \right)$$

$$= n\log_2 n - \left(\frac{n}{2} + \frac{n}{4} + \frac{n}{8} \ldots + 1 \right)$$

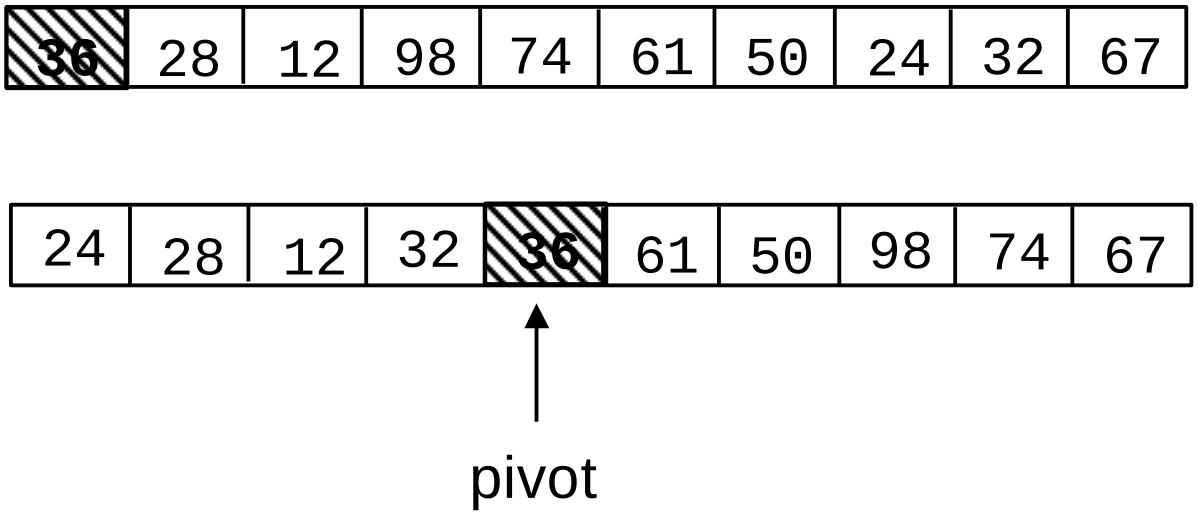
$$= n\log_2 n - n + 1$$

Quicksort



Quicksort

- เลือก pivot ได้ตามใจชอบ
- ฟังก์ชัน Partition ทำการแบ่งข้อมูลออกเป็น 2 ชุด
  ชุดทางซ้ายมีค่าของคีย์น้อยกว่า pivot และ
  ชุดทางขวามีค่าของคีย์มากกว่าหรือเท่ากับ pivot



Quicksort

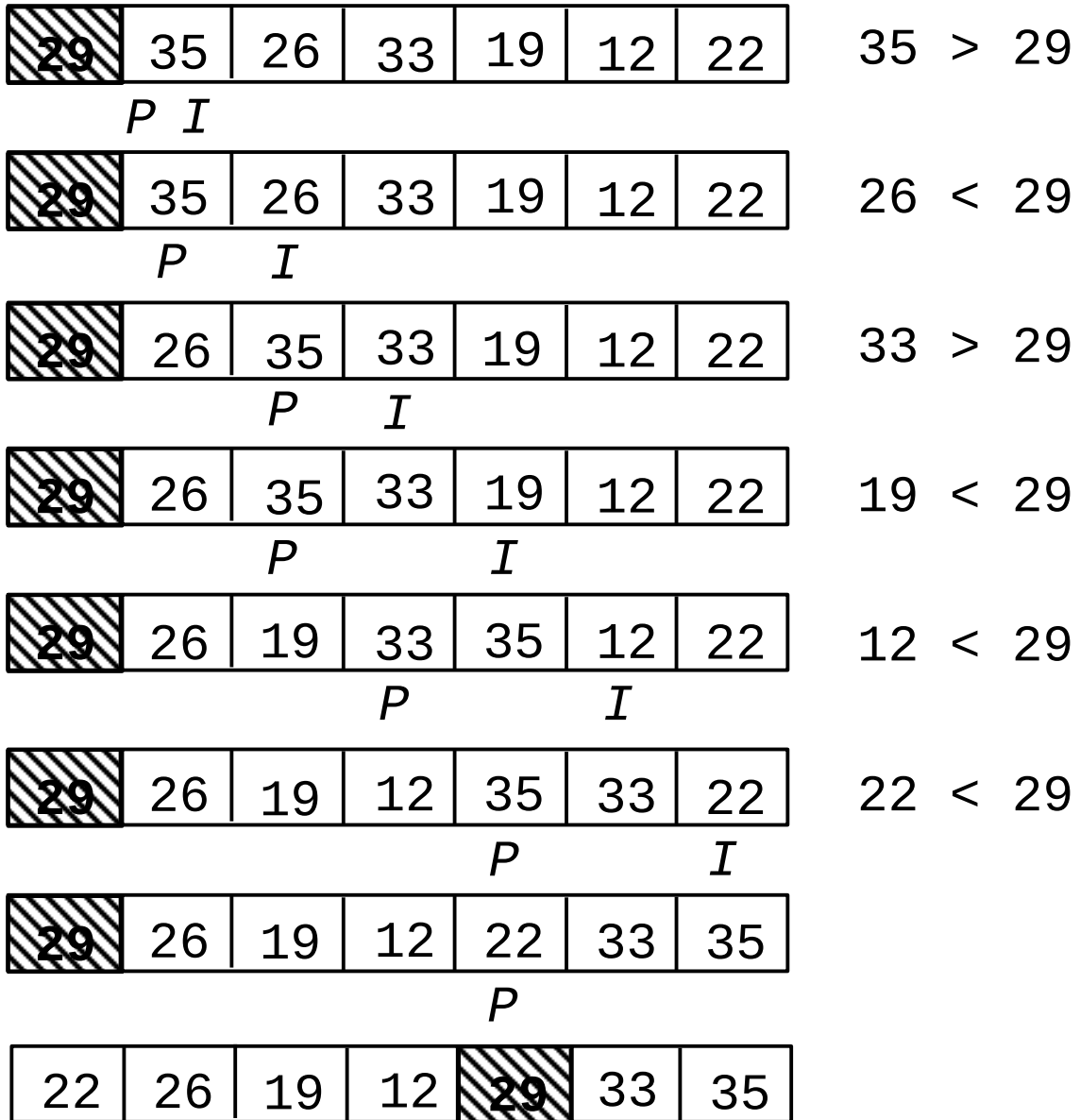
```
ListType  *Quicksort( ListType *pList );
{
   QSort( pList, 0, pList->count-1 );
   return( pList );
}

void  QSort( ListType *pList, int low, int high )
{
  if ( low < high ) {
    pivotLoc = Partition( pList, low, high );
    QSort( pList, low, pivotLoc-1 );
    QSort( pList, pivoltLoc+1, high );
  }
}
```

Quicksort

```
int  Partition( ListType *pList, int low, int high )
{
  pivolLoc = low;
  pivotKey = pList->entry[pivotLoc].key;
  for (i=low+1; i<=high; i++) {
    if ( LT( pList->entry[i].key, pivotkey ) )
      Swap( ++pivotLoc, i, pList );
  }
  Swap( low, pivotLoc, pList );
  return( pivotLoc );
}
```

Quicksort : Partition



Quicksort : Counting Swaps

- ให้ $S(n)$ คือจำนวนการสลับข้อมูล เมื่อมีข้อมูลทั้งหมด n ตัว
- เมื่อ $n \leq 1$ จะไม่มีการสลับข้อมูล
- ถ้าหลังการ partition แล้ว pivot อยู่ที่ตำแหน่ง p
  (นั่นคือได้แบ่งข้อมูลออกเป็น $(p\text{-}1)$ กับ $(n\text{-}p)$ ตัว) ดังนั้น

$$S(n) = (p\text{-}1+1) + S(p\text{-}1) + S(n\text{-}p)$$

- worst case เมื่อ $p = n$

$$S(n) = n + S(n\text{-}1)$$

$$S(n\text{-}1) = (n\text{-}1) + S(n\text{-}2)$$

$$\ldots$$

$$S(3) = 3 + S(2)$$

$$S(2) = 2$$

$$S(n) = n + (n\text{-}1) + \ldots + 2 = 0.5n^2 + 0.5n - 1$$

Worst case: $O(n^2)$

Quicksort : Counting Comparisions

- ให้ $C(n)$ คือจำนวนการเปรียบเทียบคีย์ เมื่อมีข้อมูลทั้งหมด n ตัว
- เมื่อ $n \leq 1$ จะไม่มีการเปรียบเทียบคีย์
- ถ้าหลังการ partition แล้ว pivot อยู่ที่ตำแหน่ง p
  (นั่นคือได้แบ่งข้อมูลออกเป็น $(p\text{-}1)$ กับ $(n\text{-}p)$ ตัว) ดังนั้น

$$C(n) = (n\text{-}1) + C(p\text{-}1) + C(n\text{-}p)$$

- worst case เมื่อ $p = n$

$$C(n) = (n\text{-}1) + C(n\text{-}1)$$

$$C(n\text{-}1) = (n\text{-}2) + C(n\text{-}2)$$

. . .

$$C(3) = 2 + C(2)$$

$$C(2) = 1$$

$$C(n) = (n\text{-}1) + (n\text{-}2) + \ldots + 1 = 0.5n^2 - 0.5n$$

Worst case: $O(n^2)$

Quicksort : Average # of Swaps

- ให้ $S(n)$ คือจำนวนการสลับข้อมูลเฉลี่ย เมื่อมีข้อมูลทั้งหมด n ตัว
- ให้ $S(n,p)$ คือจำนวนการสลับข้อมูลเฉลี่ย เมื่อ pivot อยู่ที่ตำแหน่ง p

$$S(n,p) = p + S(p-1) + S(n-p)$$

$$S(n) = \frac{1}{n}\sum_{p=1}^{n} S(n,p)$$

$$= \frac{1}{n}\left[\frac{n(n+1)}{2} + 2(S(0)+\ldots+S(n-1))\right]$$

$$= \frac{n+1}{2} + \frac{2}{n}\left[S(0)+\ldots+S(n-1)\right]$$

$$S(n-1) = \frac{n}{2} + \frac{2}{(n-1)}\left[S(0)+\ldots+S(n-2)\right]$$

Quicksort : Average # of Swaps

$$S(n,p) = p + S(p-1) + S(n-p)$$

$$S(n) = \frac{1}{n}\sum_{p=1}^{n} S(n,p)$$

$$= \frac{1}{n}\left[\frac{n(n+1)}{2} + 2\left(S(0)+\ldots+S(n-1)\right)\right]$$

$$= \frac{n+1}{2} + \frac{2}{n}\left(S(0)+\ldots+S(n-1)\right)$$

$$S(n-1) = \frac{n}{2} + \frac{2}{(n-1)}\left(S(0)+\ldots+S(n-2)\right)$$

Quicksort : Average # of Comparisons

- ให้ $C(n)$ คือจำนวนการสลับข้อมูลเฉลี่ย เมื่อมีข้อมูลทั้งหมด n ตัว
- ให้ $C(n,p)$ คือจำนวนการสลับข้อมูลเฉลี่ย เมื่อ pivot อยู่ที่ตำแหน่ง p

$$C(n,p) = p + C(p-1) + C(n-p)$$

$$C(n) = \frac{1}{n}\sum_{p=1}^{n} C(n,p)$$

$$= \frac{1}{n}\left[\frac{n(n+1)}{2} + 2(C(0)+\ldots+C(n-1))\right]$$

$$= \frac{n+1}{2} + \frac{2}{n}\left(C(0)+\ldots+C(n-1)\right)$$

$$C(n-1) = \frac{n}{2} + \frac{2}{(n-1)}\left(C(0)+\ldots+C(n-2)\right)$$