Strange Function

Timon Gehr

Swiss Olympiad in Informatics

January 7, 2017

```
// a >=0, b >= 0
long long funny(long long a, long long b) {
    long long x = 1;
    while (x < b) {
        x *= 2;
    }
    long long s = x % b;
    while (a >= b) {
        a = (a & (x - 1)) + s * (a / x);
    }
    return a;
}
```

```
// a >=0, b >= 0
long long funny(long long a, long long b) {
    long long x = 1;
    while (x < b) {
        x *= 2;
    }
    long long s = x % b;
    while (a >= b) {
        a = (a & (x - 1)) + s * (a / x);
    }
    return a;
}
```

1. What is computed?

```
// a >=0, b >= 0
long long funny(long long a, long long b) {
    long long x = 1;
    while (x < b) {
        x *= 2;
    }
    long long s = x % b;
    while (a >= b) {
        a = (a & (x - 1)) + s * (a / x);
    }
    return a;
}
```

1. What is computed?

```
// a >=0, b >= 0
long long funny(long long a, long long b) {
    long long x = 1;
    while (x < b) {
        x *= 2;
    }
    long long s = x % b;
    while (a >= b) {
        a = (a & (x - 1)) + s * (a / x);
    }
    return a;
}
```

1. What is computed?
2. When does it work?

```
// a >=0, b >= 0
long long funny(long long a, long long b) {
    long long x = 1;
    while (x < b) {
        x *= 2;
    }
    long long s = x % b;
    while (a >= b) {
        a = (a & (x - 1)) + s * (a / x);
    }
    return a;
}
```

1. What is computed?
2. When does it work?

```
// a >=0, b >= 0
long long funny(long long a, long long b) {
    long long x = 1;
    while (x < b) {
        x *= 2;
    }
    long long s = x % b;
    while (a >= b) {
        a = (a & (x - 1)) + s * (a / x);
    }
    return a;
}
```

1. What is computed?
2. When does it work?

Easy subtasks

Task:

- Compute `funny(a, b)` for some specific `(a, b)` values.
- `funny(a, b)` is guaranteed to return a result.

Easy Subtasks: Obvious Solution

```
long long funny(long long a, long long b)
{ ... }

int main() {
    int T;
    cin >> T;
    for (int t = 1; t <= T; t++) {
        long long a, b;
        cin >> a >> b;
        cout << "Case␣#" << t << ":␣";
        cout << funny(a, b) << endl;
    }
}
```

What's the point, then?

Full score only given for

- Constant time.
- Justification of correctness.

Understanding the Code: x

```
long long x = 1;
while (x < b) {
    x *= 2;
}
long long s = x % b;
while (a >= b) {
    a = (a & (x - 1)) + s * (a / x);
}
return a;
```

Understanding the Code: x

```
long long x = 1;
while (x < b) {
    x *= 2;
}
long long s = x % b;
while (a >= b) {
    a = (a & (x - 1)) + s * (a / x);
}
return a;
```

Understanding the Code

$x \leftarrow 2^i$, for i such that $x/2 < b \leq x$.

```
long long s = x % b;
while (a >= b) {
    a = (a & (x - 1)) + s * (a / x);
}
return a;
```

Understanding the Code

$x \leftarrow 2^i$, for i such that $x/2 < b \leq x$.

```
long long s = x % b;
while (a >= b) {
    a = (a & (x - 1)) + s * (a / x);
}
return a;
```

Understanding the Code: Notation

Floor function

For integer n:

$$\lfloor x \rfloor = n \Leftrightarrow n \leq x < n + 1$$

Understanding the Code: Notation

Floor function

For integer n:

$$\lfloor x \rfloor = n \Leftrightarrow n \leq x < n + 1$$

Examples:

$\lfloor 0.5 \rfloor = 0$ $\lfloor 1.6 \rfloor = 1$ $\lfloor 42.3 \rfloor = 42$ $\lfloor -1.3 \rfloor = -2$

Understanding the Code: Notation

Modulo (Knuth)

$$a \bmod b = a - b \cdot \lfloor a/b \rfloor$$

Understanding the Code: Notation

Modulo (Knuth)

$$a \bmod b = a - b \cdot \lfloor a/b \rfloor$$

Example (11 mod 3):

$$\lfloor 11/3 \rfloor = 3, \qquad 3 \cdot \lfloor 11/3 \rfloor = 9, \qquad 11 - 3 \cdot \lfloor 11/3 \rfloor = 2.$$

Understanding the Code: Notation

Modulo (Knuth)

$$a \bmod b = a - b \cdot \lfloor a/b \rfloor$$

Example (11 mod 3):

$$\lfloor 11/3 \rfloor = 3, \qquad 3 \cdot \lfloor 11/3 \rfloor = 9, \qquad 11 - 3 \cdot \lfloor 11/3 \rfloor = 2.$$

N.B. relation to C++

If $a \geq 0$ and $b > 0$:

$$a \bmod b = a\%b, \text{ where } \% \text{ is the corresponding operator in C++.}$$

Caution: The condition is needed!

Understanding the Code

$x \leftarrow 2^i$, for i such that $x/2 < b \leq x$.

```
long long s = x % b;
while (a >= b) {
    a = (a & (x - 1)) + s * (a / x);
}
return a;
```

Understanding the Code

$x \leftarrow 2^i$, for i such that $x/2 < b \leq x$.

$s \leftarrow x - b \cdot \underbrace{\lfloor x/b \rfloor}_{=1}$.

```
while (a >= b) {
    a = (a & (x - 1)) + s * (a / x);
}
return a;
```

Understanding the Code

$x \leftarrow 2^i$, for i such that $x/2 < b \leq x$.
$s \leftarrow x - b$.
while$(a \geq b)$: $\quad a \leftarrow (a\&(x-1)) + s \cdot \lfloor a/x \rfloor$.

```
return a;
```

Understanding the Code

$x \leftarrow 2^i$, for i such that $x/2 < b \leq x$.
$s \leftarrow x - b$.
while$(a \geq b)$: $a \leftarrow (a\&(x-1)) + s \cdot \lfloor a/x \rfloor$.

```
return a;
```

$a\&(x-1)$

base 2

base 10:	0	1	2	3	4	5	6	7	8
base 2:	0	1	10	11	100	101	110	111	1000

$a\&(x-1)$

base 2

base 10:	0	1	2	3	4	5	6	7	8
base 2:	0	1	10	11	100	101	110	111	1000

bitwise AND

$a\&b$ keeps only the bits that are set in both a and b.

Example:

$$5\&6 = 101_{(2)}\&110_{(2)} = 100_{(2)} = 4.$$

$a\&(x-1)$

base 2

base 10:	0	1	2	3	4	5	6	7	8
base 2:	0	1	10	11	100	101	110	111	1000

bitwise AND

$a\&b$ keeps only the bits that are set in both a and b.
Example:

$$5\&6 = 101_{(2)}\&110_{(2)} = 100_{(2)} = 4.$$

$a\&(x-1)$ for $x = 2^i$

- $x = 2^i$

$a\&(x-1)$

base 2

base 10:	0	1	2	3	4	5	6	7	8
base 2:	0	1	10	11	100	101	110	111	1000

bitwise AND

$a\&b$ keeps only the bits that are set in both a and b.

Example:

$$5\&6 = 101_{(2)}\&110_{(2)} = 100_{(2)} = 4.$$

$a\&(x-1)$ for $x = 2^i$

- $x = 2^i$
- $x - 1$ in base 2: $\cdots 00\underbrace{1\cdots1}_{i\text{ times}},$

$a\&(x-1)$

base 2

base 10:	0	1	2	3	4	5	6	7	8
base 2:	0	1	10	11	100	101	110	111	1000

bitwise AND

$a\&b$ keeps only the bits that are set in both a and b.

Example:

$$5\&6 = 101_{(2)}\&110_{(2)} = 100_{(2)} = 4.$$

$a\&(x-1)$ for $x = 2^i$

- $x = 2^i$
- $x-1$ in base 2: $\cdots 00\underbrace{1\cdots 1}_{i \text{ times}}$, therefore $a\&(x-1) = a \bmod x$!

Understanding the Code

$x \leftarrow 2^i$, for i such that $x/2 < b \leq x$.

$s \leftarrow x - b$.

while$(a \geq b)$: $\quad a \leftarrow (a\&(x-1)) + s \cdot \lfloor a/x \rfloor$.

```
return a;
```

Understanding the Code

$x \leftarrow 2^i$, for i such that $x/2 < b \leq x$.
$s \leftarrow x - b$.
while$(a \geq b)$: $a \leftarrow a \bmod x + s \cdot \lfloor a/x \rfloor$.

```
return a;
```

Understanding the Code

$x \leftarrow 2^i$, for i such that $x/2 < b \le x$.
$s \leftarrow x - b$.
while$(a \ge b)$: $\quad a \leftarrow (a - x \cdot \lfloor a/x \rfloor) + s \cdot \lfloor a/x \rfloor$.

```
return a;
```

Understanding the Code

$x \leftarrow 2^i,$ for i such that $x/2 < b \leq x$.

$s \leftarrow x - b$.

$\textbf{while}(a \geq b): \quad a \leftarrow (a - x \cdot \lfloor a/x \rfloor) + s \cdot \lfloor a/x \rfloor$.

```
return a;
```

Understanding the Code

$x \leftarrow 2^i$, for i such that $x/2 < b \leq x$.
while$(a \geq b)$: $\quad a \leftarrow a - x \cdot \lfloor a/x \rfloor + (x - b) \cdot \lfloor a/x \rfloor$.

```
return a;
```

Understanding the Code

$$x \leftarrow 2^i, \text{ for } i \text{ such that } x/2 < b \leq x.$$

$$\textbf{while}(a \geq b): \quad a \leftarrow a \underbrace{-x \cdot \lfloor a/x \rfloor + x \cdot \lfloor a/x \rfloor}_{=0} - b \cdot \lfloor a/x \rfloor.$$

```
return a;
```

Understanding the Code

$x \leftarrow 2^i$, for i such that $x/2 < b \leq x$.
while$(a \geq b)$: $\quad a \leftarrow a - b \cdot \lfloor a/x \rfloor$.
return a.

Understanding the Code

$x \leftarrow 2^i,$ for i such that $x/2 < b \leq x$.
while$(a \geq b)$: $a \leftarrow a - b \cdot \lfloor a/x \rfloor$.
return a.

- $a \bmod b$ does not change!

Understanding the Code

$x \leftarrow 2^i$, for i such that $x/2 < b \leq x$.
while$(a \geq b)$: $\quad a \leftarrow a - b \cdot \lfloor a/x \rfloor$.
return a.

- $a \bmod b$ does not change!
- Upon termination, $0 \leq a < b$.

Understanding the Code

$x \leftarrow 2^i$, for i such that $x/2 < b \leq x$.
while$(a \geq b)$: $a \leftarrow a - b \cdot \lfloor a/x \rfloor$.
return a.

- $a \bmod b$ does not change!
- Upon termination, $0 \leq a < b$.
- `funny(a, b) == a%b`!

Harder Subtasks

- When does the procedure give a result?

Harder Subtasks

- When does the procedure give a result?
- Computes `x % b`, hence it crashes when $b = 0$.
  $\textbf{if}(b = 0): \quad \textbf{return}$ "NOTHING".

Harder Subtasks

- When does the procedure give a result?
- Computes `x % b`, hence it crashes when $b = 0$.
  $\textbf{if}(b = 0): \quad \textbf{return}$ "NOTHING".
- What about termination?
  $x \leftarrow 2^i$, for i such that $x/2 < b \leq x$.
  $\textbf{while}(a \geq b): \quad a \leftarrow a - b \cdot \lfloor a/x \rfloor$.
  $\textbf{return}\ a$.

Harder Subtasks

- When does the procedure give a result?
- Computes `x % b`, hence it crashes when $b = 0$.
  $\textbf{if}(b = 0): \quad \textbf{return}$ "NOTHING".
- What about termination?
  $x \leftarrow 2^i$, for i such that $x/2 < b \leq x$.
  $\textbf{while}(a \geq b): \quad a \leftarrow a - b \cdot \lfloor a/x \rfloor$.
  $\textbf{return}\ a$.
- Nontermination for $b \leq a < x$.

Optimizing running time

Obvious Solution

Run while loop until completion or until a no longer changes.

Optimizing running time

Obvious Solution

Run while loop until completion or until a no longer changes.

.

if$(b = 0)$: **return** "NOTHING".
$x \leftarrow 2^i$, for i such that $x/2 < b \leq x$.
while$(a \geq b)$:
    $a' \leftarrow a - b \cdot \lfloor a/x \rfloor$.
    if$(a' = a)$: **return** "NOTHING".
    $a \leftarrow a$.
return a.

Running time $\mathcal{O}(\log(\min(a, b)))$, not so nice to analyze.

Optimizing running time

Obvious Solution plus Trick

Run while loop until completion or until a no longer changes.

if$(a < b)$: **return** a.

if$(b = 0)$: **return** "NOTHING".

$x \leftarrow 2^i$, for i such that $x/2 < b \leq x$.

while$(a \geq b)$:

  $a' \leftarrow a - b \cdot \lfloor a/x \rfloor$.

  if$(a' = a)$: **return** "NOTHING".

  $a \leftarrow a$.

return a.

Running time $\mathcal{O}(\log(\min(a, b)))$, not so nice to analyze.

Optimizing running time

Getting rid of the loop

We can show: $b \leq a < x$ reached iff: $a \bmod b + b < x$.

Optimizing running time

Getting rid of the loop

We can show: $b \leq a < x$ reached iff: $a \bmod b + b < x$.

if$(a < b)$: **return** a.

$x \leftarrow 2^i$, for i such that $x/2 < b \leq x$.

if$(b = 0 || a \bmod b + b < x)$: **return** "NOTHING".

return $a \bmod b$.

Running time $\mathcal{O}(\log(\log(\min(a, b))))$.

(Binary search for exponent of x using bitshifts.)

Solution in O(1)

Avoiding computation of x

if$(b = 0 || a \geq b \&\& (b \& (b-1)) \neq 0 \&\& ((a \% b + b) \& \sim b) < b)$:

    return "NOTHING".

return a mod b.

Running time $\mathcal{O}(1)$!

Solution in O(1)

Avoiding computation of x

if$(b = 0 || a \geq b \&\& (b\&(b-1)) \neq 0 \&\& ((a\%b + b)\& \sim b) < b)$:

  return "NOTHING".

return a mod b.

Running time $\mathcal{O}(1)$!

Solution in O(1)

Avoiding computation of x

if$(b = 0 || a \geq b \&\& (b \& (b-1)) \neq 0 \&\& ((a \% b + b) \& \sim b) < b)$:

    return "NOTHING".

return a mod b.

Running time $\mathcal{O}(1)$!

Solution in O(1)

Avoiding computation of x

if$(b = 0 || a \geq b \&\& (b \& (b-1)) \neq 0 \&\& ((a \% b + b) \& \sim b) < b)$:

return "NOTHING".

return a mod b.

Running time $\mathcal{O}(1)$!

- More details in solution booklet.