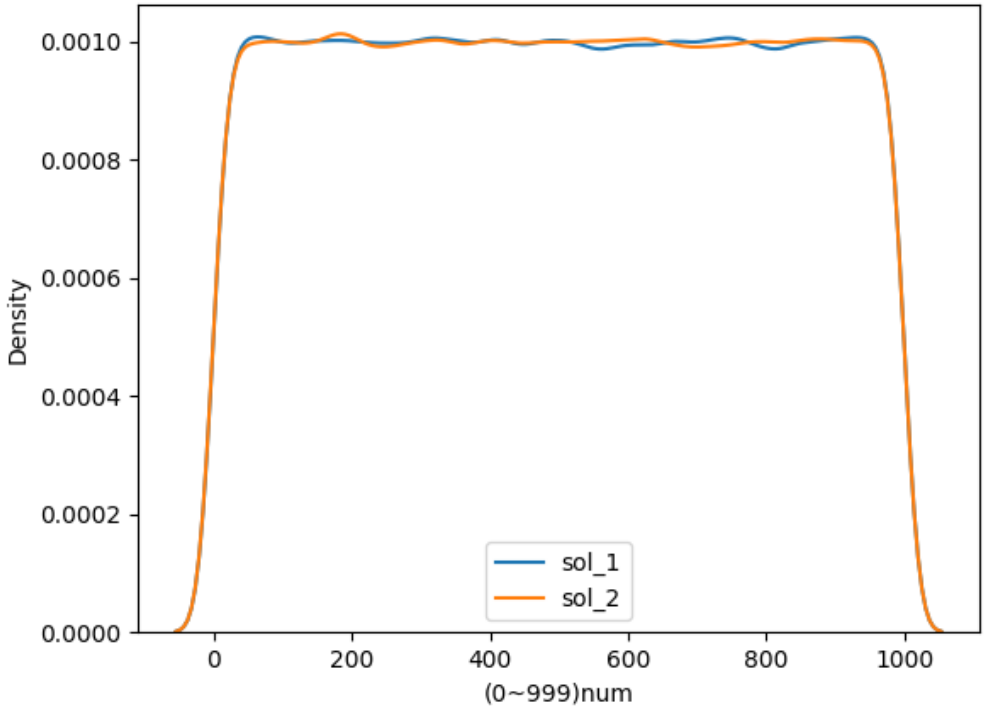
Reservoir Sampling / DGIM Algorithm

이름 : 임현진 학번 : 20181684

Reservoir sampling을 비복원 추출 방법과 복원 추출 방법 2가지 경우로 구현해보았습니다.

비복원의 경우에는 기존의 reservoir sampling 방법을 사용하였고, 복원 추출의 경우에는 reservoir sample(1) 짜리를 추출횟수만큼 반복하는 방법을 사용하였습니다.

그 결과



다음과 같습니다.

sol_1 : 비복원 추출 그래프

sol_2 : 복우너 추출 그래프

x축 : 0~999까지의 integer

y축 : 0~999까지 숫자 중에서 100개 추출하고 그것을 10000번 반복시행 하였을 때, x축 값이 나타난 비율

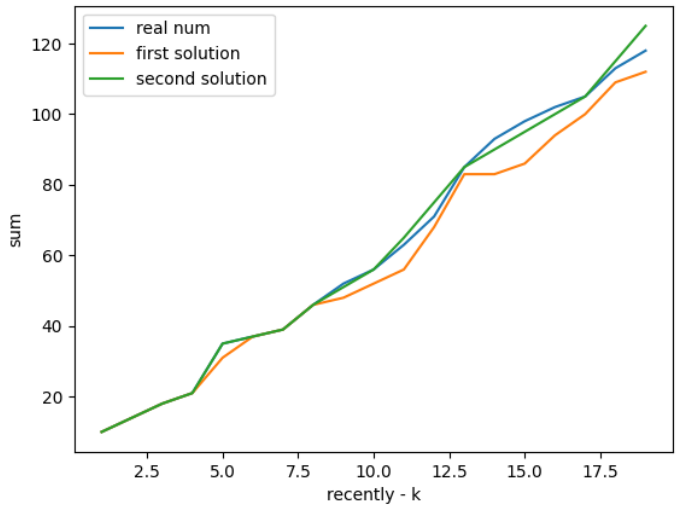
그래프를 보면 알 수 있듯이, Reservoir Sampling을 복원 추출과 비복원 추출 방법으로 시행했을 때, 모두 값이 균등하게 추출되었다는 것을 알 수 있습니다.

2. DGIM알고리즘

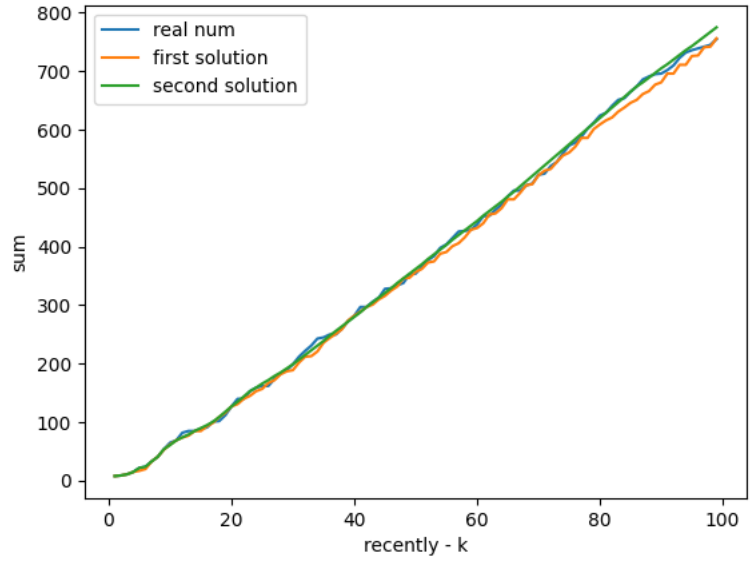
dgim알고리즘을 바탕으로 m개 비트를 별도의 스트림으로 보고 각 각에 1,2,4,8 등을 곱하여 최근 k개의 합을 구하는 첫번째 방법과 부분합을 이용하여 최근 k개의 합을 구하는 두번째 방법을 사용하였습니다.

최근 더해야하는 k개의 수가 커지면 커질수록 그래프를 분석하기 어려워서 값을 k=20, k=100, k=2000일 때 경우를 그래프로 나타내 보았습니다.

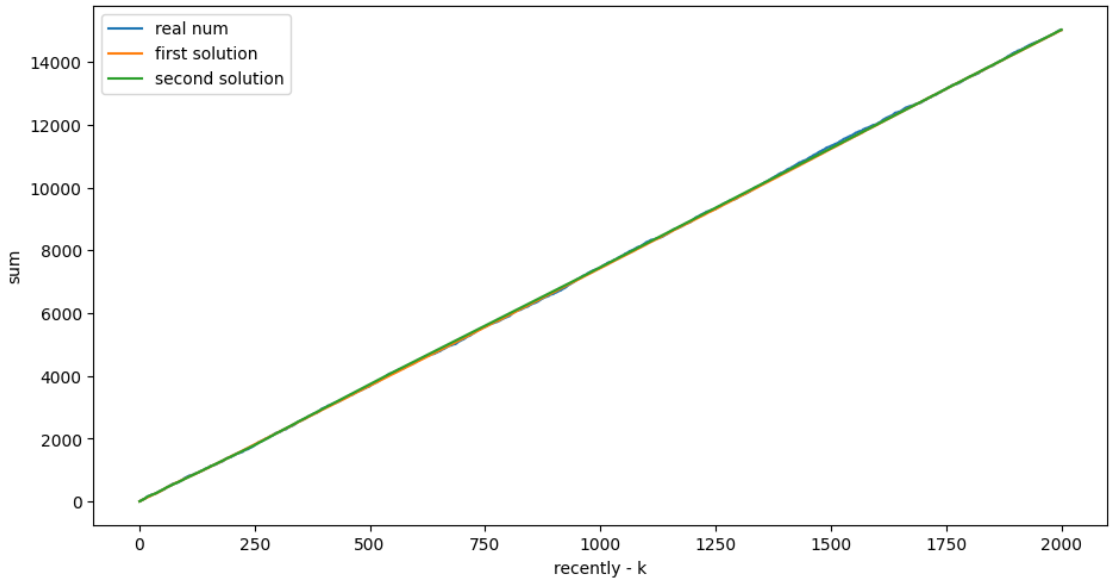
K=20



K=100



K=2000



real num : 실제 값들의 합

first solution : m개 비트를 별도의 스트림으로 보고 각 각에 1,2,4,8 등을 곱하는 방법으로 구한 합

second solution : 부분합을 이용하여 구한 합

x축 : 최근 k개의 수

y축 : 최근 k개의 합

위의 그래프를 보면 대체로 실제 합과 dgim알고리즘을 이용하여 구한 합의 오차가 크지 않다는 것을 알 수 있습니다.

그럼에도 좀더 정확하게 합을 구한 것은 부분합을 이용하여 구한 두번째 알고리즘 방법 이라고 생각합니다. 왜냐하면 첫번째 방법으로 구한 합이 두번째 방법으로 구한 합에 비해 값이 많이 튀어서 오차가 더 크게 났다는것을 k=20, k=100일 때의 그래프에서 확인 할 수 있었기 때문입니다.