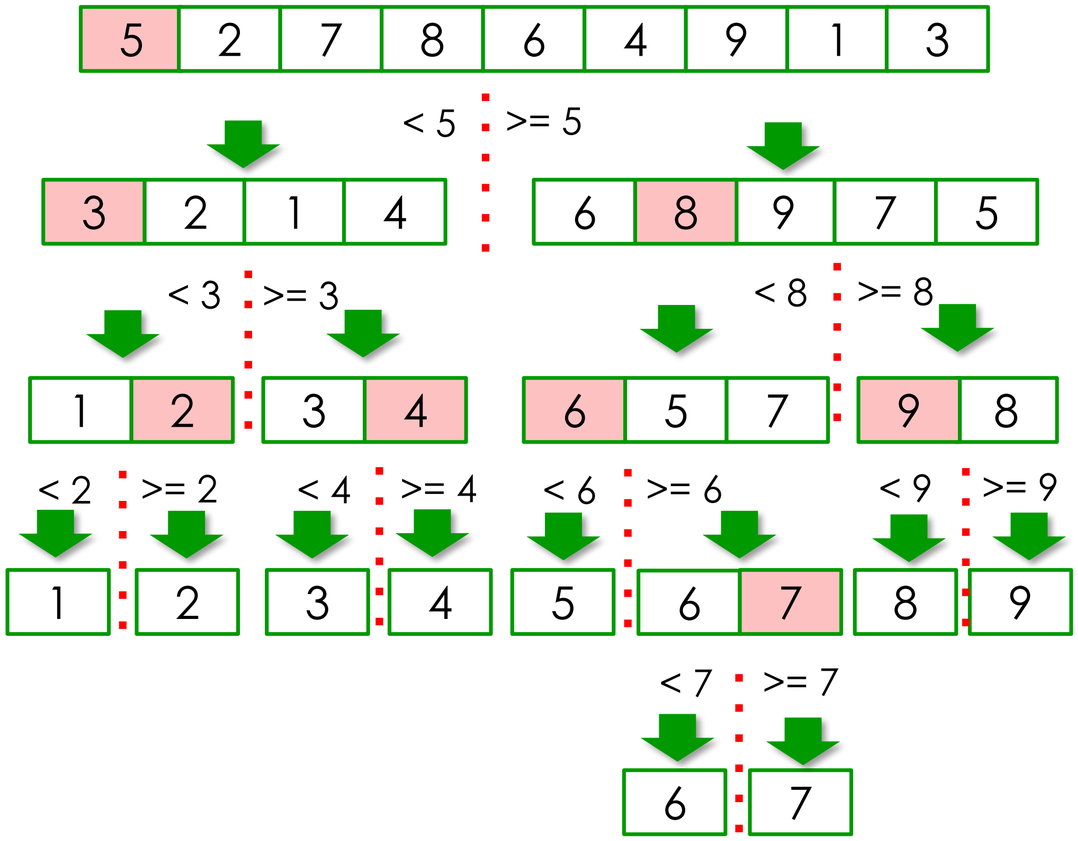
# （参考）
# 本来のクイックソート

# 本来のクイックソート

- 上で示したクイックソートでは、分割の度に新たなリストを用意していました
- 本来のクイックソートでは、新たなリストを用意せずに、リスト上の値の比較と交換だけで、ソートを行うことが可能です
  - ※その代わり、手順は若干複雑になります
- 一般に「ピボットより小さい」グループと「ピボット以上の」グループに分割する実装が多いようです

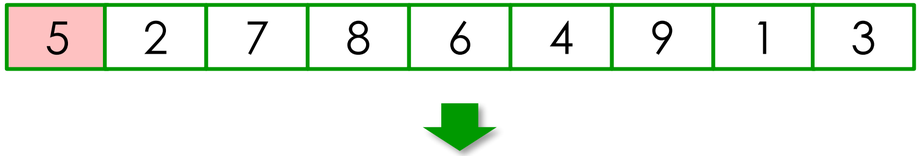
# クイックソートの本来の手順

- 全体の手順は、以下のようになります
  1. リストの要素が全て同じである場合は、終了します
  2. リストの中から「ピボット」を選択します
  3. リストを、ピボットより小さい要素のリストと、ピボット以上の要素のリストに分割します
  4. 小さいグループに対して、1からの手順を再帰的に適用します
  5. 大きいグループに対して、1からの手順を再帰的に適用します

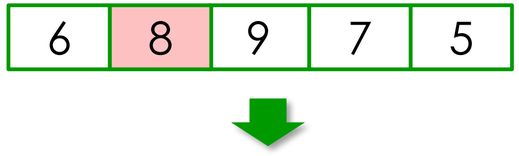


# クイックソート：ピボットの選択の手順

- ピボットとしては、最小ではない要素を選択する必要があります
- 最も簡単な方法としては、以下の手順が考えられます
  1. 先頭から要素を順に確認する
  2. 先頭よりも小さい要素が見つかれば、先頭要素をピボットとする
  3. 先頭よりも大きい要素が見るかれば、その要素をピボットとする



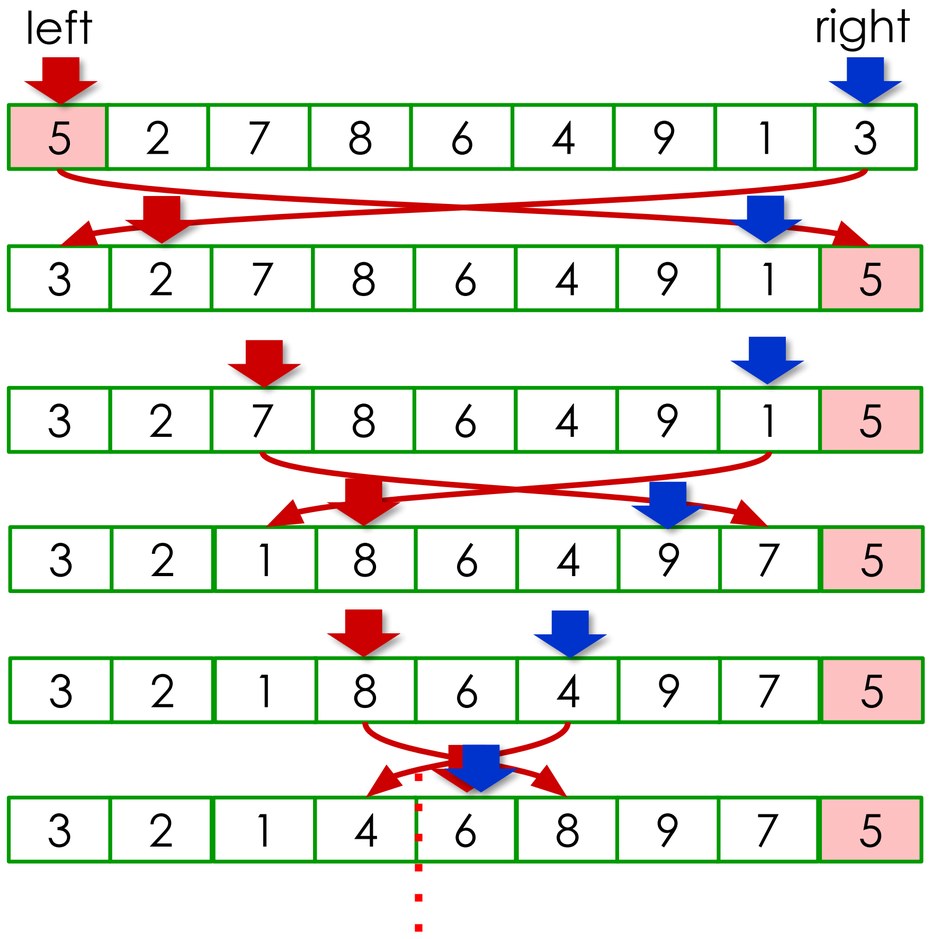

5がピボット（5 > 2）




8がピボット（8 > 6）

# クイックソート：分割の手順

- リストをピボットで分割する手順は、以下のようになります
  1. リストの先頭から、順に要素を確認していき、ピボットより小さいものは全て読み飛ばす（left）
  2. リストの末尾から、逆順に要素を確認していき、ピボット以上のものは全て読み飛ばす（right）
  3. leftとrightが一致したら、その直前で分割が完了しているので、終了する
  4. leftとrightの要素を入れ替え、1からの手順を繰り返す



# Pythonで記述すると…

```python
def qsort(lst):
    qsort_inner(lst, 0, len(lst) - 1)

def qsort_inner(lst, start, end):
    if start >= end:
        return
    pivot = find_pivot(lst, start, end)
    if pivot is None:
        return
    p = partition(lst, start, end, pivot)
    qsort_inner(lst, start, p - 1)
    qsort_inner(lst, p, end)
```

```python
def find_pivot(lst, start, end):
    for i in range(start + 1, end + 1):
        if lst[i] > lst[start]:
            return lst[i]
        if lst[i] < lst[start]:
            return lst[start]
    return None

def partition(lst, left, right, pivot):
    while True:
        while lst[left] < pivot:
            left += 1
        while (lst[right] >= pivot) and (right >= 0):
            right -= 1
        if left >= right:
            return left
        tmp = lst[left]
        lst[left] = lst[right]
        lst[right] = tmp
        left += 1
        right -= 1
```