Big Numbers

Julian Kenwood and Ben Steenhuisen

Big Numbers

Big Numbers in Programming Languages

Java: BigInteger

Python: Built-in

C/C++: None

Pascal: None

Big Numbers

C++ has no standard Big Number Library,

However, C++ is the preferred language for the IOI.

Therefore, you must learn how to write your own Big Number library.

Big Numbers

Question: When is it necessary to use Big Numbers?

Answer: When the standard integer datatypes won't cut it.

Big Numbers

Ranges of built-in datatypes

char: [-128, 127]
unsigned char: [0, 255]
short: [-32768, 32767]
unsigned short: [0, 65535]
int: [-2147483648, 2147483647]
unsigned int: [0, 4294967296]
long long: [-9223372036854775808, 9223372036854775807]
unsigned long long: [0, 18446744073709551616]

Big Numbers

The fastest datatypes are the ints.

The other built-in datatypes are next.

Big Numbers are by far the slowest.

Big Numbers

How do you code a Big Number library?

Big Numbers

Represent the number as a list of digits.

Take advantage of the fact that ints can store largish numbers, use a bigger base than 10.

But use a ^10 base, for easier printing

Sign bit only if necessary!

Big Numbers

You will need to code the various operators yourself!

However, don't code all the operators! Only the subset that you need.

Big Numbers

If you don't know big your numbers will get. Then use a vector.

If you are using a vector then it would probably be better to implement the 'int get(int digit)' and 'void set(int digit, int value)' functions.

They will allow you to handle cases where you need to grow your vector or need to access a digit past the end of the vector.

Big Numbers

Constructing a Big Number from a built-in datatype

Involves getting the digits of the number in the base your Big Number is in

Handle the zero case and negative cases!

Big Numbers

Another useful function: reduce()

When we perform most operations with Big Numbers we get carry digits which we must handle.

Instead of handling the carry digits separately, we write one function to handle it in all cases. This will also handle growing of the list.

Big Numbers

Addition of two (positive) Big Numbers

Same as how you did it in primary school.

Add the corresponding digits to get the next digit.

Remember don't worry about the carry.

Big Numbers

Comparing Big Numbers

When is A > B ?

If A < 0 and B > 0 then it is false
Else If A < 0 and B < 0 then it is -B < -A
Else If size A != size B then it is size A > size B
Otherwise compare the number lexicographically (STL :))

Big Number

Subtraction of two (positive) Big Numbers

Only handle subtraction with a positive result.
In A – B
If A > B then A – B > 0
If A < B then A – B < 0 so A – B = -(B – A)

Use borrow digits instead of carry digits, so no need to reduce.
Must handle leading zeroes in the answer!

Big Number

Multiplication of two Big Numbers

Sign bit can be handled separately

Slower than addition, however there are faster algorithms than the one presented.

Must be careful about the choice of base.

Remember to reduce()!

Big Number

Division of a Big Number and an int

Sign can be handled separately

Recall the long division algorithm?

Again, be careful about the choice of base.

Big Numbers

Tricks

Choice of base

get and set methods

Reduce method

Operator overloading and proper constructors

Big Numbers

Conclusion

Increasing occurrence in competitions like COCI.

Easy points, just understand how numbers work!

Related topic: Big Decimals, harder but probably not in the IOI.