Quicksort

Setup

- Set recursion limit to `maxint`, $2^{31} - 1$
  - This is the highest value Python allows
- Import `time` to time executions

In [1]:
```
import sys
sys.setrecursionlimit(2**31-1)
import time
```

In [2]:
```
def quicksort(L,l,r):  # Sort L[l:r]
  if (r - l <= 1):
    return
  (pivot,lower,upper) = (L[l],l+1,l+1)
  for i in range(l+1,r):
    if L[i] > pivot:  # Extend upper segment
      upper = upper+1
    else:  # Exchange L[i] with start of upper segment
      (L[i], L[lower]) = (L[lower], L[i])
      # Shift both segments
      (lower,upper) = (lower+1,upper+1)
  # Move pivot between lower and upper
  (L[l],L[lower-1]) = (L[lower-1],L[l])
  lower = lower-1
  # Recursive calls
  quicksort(L,l,lower)
  quicksort(L,lower+1,upper)
  return
```

Small input to check correctness

In [3]:
```
qlist = [1,3,5,0,2,4,17,2,-5,6,4,3]
quicksort(qlist,4,8)
print(qlist)
```

```
[1, 3, 5, 0, 2, 2, 4, 17, -5, 6, 4, 3]
```

Quicksort performance

- Random input of size 10^6
- Sorted inputs of size 2×10^4

In [4]:
```
import random
random.seed(2023)
inputlists = {}
inputlists["random"] = [random.randrange(100000000) for i in range(1000000)]
inputlists["ascending"] = [i for i in range(15000)]
inputlists["descending"] = [i for i in range (14999,-1,-1)]
for k in inputlists.keys():
    tmplist = inputlists[k][:]
    starttime = time.perf_counter()
    quicksort(tmplist,0,len(tmplist))
    elapsed = time.perf_counter() - starttime
    print(k,elapsed)
```

```
random 2.440271896000013
ascending 4.733194645999902
descending 6.899413855000148
```

Randomized quicksort

- Choose the pivot position uniformly at random between `l` and `r-1`
- Swap pivot to `L[l]` and proceed as usual

In [5]:
```
import random
def quicksortrand(L,l,r):  # Sort L[l:r]
  if (r - l <= 1):
    return(L)

  # Choose a random postion between l and r-l as pivot, swap with L[l]
  randpivot = random.randrange(r-l)
  (L[l],L[l+randpivot]) = (L[l+randpivot],L[l])

  # Rest is same as before
  (pivot,lower,upper) = (L[l],l+1,l+1)
  for i in range(l+1,r):
    if L[i] > pivot:  # Extend upper segment
      upper = upper+1
    else:  # Exchange L[i] with start of upper segment
      (L[i], L[lower]) = (L[lower], L[i])
      # Shift both segments
      (lower,upper) = (lower+1,upper+1)
  # Move pivot between lower and upper
  (L[l],L[lower-1]) = (L[lower-1],L[l])
  lower = lower-1
  # Recursive calls
  quicksortrand(L,l,lower)
  quicksortrand(L,lower+1,upper)
  return(L)
```

In [6]:
```
import random
random.seed(2023)
inputlists = {}
inputlists["random"] = [random.randrange(100000000) for i in range(1000000)]
inputlists["ascending"] = [i for i in range(15000)]
inputlists["descending"] = [i for i in range (14999,-1,-1)]
inputlists["ascendingbig"] = [i for i in range(1000000)]
inputlists["descendingbig"] = [i for i in range (999999,-1,-1)]
for k in inputlists.keys():
    tmplist = inputlists[k][:]
    starttime = time.perf_counter()
    quicksortrand(tmplist,0,len(tmplist))
    elapsed = time.perf_counter() - starttime
    print(k,elapsed)
```

```
random 2.6367314690000967
ascending 0.033154242999899
descending 0.02195621900000333
ascendingbig 2.069635259000279
descendingbig 1.9497650219996103
```