http://algs4.cs.princeton.edu

1.4 Analysis of Algorithms

- introduction
- observations
- mathematical models
- order-of-growth classifications
- theory of algorithms
- memory

Basics

Bit. 0 or 1.

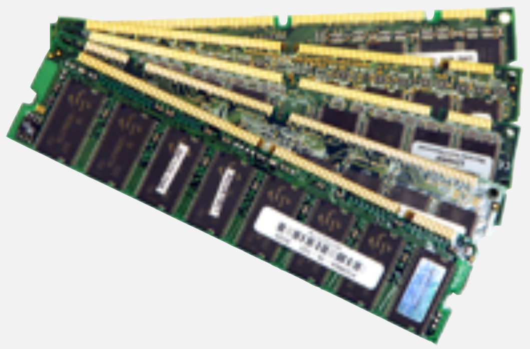
Byte. 8 bits.

NIST

most computer scientists

Megabyte (MB). 1 million or 2^{20} bytes.

Gigabyte (GB). 1 billion or 2^{30} bytes.



64-bit machine. We assume a 64-bit machine with 8-byte pointers.

some JVMs "compress" ordinary object pointers to 4 bytes to avoid this cost

Typical memory usage for primitive types and arrays

type	bytes
`boolean`	1
`byte`	1
`char`	2
`int`	4
`float`	4
`long`	8
`double`	8

primitive types

type	bytes
`char[]`	$2N + 24$
`int[]`	$4N + 24$
`double[]`	$8N + 24$

one-dimensional arrays

type	bytes
`char[][]`	$\sim 2MN$
`int[][]`	$\sim 4MN$
`double[][]`	$\sim 8MN$

two-dimensional arrays

Typical memory usage for objects in Java

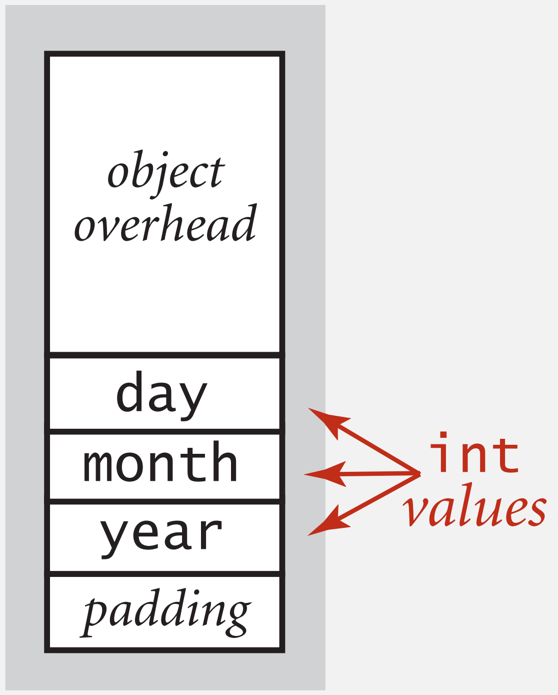
Object overhead. 16 bytes.

Reference. 8 bytes.

Padding. Each object uses a multiple of 8 bytes.

Ex 1. A `Date` object uses 32 bytes of memory.

```
public class Date
{
   private int day;
   private int month;
   private int year;
...
}
```



16 bytes (object overhead)	
4 bytes (int)	
4 bytes (int)	
4 bytes (int)	
4 bytes (padding)	
32 bytes	

Typical memory usage summary

Total memory usage for a data type value:

- Primitive type: 4 bytes for `int`, 8 bytes for `double`, ...
- Object reference: 8 bytes.
- Array: 24 bytes + memory for each array entry.
- Object: 16 bytes + memory for each instance variable.
- Padding: round up to multiple of 8 bytes.

+ 8 extra bytes per inner class object
(for reference to enclosing class)

Note. Depending on application, we may want to count memory for any referenced objects (recursively).