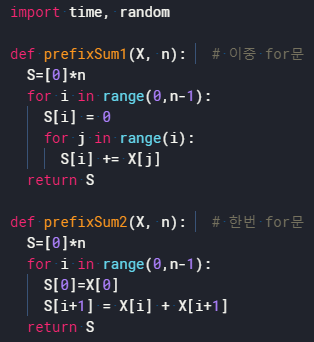
**자료구조 및 실습 레포트**

201903868 황가은



Prefixsum 은 X = [3, 1, -2, 6, 7] 리스트가 있을 때, S = [3, 4, 2, 8, 15] 가 되도록 만드는 함수이다. 위의 코드는 이중 for문을 쓴 prefixSum1 과 for문 한 개를 쓴 prefixSum2의 수행시간(시간복잡도)을 비교하는 코드이다.

prefixSum1은 구조 상 O()의 시간복잡도를 가지며, prefixSum2는 O()의 시간복잡도를 가진다. 시간복잡도는 문제를 해결하는 데 걸리는 시간과 입력의 함수 관계를 나타낸다. 또한 시간복잡도는 가장 복잡한 케이스일 경우를 생각하여 계산한다. 주로 Big-O 표기법으로 나타내는데 이 표기법은 계수와 낮은 차수의 항을 제외한 최고차항만을 이용한다. 만약 크기 의 모든 입력에 관해 필요한 시간이 최대 의 식을 가진다면, 이 알고리즘의 시간복잡도는 O() 이라고 할 수 있다.

prefixSum1 함수의 의 모든 입력에 관해 필요한 시간을 계산해보면 이 되며, prefixSum2 함수의 의 모든 입력에 관해 필요한 시간은 이 된다. prefixSum1 은 O()이며, prefixSum2 은 O()을 가진다. O(n)은 선형적으로 비례하지만, O(n2)은 제곱에 비례하므로 n의 크기가 커질수록 차이가 커지게 된다.

수행시간의 차이를 알아보기 위해 1000 부터 100000사이의 수를 넣어 계산한 결과를 그래프와 표로 나타내면 아래와 같다.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | **1000** | **3000** | **5000** | **8000** | **10000** | **20000** |
| **prefixSum1** | 0.044 | 0.39 | 1.108 | 2.85 | 4.75 | 20.3 |
| **prefixSum2** | 0.00026 | 0.0005 | 0.0007 | 0.0013 | 0.0016 | 0.0035 |

자료들을 보면 한 개의 반복문을 가진 prefixSum2 가 이중 반복문을 가진 prefixSum1 보다 수행시간이 굉장히 빠른 것을 알 수 있다. N의 값이 작을수록 prefixSum1 과 prefixSum2의 수행시간 차이는 적지만, N의 값이 커질수록 수행시간의 차이가 굉장히 커진다. N의 값을 30000으로 입력했을 때 수행시간이 60초를 넘어가서 실행되지 않았다.

두 함수가 어느 구간에서 빠르고 느린지를 확인하기 위해 교점을 찾아보면 n=0, n=1일 때이다. 중간이라고 볼 수 있는 n=1/2 일 때 prefixSum1은 1.875를 가지며, prefixSum2는 2를 가진다. 그러므로 n>0 이며 n<1 일 때 prefixSum1이 prefixSum2 보다 수행시간이 빠르다는 것을 알 수 있다. N>1 이면 prefixSum1이 prefixSum2 보다 수행속도가 현저히 느리게 된다.