CSCI 136: Data Structures and Advanced Programming

Lecture 21
Ordered Vector / Numbers

Instructor: Dan Barowy Williams

Topics

• Ordered Vector
• Integer representation

Your to-dos

1. Read before Wed: Bailey, Ch 14-14.1, 14.3
2. Quiz due Sat by 6pm

Announcements

• CS Colloquium this Friday, Nov 4 @ 2:35pm in Wege Auditorium (TCL 123)

Rachit Nigam (Cornell University)
Programming Support for Hardware Accelerators

Rachit Nigam is a visiting researcher in the PLSE group at University of Washington and a PhD candidate studying computer science at Cornell University. He is a part of the CAPRA and PL@Cornell research groups and is advised by Adrian Sampson. His research (Dahlia, Calyx) is focused on building high-level programming models for designing hardware accelerators.
Nonetheless, we can signal our intent with an interface.

How would we write an OrderedStructure interface?

Do its elements need to have any special property? (i.e., how would we compare them?)

Let's think about how we might implement this.

(OrderedVector)

Integer representation

The bits of an integer

An integer is represented in computer memory as a sequence of bits, each having a value of either 0 or 1. This representation is called binary.

Binary is number system where each digit can take one of two values; i.e., the base of the system is 2.

You are probably more familiar with the base 10 number system, aka decimal.

Any integer can be represented in either system.
Java int

The int data type in Java has 32 bits.

00000000 00000000 00000000 00010111

is the number 23.

\[
\begin{align*}
& (00000000000000000000000000010111)_2 \\
= & (0 \times 2^{31}) + (0 \times 2^{30}) + (0 \times 2^{29}) + (0 \times 2^{28}) \\
+ & (0 \times 2^{27}) + (0 \times 2^{26}) + (0 \times 2^{25}) + (0 \times 2^{24}) \\
+ & (0 \times 2^{23}) + (0 \times 2^{22}) + (0 \times 2^{21}) + (0 \times 2^{20}) \\
+ & (0 \times 2^{19}) + (0 \times 2^{18}) + (0 \times 2^{17}) + (0 \times 2^{16}) \\
+ & (0 \times 2^{15}) + (0 \times 2^{14}) + (0 \times 2^{13}) + (0 \times 2^{12}) \\
+ & (0 \times 2^{11}) + (0 \times 2^{10}) + (0 \times 2^{9}) + (1 \times 2^{8}) \\
+ & (0 \times 2^{7}) + (1 \times 2^{6}) + (1 \times 2^{5}) + (1 \times 2^{4}) \\
= & (23)_{10}
\end{align*}
\]

Bitwise Operations

We can use bitwise operations to manipulate the 1s and 0s in the binary representation

- Bitwise ‘and’: &
- Bitwise ‘or’: |

Also useful: bit shifts

- Bit shift left: <<
- Bit shift right: >>

& and |

Given two integers \(a\) and \(b\), the bitwise or expression \(a \lor b\) returns an integer s.t.

- At each bit position, the result has a 1 if that bit position had a 1 in EITHER \(a\) OR \(b\)

\[
\begin{align*}
& 3 \lor 6 = ? \\
& 011 \lor 110 = 111
\end{align*}
\]

Given two integers \(a\) and \(b\), the bitwise and expression \(a \land b\) returns an integer s.t.

- At each bit position, the result has a 1 if that bit position had a 1 in BOTH \(a\) AND \(b\)

\[
\begin{align*}
& 3 \land 6 = ? \\
& 011 \land 110 = 010
\end{align*}
\]

>> and <<

Given two integers \(a\) and \(i\), the expression \((a << i)\) returns \((a \times 2^i)\)

- Why? It shifts all bits left by \(i\) positions

\[
\begin{align*}
& 1 << 4 = ? \\
& 00001 << 4 = 10000
\end{align*}
\]

Given two integers \(a\) and \(i\), the expression \((a >> i)\) returns \((a / 2^i)\)

- Why? It shifts all bits right by \(i\) positions

\[
\begin{align*}
& 1 >> 4 = ? \\
& 00001 >> 4 = 00000
\end{align*}
\]

\[
\begin{align*}
& 97 >> 3 = ? \\
& 1100001 >> 3 = 1100
\end{align*}
\]
Recap & Next Class

**Today:**
- Binary search
- Number representations

**Next class:**
- Tree ADT