**고급 소프트웨어실험 10주차 과제**

**실습 문제 1**

1. 행렬 계산의 속도 향상을 위하여 자신이 적용한 방법을 기술하여라.

* 첫째로, 메모리 접근의 응집성을 높여 속도를 향상시켰다. 이 때, 행의 값을 바꾸어주며 메모리를 이동하는 것 보다는 열의 값을 바꾸며 이동하는 것이 접근의 응집성이 높은 방법이므로, B행렬의 전치 행렬을 구해 이를 열 우선적으로 접근하여 곱셈을 수행하는 방식을 사용하였다.
* 둘째로, Loop Unrolling 방식을 사용하였다.

1. 어떤 근거로 자신이 적용한 방법이 더 효율적일 지 기술하여라.

* 첫 번째 방법을 사용했을 때, 열 우선적으로 접근하면 참조하는 메모리가 바뀔 때 이동해야하는 거리가 짧아지고, 이를 통해 곱셈을 수행하는 데에 필요한 시간을 줄일 수 있다.
* 두 번째 방법에서 loop문에서 조건을 확인하는 횟수를 줄임으로써 연산 량을 줄일 수 있고, 이를 통해 곱셈을 수행하는 데에 필요한 시간을 줄일 수 있다.

1. 어떤 m값에 대하여 loop unrolling 방법이 가장 효과적이었는지를 요약하여 기술하여라.

* 수행시간을 비교해보았을 때 큰 차이는 없었지만, m이 16인 경우에서 가장 수행시간이 짧았고, 이를 통해 m이 16 일 때 가장 효과적이라고 생각할 수 있다.

**실습 문제 2**

1. 각 항목을 통해 발견한 사실을 기술하여라.

* Horner’s Rule을 사용하였을 때, 곱셈 연산의 수를 줄이는 것을 통해 전체 수행시간을 줄일 수 있었다.

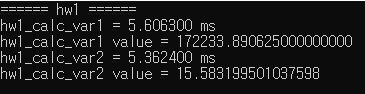
**실습 문제 3**

1. 각 항목을 통해 발견한 사실을 기술하여라.

* Horner’s Rule을 사용하였을 때, 곱셈 연산의 감소로 전체 수행시간이 줄었다.
* Float 타입을 사용했을 때, 사용하는 bit의 수가 적어져 부동소수점 연산을 하는 과정에서 오차가 발생했을 때 그 오차가 더 크게 발생하였다. 따라서 결과를 보았을 때 double 타입을 사용했을 때 보다 오차가 크게 발생한 것을 확인할 수 있었다.

**과제 문제 1**

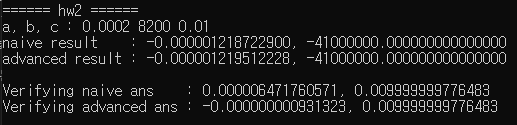
식을 통해 살펴보았을 때, 2번 식에서는 비슷한 숫자끼리의 뺄셈을 수행하게 되어 오차가 발생할 확률이 더 높고, 1번 식의 경우 비슷한 숫자끼리의 뺄셈은 최대한 피했으나 곱셈 연산량이 늘어 수행시간이 더 오래 걸릴것으로 기대하였다.

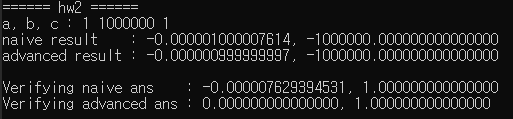


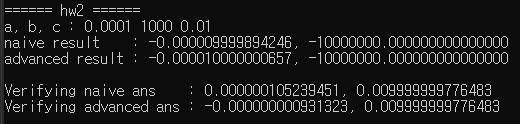
그러나, n값을 바꾸며 관찰해 보았을 때, 1번 값의 결과로 음수가 나오는 등 기대와 어긋나는 결과가 도출되었다.

**과제 문제 2**

에서 의 값이 매우 작은 경우 비슷한 숫자끼리 뺄셈을 수행하게 되어 오차가 발생할 수 있다. 구체적인 경우는 다음과 같다.







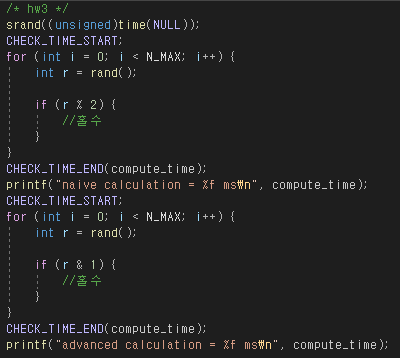
이 때 Verifying naive ans의 값을 보면 오차가 비교적 크게 나타나는 것을 볼 수 있다.

이 문제를 해결하기 위하여, b가 양수이고 b2이 4ac의 값보다 훨씬 큰 경우에 를 로 바꾸어 계산하였다. Verifying advanced ans와 Verifying naïve ans를 보면 오차가 확연히 줄어든 것을 볼 수 있다.

**과제 문제 3**

1. 산술 연산을 비트 연산으로 전환한다. (cost가 더 적은 연산 사용)

같은 결과를 낼 수 있는 연산을 수행할 때에, cost가 큰 산술연산보다는 AND, OR 등의 비트 연산을 사용하여 코드의 수행시간을 줄일 수 있다.



위는 난수를 생성하여 해당 수가 홀수인지 검사하는 코드로, N\_MAX가 1000000일 때 실행 결과는 다음과 같다.

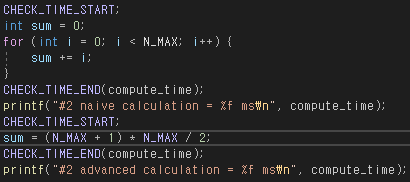


이를 통해, 산술연산보다는 비트연산을 사용할 때에 더 효율적이라는 것을 볼 수 있다.

1. 이미 알고 있는 계산 결과를 활용하여 loop를 줄인다.

for문을 통해 반복적인 계산을 수행할 때에, 계산이 충분히 간단하여 그 결과를 loop 없이 구할 수 있다면 이를 활용하여 계산 시간을 줄일 수 있다.

다음은 1부터 N\_MAX까지의 수를 더하는 두 가지 코드로, 후자의 방식을 사용하는 경우 loop를 사용하지 않아 계산에 소요되는 시간을 줄일 수 있다.



N\_MAX가 1000000일 때 실행 결과는 다음과 같다.

