알고리즘 분석 과제1

(1)

In [10]:
```
from time import time
```

- 알고리즘1

In [13]:
```
def fun1(n):
    if n==0 or n==1:
        return 1
    else:
        sum = 0
        for i in range(n):
            sum += fun(i)
        return sum

for i in range(24, 29):
    stime = time()
    fun1(i)
    print(f'{i:2d} {time()-stime:10.5f}')
```

```
24    2.18951
25    4.35539
26    8.75388
27   17.84193
28   35.34316
```

- 알고리즘2

In [15]:
```
def fun2(n):
    arr = []
    arr.append(1)
    arr.append(1)

    i = 0
    result = 0
    while i < n:
        result = sum(arr)
        arr.append(result)
        i += 1

    return arr[n]

for i in range(5):
    num = pow(2, i) * 1600
    stime = time()
    fun2(num)
    print(f'{num:5d} {time()-stime:10.5f}')
```

```
 1600    0.06628
 3200    0.25484
 6400    2.11720
12800   16.96964
25600  129.59143
```

알고리즘1		알고리즘2	
n	수행시간(초)	n	수행시간(초)

알고리즘1		알고리즘2	
24	2.18951	1,600	0.05735
25	4.35539	3,200	0.25142
26	8.75388	6,400	2.19987
27	17.84193	12,800	16.96964
28	35.34316	25,600	128.71195

(2)

(1)의 표 결과를 보면 대략적으로 T1(n+1) = T1(n)*2 그리고
T2(2n) = T2(n)*7 으로 나타낼 수 있다.

(3)

- 단위연산: 덧셈
- 입력크기: 배열의 크기 n
- 모든경우 분석: 배열의 크기에 따라서 while문 루프가 n번 반복된다. 각 루프마다 덧셈이 n-1번 수행된다.(sum함수) 따라서 덧셈이 수행되는 횟수 T2(n) = n(n-1) 이다.

(4)

해당 문제를 해결하기 위해 알고리즘1의 시간 복잡도도 구해보았다.
윗 표와는 조금 다른 결과가 나온것같지만 어쨋든 알고리즘1 의 시간복잡도 T1(n)은 2^n 이고 알고리즘2의 시간복잡도 T2(n)은 n(n-1)이다. 강의자료에 '시간복잡도별 실행시간 비교'를 참고해보면 알고리즘1, 알고리즘2를 10분간 수행할때 해결할 수 있는 문제의 크기 n1, n2는 각각 39, 10^6 이다.

- 알고리즘1의 시간복잡도는 T1(n) = T1(n-1)+ ... T1(1) 이다. 단위연산은 호출 횟수.

(5)

n이 100일때 알고리즘1의 수행시간 t1은 4*(10^13) years 이고
n 이 1,000만(=10^7)일때 알고리즘2의 수행시간 t2는 1.16days이다.