CS 공부

1. (int) 2의 보수
2. 2의 보수를 쓰는 이유
3. 언더플로우 & 오버플로우 조사
4. (float) 부동소수점
5. 부동소수점 쓰는 이유
6. 1.1 + 0.1 이 1.2 가 아닌 이유

1,2. 음수, 보수

빼기 계산을 하는 경우 그대로 빼기를 하면 계산 속도가 느려진다.

2진수로 빼기 계산을 할 경우 수의 보수를 사용하여 뺀다.

3. 오버플로우, 언더플로우

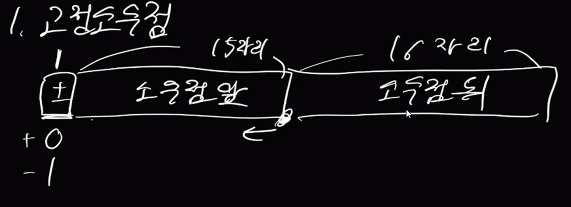
2진수로 저장하기에 최대값, 최소값을 넘는 수를 더하게 되면 최대값, 최소값이 최소값, 최대값이 되어버리는 현상

예시 255 +1 -> -255

4. 고정 소수점, 부동소수점

참고 - <https://steemit.com/kr/@modolee/floating-point>

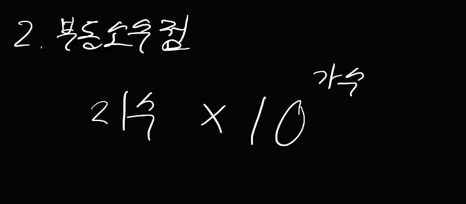
고정소수점



그대로 저장하면 되어 쉽지만 데이터의 크기가 크고 (점 앞 점 뒤 모두 저장할 공간이 필요) 느리기떄문에 효율적이지 않다.

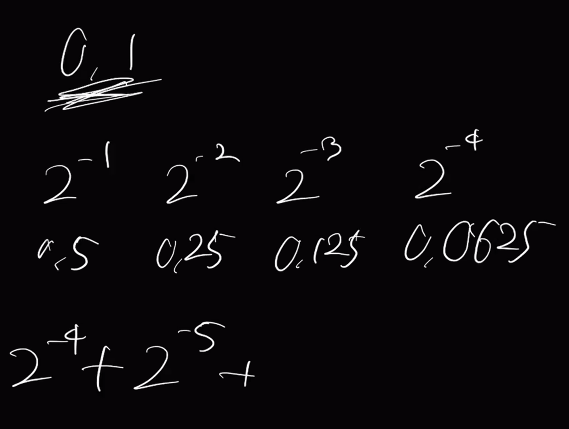
-> 이를 효율적인 방식으로 저장하기 위한 방법으로 부동소수점이 만들어졌다.

부동소수점



5.

6.부동소수점의 저장 방식 때문에 그렇다.



2진수로 0.1을 표현할 방식이 없기에 근소한 값을 저장한다. 즉 1.1+0.1을 했을 때 1.2가 아닌 근소한 값을 반환하기에 오류가 나오는 것이다.

Thread, coroutine 차이

실제 병렬처리, 싱글 쓰레드지만 병렬로 돌리는 방식

Coroutine은 다른 시스템이 어떤 리소스를 사용 중에 이 리소스를 사용하려는 쪽이 계속 뒤로 밀려 처리속도가 느려지는 것을 방지하기 위하여 리소스, 작업에 lock을 걸고, 리소스 사용 순서를 정해주는 방식을 사용