이산수학 PBL

- Sorting과 시간 복잡도 -

팀장님, 잠시만요

- 공대영
- 김용호
- 양원준
- 임승호
- 전지웅

현재 상황

- Bubble sort로 회사들의 Ticker를 sorting한 후 매매에 활용

- 우리 증권사가 sorting하는 주식의 수는 10개 이하

EX)

	A	B
1	**Symbol**	**Name**
2	005930	삼성전자
3	AAPL	애플(Apple)
4	TSLA	테슬라(Tesla)
5	005935	삼성전자우
6	005380	현대차
7	000660	SK하이닉스
8	KO	코카콜라
9	NFLX	넷플릭스
10	005385	현대차우

- 10개의 주식을 bubble sort로 sorting 하면

```r
bubble_sort <- function(stock_tickers) {
  n <- length(stock_tickers)

  for (i in 1:(n - 1)) {
    for (j in 1:(n - i)) {
      if (stock_tickers[j] > stock_tickers[j + 1]) {
        temp <- stock_tickers[j]
        stock_tickers[j] <- stock_tickers[j + 1]
        stock_tickers[j + 1] <- temp
      }
    }
  }

  return(stock_tickers)
}

# 주어진 주식 ticker 벡터를 만듭니다.
stock_tickers <- c("005930", "AAPL", "TSLA", "005935", "005380", "000660", "KO", "NFLX", "005385")

# Bubble sort를 사용하여 주식 ticker를 정렬합니다.
sorted_stock_tickers <- bubble_sort(stock_tickers)

# 정렬된 주식 ticker를 출력합니다.
cat("Bubble Sort로 정렬된 주식 Ticker:\n", sorted_stock_tickers, "\n")
```

```
Bubble Sort로 정렬된 주식 Ticker:
 000660 005380 005385 005930 005935 AAPL KO NFLX TSLA
```

변경점

- 우리가 sorting하는 주식의 개수가 **수백**, **수천개**로 증가하였다.

Merge sort

◆ 목차

1. Bubble sort 자체가 시간 복잡도가 높다.

2. 국내외 주식을 구분하지 않고 정렬하여 시간 복잡도가 더욱 증가하였다.

국내와 미국의 주식을 각각 구분하여 최적화된 sorting 방법을 찾았다.

초기 설정

In [67]:

```
import pandas as pd
import FinanceDataReader as fdr
import time

kospi = fdr.StockListing('KOSPI')
kosdaq = fdr.StockListing('KOSDAQ')
nasdaq = fdr.StockListing('NASDAQ')
# code = kospi['Code'].tolist()
# code1 = nasdaq['Symbol'].tolist()

df = pd.concat([kospi, kosdaq])
df1 = df.copy()
df1 = df1.drop_duplicates('Code')

K_Stock = df1['Code'].tolist()
US_Stock = nasdaq['Symbol'].tolist()

```

Merge Sort - 정렬 방식 -

주어진 숫자들을 더 작은 숫자들로 분할

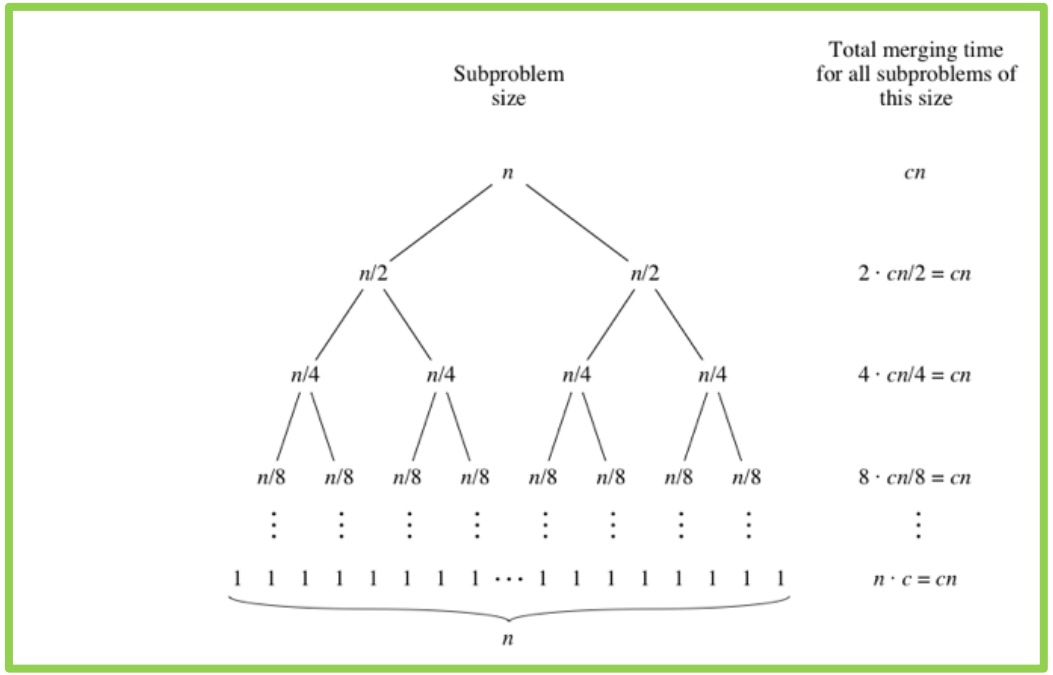
분할된 숫자들을 정렬한 뒤 합침

Merge Sort - pseudo code-

```
mergeSort(A[], p, r) ▷ A[p...r]을 정렬한다
{
    if (p < r) then {
        q ← (p+q)/2; ----------------- ① ▷ p, q의 중간 지점 계산
        mergeSort(A, p, q); ---------- ② ▷ 전반부 정렬
        mergeSort(A, q+1, r); -------- ③ ▷ 후반부 정렬
        merge(A, p, q, r);------------ ④ ▷ 합병
    }
}

merge(A[], p, q, r)
{
    정렬되어 있는 두 배열 A[p...q]와 A[q+1...r]을 합하여
    정렬된 하나의 배열 A[p...r]을 만든다.
}
```

Merge Sort - 시간 복잡도-



합병 단계의 수 × 각 단계에서의 비교 수

logN × N ->O(NlogN)

Merge Sort - 시간 복잡도(재귀함수)-

$$T(n)=\begin{cases}0 & (n=1)\\ T([n/2]+T(n/2)+n & (otherwise)\end{cases}$$

$2(2T(n/4)+n/2)+n \rightarrow 4T(n/4)+2n$

$4(2T(n/8)+n/4)+2n \rightarrow 8T(n/8)+3n$

$\Downarrow$

$n \times T(1)+ \log n \times n \rightarrow O(N\log N)$

Merge Sort (국내주식)

In [69]:

```python
def merge_sort(arr):
    if len(arr) <= 1:
        return arr

    mid = len(arr) // 2
    left = merge_sort(arr[:mid])
    right = merge_sort(arr[mid:])

    return merge(left, right)

def merge(left, right):
    result = []
    i = j = 0

    while i < len(left) and j < len(right):
        if left[i] <= right[j]:
            result.append(left[i])
            i += 1
        else:
            result.append(right[j])
            j += 1

    result.extend(left[i:])
    result.extend(right[j:])
    return result

start_time = time.time()
sorted_data = merge_sort(K_Stock)
end_time = time.time()

print(end_time - start_time)
```

```
0.013961076736450195
```

걸린 시간:

0.013961076736450195 초

Merge Sort (미국주식)

In [71]:

```
def merge_sort(arr):
    if len(arr) <= 1:
        return arr

    mid = len(arr) // 2
    left_half = arr[:mid]
    right_half = arr[mid:]

    left_half = merge_sort(left_half)
    right_half = merge_sort(right_half)

    return merge(left_half, right_half)

def merge(left, right):
    merged = []
    left_index, right_index = 0, 0

    while left_index < len(left) and right_index < len(right):
        if left[left_index] < right[right_index]:
            merged.append(left[left_index])
            left_index += 1
        else:
            merged.append(right[right_index])
            right_index += 1

    merged += left[left_index:]
    merged += right[right_index:]

    return merged

start_time = time.time()
data = merge_sort(US_Stock)
end_time = time.time()

print(end_time - start_time)
```

```
0.019945144653320312
```

걸린 시간:

0.019945144653320312 초

Quick sort

◆ 목차

Quick Sort (정렬 방법)

피벗을 설정

피벗을 기준으로 두개의 영역으로 분할

피벗을 기준으로 큰 수와 작은 수를 교환

Quick Sort (시간 복잡도)

피벗을 설정

↓

피벗과 n - 1개의 숫자들을 차례대로 비교

↓

비교연산을 총 n-1 번 수행 → O(n)

Quick Sort (Blanced partitioning)

피벗이 중간에 위치

피벗을 기준으로 좌우 양변이 계속 이등분

시간 복잡도: $T(n) \leq 2T(n/2) + O(n) = O(n\log n)$

Binary search와 유사함.

Quick Sort (Unblanced partitioning)

n개의 ticker가 계속 나뉘어짐

매번 1개씩 분할 → 전체 연산 횟수는 n번

매번 O(n)걸림 & n번 쪼개짐 → n * O(n) = O(n^2)

시간 복잡도: T(n) = T(n-1) + O(n) = O(n^2).

Quick Sort (Average complexity)

$$
\begin{aligned}
E[X] &= \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} \frac{2}{j-i+1} \\
&= \sum_{i=1}^{n-1} \sum_{k=1}^{n-i} \frac{2}{k+1} \\
&< \sum_{i=1}^{n-1} \sum_{k=1}^{n} \frac{2}{k} \\
&= \sum_{i=1}^{n-1} O(\lg n) \\
&= O(n \lg n)
\end{aligned}
$$

Quick Sort (국내주식)

In [68]:

```
1  def quick_sort(arr):
2      if len(arr) <= 1:
3          return arr
4      else:
5          pivot = arr[0]
6          less = [x for x in arr[1:] if x <= pivot]
7          greater = [x for x in arr[1:] if x > pivot]
8          return quick_sort(less) + [pivot] + quick_sort(greater)
9
10 start_time = time.time()
11 sorted_data = quick_sort(K_Stock)
12 end_time = time.time()
13
14 print(end_time - start_time)
```

```
0.008976459503173828
```

걸린 시간:

0.008976459503173828 초

Quick Sort (미국주식)

In [70]:

```
1  def quick_sort(arr):
2      if len(arr) <= 1:
3          return arr
4      else:
5          pivot = arr[0]
6          less = [x for x in arr[1:] if x <= pivot]
7          greater = [x for x in arr[1:] if x > pivot]
8          return quick_sort(less) + [pivot] + quick_sort(greater)
9
10 start_time = time.time()
11 a = quick_sort(US_Stock)
12 end_time = time.time()
13
14 print(end_time - start_time)
```

```
0.011967420578002930
```

걸린 시간:

0.011967420578002930 초

국내주식을 Sorting 하는데 걸린 시간:

Bubble Sort: 0.62812초

vs

Merge Sort: 0.01396 초

vs

Quick Sort: 0.00897 초

미국주식을 Sorting 하는데 걸린 시간:

Bubble Sort: 1.73347 초

vs

Merge Sort: 0.01994 초

vs

Quick Sort: 0.01196 초

Quick Sort

국내 주식, 미국 주식 모두

Quick sort가 빠르다!!!