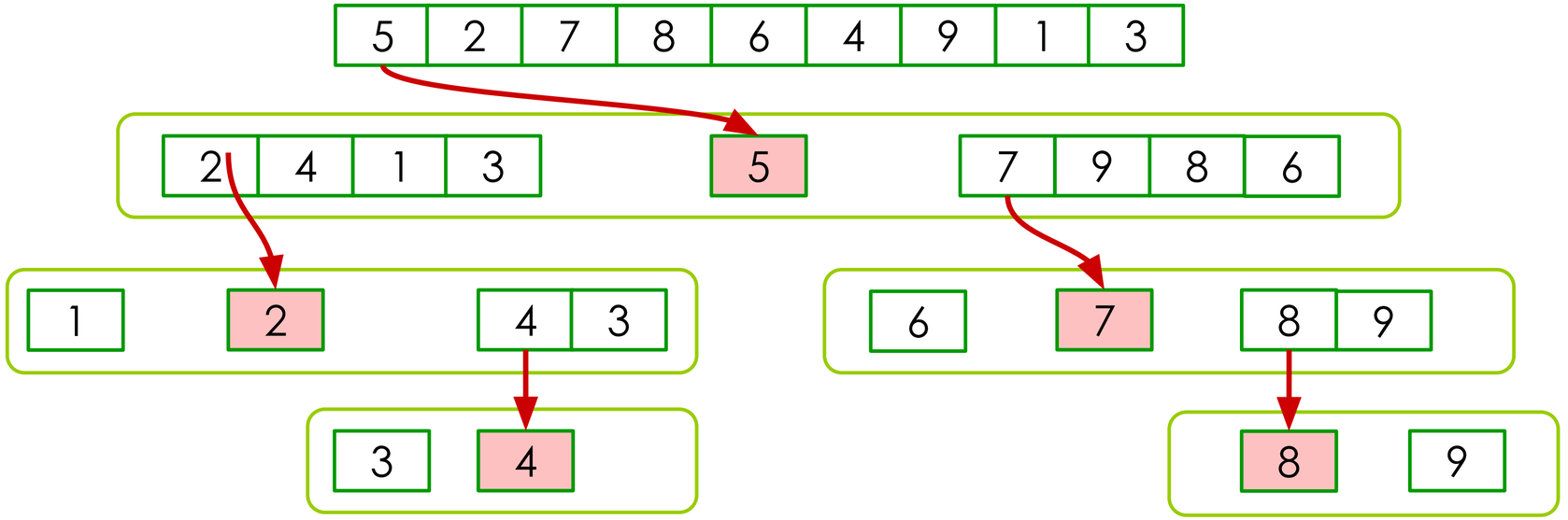
クイックソートを実装してみよう

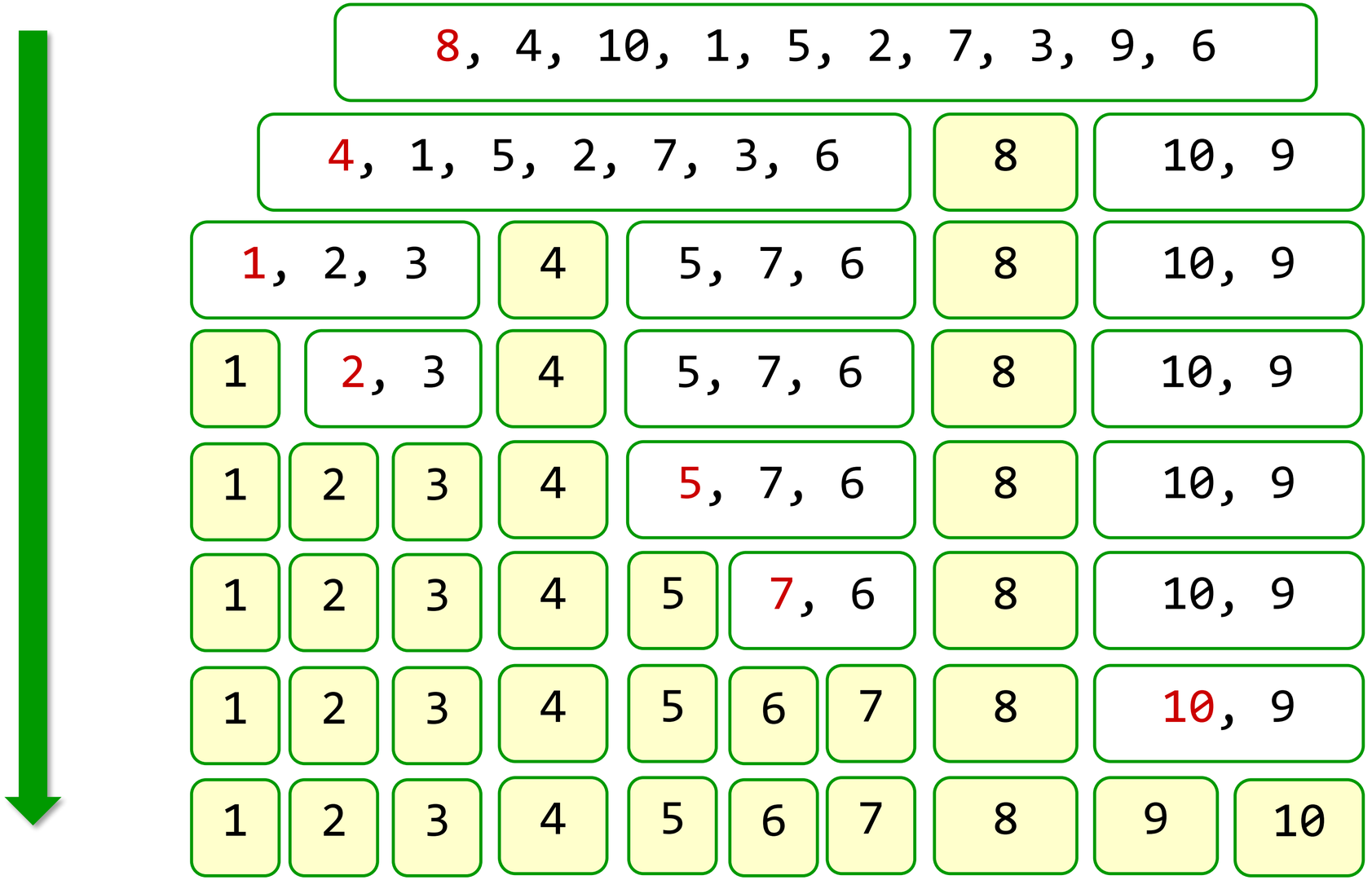
(再掲)クイックソート(簡易版)

- 以下の手順を再帰的に繰返すアルゴリズムです
  1. ソート対象から「ピボット(軸)」を選ぶ
  2. ソート対象をピボットより小さいグループと大きいグループに分割する
  3. 各グループを新たにソート対象として、再帰的にソートする



(再掲)クイックソート：具体例

※先頭要素を「ピボット」とした場合



（再掲）Python で定義してみると...

- 実は、クイックソートは、以下のように非常にシンプルに実装することができます

```
def simple_qsort(lst):
    if len(lst) <= 1:
        return lst
    pivot = lst[0]

    lst1 = [x for x in lst if x < pivot]
    lst2 = [x for x in lst if x == pivot]
    lst3 = [x for x in lst if x > pivot]

    return simple_qsort(lst1) + lst2 + simple_qsort(lst3)
```

長さが1以下のリストはソートする必要はありません

先頭の要素をピボットとする

ピボット以下、ピボット一致、ピボット以上の要素からなるリストを作成

再帰呼出しの結果を結合

```
simple_qsort([10, 2, 6, 1, 3, 4, 5, 8, 7, 9])
```

```
[1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
```

まだ実装できていない場合は
この演習時間内に実装しましょう！

クイックソートの計算回数

- 簡易的なクイックソートは再帰呼び出しによって実装されている
- 赤字のコードを追加して、再帰呼び出しの回数を調べてみよう

```
def simple_qsort(lst):
    global n
    n += 1
    if len(lst) <= 1:
        return lst
    pivot = lst[0]

    lst1 = [x for x in lst if x < pivot]
    lst2 = [x for x in lst if x == pivot]
    lst3 = [x for x in lst if x > pivot]

    return simple_qsort(lst1) + lst2 +
simple_qsort(lst3)
```

```
n = 0
simple_qsort([10, 2, 6, 1, 3, 4, 5, 8, 7, 9])
print(n)
```

簡易版クイックソートの再帰呼び出し回数

- 1回の分割で、だいたいリストの長さが半分になります
- リストの長さを n とすれば、分割の段数は、約 $\log_2 n$ になります
- n = 10 とすると $\log_2 10 = 3.3219$
- （平均的に）3段目まで分割が行われるので、再帰回数は
  - 1 + 2 + 4 + 8 = 15 回 程度