[과제 #1] 수행시간 비교 – 분석 레포트

202002397 통계학과 이도은

수행 환경은 구글 colab을 이용하였다.

```
import time, random

def compute_e_ver1(n):
  # code for O(n^2)-time version
  denom = 1
  e = 1
  for i in range(n):
    for j in range(1,i+1):
      e = e + 1/ (denom * j)



def compute_e_ver2(n):
  # code for O(n)-time version
  e = 1
  for i in range(n):
    if i  == 0 :
      e = 1
    else:
      e = e * 1 / i

# n 입력받음
n = int(input())

# compute_e_ver1 호출
before_1 = time.process_time()
compute_e_ver1(n)
after_1 = time.process_time()

# compute_e_ver2 호출
before_2 = time.process_time()
compute_e_ver2(n)
after_2 = time.process_time()

# 두 함수의 수행시간 출력
print("compute_e_ver1의 수행시간 :", after_1 - before_1)
print("compute_e_ver2의 수행시간 :", after_2 - before_2)
```

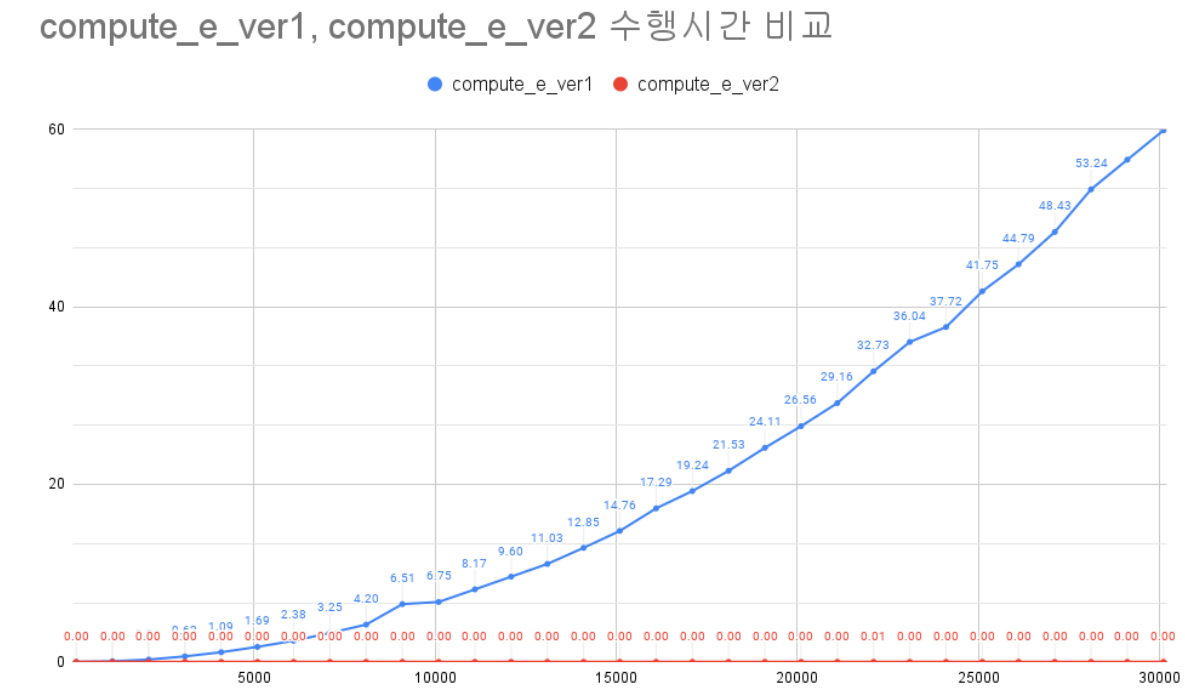
Compute_e_ver_1는 이중 for루프문이 포함되어 있다. Worst case running time을 계산해보았을 때 수행시간의 최고 차항은 n^2이다. 따라서 이를 Big-O표기법으로 표현하면 $O(n^2)$으로 표기할 수 있다.

Compute_e_ver_2는 단일 for 루프문이 포함 되어있다. 위와 같이 Worst case running time을 계산해보았을 때 수행시간의 최고 차항은 n이고 이를 Big-O 표기법으로 표현하면 $O(n)$ 으로 표기할 수 있다.

이제 각 함수를 실행하여 n 값이 변함에 따라 실제 수행시간이 어떻게 바뀌는지 실습해본다. 문제에서 제시한대로 $n = 100 \sim 500000$까지 범위 내에서 시행했다. for문을 이용하여 100부터 시작하여 1000 단위로 증가하며 60초를 초과하지 않는 n까지 수행시간을 측정했다.

수행시간을 측정해보았을 때, n = 30100일 때 compute_e_ver1의 수행시간이 59.88로 이후의 n에

서는 60초를 초과하게 되어 측정을 중단했다. 이를 시각화한 차트를 보았을 때 n이 증가함에 따라, compute_e_ver1의 수행시간이 폭발적으로 증가하는 것을 볼 수 있다. 즉, n이 증가할 수록 수행시간이 $O(n^2)$인 compute_e_ver1이 compute_e_ver2 보다 극명하게 수행시간이 차이남을 확인할 수 있었다.



n	100	1100	2100	3100	4100	5100	6100	7100	8100	9100	10100	11100	12100	13100	14100	15100	16100	17100	18100	19100	20100	21100	22100	23100	24100	25100	26100	27100	28100	29100	30100
compute_e_ver1	0.000	0.072	0.264	0.618	1.087	1.691	2.378	3.254	4.199	6.509	6.749	8.167	9.600	11.034	12.850	14.756	17.294	19.245	21.526	24.113	26.557	29.156	32.735	36.044	37.722	41.752	44.794	48.435	53.235	56.571	59.889
compute_e_ver2	0.000	0.000	0.000	0.000	0.000	0.001	0.001	0.001	0.001	0.001	0.001	0.002	0.001	0.002	0.002	0.002	0.002	0.002	0.002	0.002	0.003	0.003	0.006	0.003	0.004	0.003	0.004	0.004	0.003	0.004	0.004

[느낀점]

이를 통해 코드를 짜는 사람이 얼마나 효율적으로 짜는 가는 입력 값의 크기가 커질 때 더욱 중요해짐을 알 수 있었다. 이때까지 코딩하면서 입력 값의 크기는 크지 않았기 때문에 수행시간이 크게 문제가 된 적은 없었다. 그러나 앞으로 개발자로써 성장하면서 입력 값이 매우 큰 상황들을 마주하게 될 것이고 이런 상황에 대비하여 효율적으로 코드를 짜는 법을 길러야 한다는 생각이 들게 해준 유익한 실습이었다.