Binary Representation and Bitmasks

Ieuan David Vinluan

July 2024

Introduction

When we represent integers, we use the positional base-10 numeral system.

Introduction

When we represent integers, we use the positional base-10 numeral system.

- We use 10 symbols (0 to 9)

Introduction

When we represent integers, we use the positional base-10 numeral system.

- We use 10 symbols (0 to 9)
- The value of a digit is the product of the digit and some power of 10 (the place value)

Introduction

When we represent integers, we use the positional base-10 numeral system.

- We use 10 symbols (0 to 9)
- The value of a digit is the product of the digit and some power of 10 (the place value)
- The value of the numeral is the sum of the values of each digit

Introduction

When we represent integers, we use the positional base-10 numeral system.

- We use 10 symbols (0 to 9)
- The value of a digit is the product of the digit and some power of 10 (the place value)
- The value of the numeral is the sum of the values of each digit
- In this case, our base or radix is 10

Introduction

Consider the number 6875_{10}, in base-10.

Consider the number 6875_{10}, in base-10.

- The place value of 6 is $10^3 = 1000$, and its value is $6 \cdot 10^3 = 6000$

Consider the number 6875_{10}, in base-10.

- The place value of 6 is $10^3 = 1000$, and its value is $6 \cdot 10^3 = 6000$
- The place value of 8 is $10^2 = 100$, and its value is $8 \cdot 10^2 = 800$

Introduction

Consider the number 6875_{10}, in base-10.

- The place value of 6 is $10^3 = 1000$, and its value is $6 \cdot 10^3 = 6000$
- The place value of 8 is $10^2 = 100$, and its value is $8 \cdot 10^2 = 800$
- The place value of 7 is $10^1 = 10$, and its value is $7 \cdot 10^1 = 70$

Introduction

Consider the number 6875_{10}, in base-10.

- The place value of 6 is $10^3 = 1000$, and its value is $6 \cdot 10^3 = 6000$
- The place value of 8 is $10^2 = 100$, and its value is $8 \cdot 10^2 = 800$
- The place value of 7 is $10^1 = 10$, and its value is $7 \cdot 10^1 = 70$
- The place value of 5 is $10^0 = 1$, and its value is $5 \cdot 10^0 = 5$

Introduction

Consider the number 6875_{10}, in base-10.

- The place value of 6 is $10^3 = 1000$, and its value is $6 \cdot 10^3 = 6000$
- The place value of 8 is $10^2 = 100$, and its value is $8 \cdot 10^2 = 800$
- The place value of 7 is $10^1 = 10$, and its value is $7 \cdot 10^1 = 70$
- The place value of 5 is $10^0 = 1$, and its value is $5 \cdot 10^0 = 5$

Thus, the value of 6875_{10} in base-10 is $6000 + 800 + 70 + 5 = 6875$.

Binary Representation

The binary numeral system, meanwhile, only uses only 2 symbols, 0 and 1, to denote integers.

Binary Representation

The binary numeral system, meanwhile, only uses only 2 symbols, 0 and 1, to denote integers.

Consider the number 1011_2. We can consider the place values of each binary digit (a "bit") to find its value in base-10.

Binary Representation

The binary numeral system, meanwhile, only uses only 2 symbols, 0 and 1, to denote integers.

Consider the number 1011_2. We can consider the place values of each binary digit (a "bit") to find its value in base-10.

- The place value of 1 is $2^3 = 8$, and its value is $1 \cdot 2^3 = 8$

Binary Representation

The binary numeral system, meanwhile, only uses only 2 symbols, 0 and 1, to denote integers.

Consider the number 1011_2. We can consider the place values of each binary digit (a "bit") to find its value in base-10.

- The place value of 1 is $2^3 = 8$, and its value is $1 \cdot 2^3 = 8$
- The place value of 0 is $2^2 = 4$, and its value is $0 \cdot 2^2 = 0$

Binary Representation

The binary numeral system, meanwhile, only uses only 2 symbols, 0 and 1, to denote integers.

Consider the number 1011_2. We can consider the place values of each binary digit (a "bit") to find its value in base-10.

- The place value of 1 is $2^3 = 8$, and its value is $1 \cdot 2^3 = 8$
- The place value of 0 is $2^2 = 4$, and its value is $0 \cdot 2^2 = 0$
- The place value of 1 is $2^1 = 2$, and its value is $1 \cdot 2^1 = 2$

Binary Representation

The binary numeral system, meanwhile, only uses only 2 symbols, 0 and 1, to denote integers.

Consider the number 1011_2. We can consider the place values of each binary digit (a "bit") to find its value in base-10.

- The place value of 1 is $2^3 = 8$, and its value is $1 \cdot 2^3 = 8$
- The place value of 0 is $2^2 = 4$, and its value is $0 \cdot 2^2 = 0$
- The place value of 1 is $2^1 = 2$, and its value is $1 \cdot 2^1 = 2$
- The place value of 1 is $2^0 = 1$, and its value is $1 \cdot 2^0 = 1$

Binary Representation

The binary numeral system, meanwhile, only uses only 2 symbols, 0 and 1, to denote integers.

Consider the number 1011_2. We can consider the place values of each binary digit (a "bit") to find its value in base-10.

- The place value of 1 is $2^3 = 8$, and its value is $1 \cdot 2^3 = 8$
- The place value of 0 is $2^2 = 4$, and its value is $0 \cdot 2^2 = 0$
- The place value of 1 is $2^1 = 2$, and its value is $1 \cdot 2^1 = 2$
- The place value of 1 is $2^0 = 1$, and its value is $1 \cdot 2^0 = 1$

Thus, the value of 1011_2 in base-10 is $8 + 2 + 1 = 11$.

Binary Representation

To convert an integer n in base-10 to base-2, follow this algorithm:

Binary Representation

To convert an integer n in base-10 to base-2, follow this algorithm:

1. Initially, let $a = n$.

Binary Representation

To convert an integer n in base-10 to base-2, follow this algorithm:

1. Initially, let $a = n$.
2. Take $a \pmod 2$. This is the next rightmost bit of our binary representation.

Binary Representation

To convert an integer n in base-10 to base-2, follow this algorithm:

1. Initially, let $a = n$.
2. Take $a \pmod 2$. This is the next rightmost bit of our binary representation.
3. Set $a := \left\lfloor \frac{1}{2} a \right\rfloor$

Binary Representation

To convert an integer n in base-10 to base-2, follow this algorithm:

1. Initially, let $a = n$.
2. Take $a \pmod 2$. This is the next rightmost bit of our binary representation.
3. Set $a := \left\lfloor \dfrac{1}{2} a \right\rfloor$
4. Repeat steps 2 and 3 until $a = 0$.

Example: Convert 11_{10} to binary.

Example: Convert 11_{10} to binary.

Example

These are the iterations of our algorithm:

$$\begin{aligned} 11 &= 2 \cdot 5 + 1 \\ 5 &= 2 \cdot 2 + 1 \\ 2 &= 2 \cdot 1 + 0 \\ 1 &= 2 \cdot 0 + 1 \end{aligned}$$

Thus, $11_{10} = 1011_2$, which checks out!

Exercises :3

Convert the following decimal integers into binary:

Exercises :3

Convert the following decimal integers into binary:

- 924_{10}

Exercises :3

Convert the following decimal integers into binary:

- $924_{10} = 1110011100_2$

Exercises :3

Convert the following decimal integers into binary:

- $924_{10} = 1110011100_2$
- 2007_{10}

Exercises :3

Convert the following decimal integers into binary:

- $924_{10} = 1110011100_2$
- $2007_{10} = 11111010111_2$

Exercises :3

Convert the following decimal integers into binary:

- $924_{10} = 1110011100_2$
- $2007_{10} = 11111010111_2$
- 1922_{10}

Exercises :3

Convert the following decimal integers into binary:

- $924_{10} = 1110011100_2$
- $2007_{10} = 11111010111_2$
- $1922_{10} = 11110000010_2$

Exercises :3

Convert the following decimal integers into binary:

- $924_{10} = 1110011100_2$
- $2007_{10} = 11111010111_2$
- $1922_{10} = 11110000010_2$
- 1991_{10}

Exercises :3

Convert the following decimal integers into binary:

- $924_{10} = 1110011100_2$
- $2007_{10} = 11111010111_2$
- $1922_{10} = 11110000010_2$
- $1991_{10} = 11111000111_2$

Bitwise Operations

We can perform operations on the individual bits of the binary representations of integers, instead of the integers themselves. We'll go over a few important ones that we'll be needing later.

Bitwise AND

- Syntax: `i & j`
- For every corresponding bit of two integers, set the resulting bit to 1 if both bits are 1 and 0 otherwise

Bitwise AND

- Syntax: `i & j`
- For every corresponding bit of two integers, set the resulting bit to 1 if both bits are 1 and 0 otherwise

Example: evaluate `23 & 18`

$$
\begin{array}{cccccc}
 & 1 & 0 & 1 & 1 & 1 \\
\& & 1 & 0 & 0 & 1 & 0 \\
\hline
 & 1 & 0 & 0 & 1 & 0
\end{array}
$$

Thus, `23 & 18` evaluates to $10010_2 = 18_{10}$.

Bitwise Left Shift

- Syntax: `i << j`
- Shift every bit to the left by `j` bits; the bits to the right are filled with `0`s

Bitwise Left Shift

- Syntax: `i << j`
- Shift every bit to the left by `j` bits; the bits to the right are filled with 0s

Example: evaluate `3 << 2`

$$\begin{array}{ccccc} 0 & 0 & 0 & 1 & 1 \\ & & \downarrow & & \\ \hline 0 & 1 & 1 & 0 & 0 \end{array}$$

Thus, `3 << 2` evaluates to $1100_2 = 12_{10}$.

Bitwise Right Shift

- Syntax: `i >> j`
- Shift every bit to the right by `j` bits; the bits to the right are filled with the most significant bit, which is the sign bit (0 if positive; 1 if negative)

Bitwise Right Shift

- Syntax: `i >> j`
- Shift every bit to the right by `j` bits; the bits to the right are filled with the most significant bit, which is the sign bit (0 if positive; 1 if negative)

Example: evaluate `15 >> 2`

$$\begin{array}{ccccc} 0 & 1 & 1 & 1 & 1 \\ & & \downarrow & & \\ \hline 0 & 0 & 0 & 1 & 1 \end{array}$$

Thus, `15 >> 2` evaluates to $11_2 = 3_{10}$.

Binary Representation

How is this relevant in CompProg?

Binary Representation

How is this relevant in CompProg?

- Recall that binary numerals only use two symbols to represent integers

Binary Representation

How is this relevant in CompProg?

- Recall that binary numerals only use two symbols to represent integers
- With these two symbols, we can represent yes and no, true and false, present and absent

Binary Representation

How is this relevant in CompProg?

- Recall that binary numerals only use two symbols to represent integers
- With these two symbols, we can represent yes and no, true and false, present and absent
- Thus, representing integers as a series of bits–bitmasks–can be a powerful tool when we need to access all subsets of a set!

Consider a 5-element set $S = \{1, 3, 5, 7, 9\}$, whose elements are indexed from 0 to 4.

Bitmasks

Consider a 5-element set $S = \{1, 3, 5, 7, 9\}$, whose elements are indexed from 0 to 4.

Let us represent a subset of this set using a binary integer, with the rightmost bit indicating the 0th index. If the bit at index k is 1, then the element at index k is included in the subset; otherwise, it is not included.

Bitmasks

Consider a 5-element set $S = \{1, 3, 5, 7, 9\}$, whose elements are indexed from 0 to 4.

Let us represent a subset of this set using a binary integer, with the rightmost bit indicating the 0th index. If the bit at index k is 1, then the element at index k is included in the subset; otherwise, it is not included.

0	1	2	3	4
0	1	0	1	1

We include the elements at indices 1, 3, and 4 in our subset. Thus, the integer $11010_2 = 26_{10}$ represents the subset $\{3, 7, 9\}$.

Questions :D

Consider an n-element set S. Answer the following questions:

Questions :D

Consider an n-element set S. Answer the following questions:

- How would you represent an empty subset with an integer?

Questions :D

Consider an n-element set S. Answer the following questions:

- How would you represent an empty subset with an integer?
  $\rightarrow 0$

Questions :D

Consider an n-element set S. Answer the following questions:

- How would you represent an empty subset with an integer?
  $\rightarrow 0$
- What integer represents the improper subset of S?

Questions :D

Consider an n-element set S. Answer the following questions:

- How would you represent an empty subset with an integer?
  $\rightarrow 0$
- What integer represents the improper subset of S?
  $\rightarrow 2^n - 1$

Questions :D

Consider an n-element set S. Answer the following questions:

- How would you represent an empty subset with an integer?
  $\rightarrow 0$
- What integer represents the improper subset of S?
  $\rightarrow 2^n - 1$
- What is the range of values for the integers representing a subset of this n-element set?

Questions :D

Consider an n-element set S. Answer the following questions:

- How would you represent an empty subset with an integer?
  $\rightarrow 0$
- What integer represents the improper subset of S?
  $\rightarrow 2^n - 1$
- What is the range of values for the integers representing a subset of this n-element set?
  $\rightarrow [0, 2^n - 1]$ (why?)

Bitmasks

Using everything we know, how can we determine which elements are part of a subset given a bitmask m?

Bitmasks

Using everything we know, how can we determine which elements are part of a subset given a bitmask m?

- We know that the ith element is part of the subset if the ith bit of m is 1

Bitmasks

Using everything we know, how can we determine which elements are part of a subset given a bitmask m?

- We know that the ith element is part of the subset if the ith bit of m is 1
- We can check the value of the ith bit by bit-shifting m right by i bits, and then doing a bitwise AND with the result and 1

Bitmasks

Using everything we know, how can we determine which elements are part of a subset given a bitmask m?

- We know that the ith element is part of the subset if the ith bit of m is 1
- We can check the value of the ith bit by bit-shifting m right by i bits, and then doing a bitwise AND with the result and 1
- Alternatively, we can do the same thing by bit-shifting 1 left by i bits, and then doing a bitwise AND with the result and m

Bitmasks

Using everything we know, how can we determine which elements are part of a subset given a bitmask m?

- We know that the ith element is part of the subset if the ith bit of m is 1
- We can check the value of the ith bit by bit-shifting m right by i bits, and then doing a bitwise AND with the result and 1
- Alternatively, we can do the same thing by bit-shifting 1 left by i bits, and then doing a bitwise AND with the result and m

Why do these work?

Bitmasks

Now, let us translate what we know into code. Let's once again consider $S = \{1, 3, 5, 7, 9\}$ and the bitmask given by 11010_2. If an element is in the subset given by our bitmask, let's print a statement saying so!

Bitmasks

Now, let us translate what we know into code. Let's once again consider $S = \{1, 3, 5, 7, 9\}$ and the bitmask given by 11010_2. If an element is in the subset given by our bitmask, let's print a statement saying so!

```
vector<int> S = {1, 3, 5, 7, 9};
int mask = 26;
for (int i = 0; i < 5; i++) {
    if ((mask >> i) & 1) {
        cout << S[i] << " is in the subset!" <<
            endl;
    }
}
```

Bitmasks

This is the output of the code:

```
3 is in the subset!
7 is in the subset!
9 is in the subset!
```

As expected, the elements with indices 1, 3, and 4 are included in our subset!

Bitmasks

Recall that for a set S containing n elements, the bitmasks representing subsets of this set range from 0 to $2^n - 1$. Then, enumerating each subset is simple!

```
vector<int> S = {1, 3, 5, 7, 9}; // the set
int n = 5; // the size of the set
for (int mask = 0; mask < (1 << n); mask++) {
    for (int i = 0; i < n; i++) {
        if ((mask >> i) & 1) {
            cout << S[i] << " ";
        }
    }
    cout << endl;
}
```

Bitmask Enumeration

A submask of a given mask m is a mask whose set bits are a subset of the set bits of m. You can think of submasks as subsets of a subset.

For example, given a mask $m = 1011_2$, the mask 1000_2 is a submask of m, but 1110_2 is not.

Bitmask Enumeration

How can we go through all of the submasks of a given bitmask?

Bitmask Enumeration

How can we go through all of the submasks of a given bitmask?

Here are some useful properties of binary integers that can help us out:

Bitmask Enumeration

How can we go through all of the submasks of a given bitmask?

Here are some useful properties of binary integers that can help us out:

- Subtracting an integer by 1 flips the rightmost set bit and all the bits to its right

Bitmask Enumeration

How can we go through all of the submasks of a given bitmask?

Here are some useful properties of binary integers that can help us out:

- Subtracting an integer by 1 flips the rightmost set bit and all the bits to its right
- That is, the rightmost set bit becomes 0, and all the bits to its right become 1

Bitmask Enumeration

How can we go through all of the submasks of a given bitmask?

Here are some useful properties of binary integers that can help us out:

- Subtracting an integer by 1 flips the rightmost set bit and all the bits to its right
- That is, the rightmost set bit becomes 0, and all the bits to its right become 1
- We can use the & bitwise operator to keep the set bits as a subset of our given bitmask

Bitmask Enumeration

How can we go through all of the submasks of a given bitmask?

Here are some useful properties of binary integers that can help us out:

- Subtracting an integer by 1 flips the rightmost set bit and all the bits to its right
- That is, the rightmost set bit becomes 0, and all the bits to its right become 1
- We can use the & bitwise operator to keep the set bits as a subset of our given bitmask

Using these, we can enumerate the submasks of a given mask

Bitmask Enumeration

Let us implement this in code.

Bitmask Enumeration

Let us implement this in code.

```
vector<int> S = {1, 3, 5, 7, 9};
int mask = 26, submask = mask;
while (submask > 0) {
    for (int i = 0; i < 5; i++) {
        if ((submask >> i) & 1) {
            cout << S[i] << " ";
        }
    }
    cout << endl;
    submask = (submask - 1) & mask;
    if (submask == 0) {
        cout << "Empty subset" << endl;
        break;
    }
}
```

Bitmask Enumeration

Here is the output of our code:

```
3 7 9
7 9
3 9
9
3 7
7
3
Empty subset
```

Indeed, we have successfully iterated through the submasks!

Understanding the Algorithm

Answer the following:

Answer the following:

- What happens if we remove the `if` block explicitly telling our program to break out of the `while` loop once `submask` becomes 0?

Understanding the Algorithm

Answer the following:

- What happens if we remove the `if` block explicitly telling our program to break out of the `while` loop once `submask` becomes 0?
- What if do the same thing but change the boolean condition of the `while` loop from `submask >= 0` to `submask > 0`? What's wrong with this?

More Resources

Here are some more helpful links that you might find useful:

- A bit of fun: fun with bits
- Submask Enumeration

Homework :3

Problems are linked in the Reboot website :3 As always, do not hesitate to ask for help in the Reboot Discord server!