In [1]:
```
def quicksort(array):
    print(f"quicksorting {array}")
    if len(array) <= 1:
        return array
    pivot = array[0]
    first_half = []
    second_half = []
    for i in range(1, len(array)):
        if array[i] < pivot:
            first_half.append(array[i])
        else:
            second_half.append(array[i])
    print(f"split: {first_half} {pivot} {second_half}")
    first_half_sorted = quicksort(first_half)
    second_half_sorted = quicksort(second_half)
    return first_half_sorted + [pivot] + second_half_sorted
```

In [2]:
```
import random
```

In [3]:
```
rl = [random.randrange(100) for _ in range(20)]
```

In [5]:
```
rl
```

Out[5]:
```
[77,
 17,
 28,
 10,
 71,
 90,
 77,
 36,
 83,
 86,
 69,
 23,
 91,
 17,
 84,
 38,
 47,
 45,
 78,
 63]
```

In [8]:
```
list(enumerate(quicksort(rl)))
```

```
quicksorting [77, 17, 28, 10, 71, 90, 77, 36, 83, 86, 69, 23, 91, 17
, 84, 38, 47, 45, 78, 63]
split: [17, 28, 10, 71, 36, 69, 23, 17, 38, 47, 45, 63] 77 [90, 77,
83, 86, 91, 84, 78]
quicksorting [17, 28, 10, 71, 36, 69, 23, 17, 38, 47, 45, 63]
split: [10] 17 [28, 71, 36, 69, 23, 17, 38, 47, 45, 63]
quicksorting [10]
quicksorting [28, 71, 36, 69, 23, 17, 38, 47, 45, 63]
split: [23, 17] 28 [71, 36, 69, 38, 47, 45, 63]
quicksorting [23, 17]
split: [17] 23 []
quicksorting [17]
quicksorting []
quicksorting [71, 36, 69, 38, 47, 45, 63]
split: [36, 69, 38, 47, 45, 63] 71 []
quicksorting [36, 69, 38, 47, 45, 63]
split: [] 36 [69, 38, 47, 45, 63]
quicksorting []
quicksorting [69, 38, 47, 45, 63]
split: [38, 47, 45, 63] 69 []
quicksorting [38, 47, 45, 63]
split: [] 38 [47, 45, 63]
quicksorting []
quicksorting [47, 45, 63]
split: [45] 47 [63]
quicksorting [45]
quicksorting [63]
quicksorting []
quicksorting []
quicksorting [90, 77, 83, 86, 91, 84, 78]
split: [77, 83, 86, 84, 78] 90 [91]
quicksorting [77, 83, 86, 84, 78]
split: [] 77 [83, 86, 84, 78]
quicksorting []
quicksorting [83, 86, 84, 78]
split: [78] 83 [86, 84]
quicksorting [78]
quicksorting [86, 84]
split: [84] 86 []
quicksorting [84]
quicksorting []
quicksorting [91]
```

Out[8]:
```
[(0, 10),
 (1, 17),
 (2, 17),
 (3, 23),
 (4, 28),
 (5, 36),
 (6, 38),
 (7, 45),
 (8, 47),
 (9, 63),
 (10, 69),
 (11, 71),
 (12, 77),
 (13, 77),
 (14, 78),
 (15, 83),
 (16, 84),
 (17, 86),
 (18, 90),
 (19, 91)]
```

In [24]:
```
l = [(3/4)**i for i in range(100)]
```

In [27]:
```
[sum(l[0:i]) for i in range(30)]
```

```
Out[27]: [0,
          1.0,
          1.75,
          2.3125,
          2.734375,
          3.05078125,
          3.2880859375,
          3.466064453125,
          3.59954833984375,
          3.6996612548828125,
          3.7747459411621094,
          3.831059455871582,
          3.8732945919036865,
          3.904970943927765,
          3.9287282079458237,
          3.9465461559593678,
          3.959909616969526,
          3.9699322127271444,
          3.9774491595453583,
          3.9830868696590187,
          3.987315152244264,
          3.990486364183198,
          3.9928647731373985,
          3.994648579853049,
          3.9959864348897867,
          3.99698982616734,
          3.997742369625505,
          3.9983067772191285,
          3.998730082914346,
          3.9990475621857597]
```

```
In [11]: 374*225
```

```
Out[11]: 84150
```

```
In [12]: 374374374374374*225225225225
```

```
Out[12]: 84318552786936852618384150
```

```
In [13]: 84318552786936852618384150*84318552786936852618384150
```

```
Out[13]: 7109618344083456475708730224430057037790556971222500
```

In [1]:
```
# number: array of "digits"
# stored in little endian order for convenience
def toarray(n):
    if n < 10:
        return [n]
    else:
        return [n%10] + toarray(n//10)
```

In [2]:
```
toarray(374)
```

Out[2]: [4, 7, 3]

In [4]:
```
toarray(225)
```

Out[4]: [5, 2, 2]

In [30]:
```
def lattice_multiply(x, y):
    """ `x` and `y` are arrays of digits (base 10) of two numbers """
    # note that this was cleaned up after lecture
    result = []
    product = 0
    for i in range(len(x) + len(y)):
        for j in range(i+1):
            if j < len(x) and i-j < len(y):
                product += x[j] * y[i-j]
                print(product,i,i-j)
        result.append(product % 10)
        product = product // 10 # carry to next digit
        print(f"carry is {product}")
    return result
```

In [31]:
```
lattice_multiply([4,7,3], [5,2,2])
```

```
20 0 0
carry is 2
10 1 1
45 1 0
carry is 4
12 2 2
26 2 1
41 2 0
carry is 4
18 3 2
24 3 1
carry is 2
8 4 2
carry is 0
carry is 0
```

Out[31]:
```
[0, 5, 1, 4, 8, 0]
```

In [32]:
```
374*225
```

Out[32]:
```
84150
```