NUMBER THEORY REVIEW:

1. A basic property of numbers in any base b ≥ 2: The sum of any three single-digit numbers is at most two digits long.
2. How many digits (k) are needed to represent the number N ≥ 0 in base b? With k digits in base b, there are numbers {N | bk-1 ≤ N < bk and N has k number of digits}
3. Design an algorithm to find the number k of binary digits in the binary representation of a positive decimal integer n.

Analysis:

Since 2k-1 ≤ n < 2k

then log2 2k-1≤ log2 n < log2 2k (log2 2k-1 = k-1)

k-1 ≤ log2 n < k

k-1 ≤ └ log2 n ┘ < k

k ≤ └ log2 n ┘ + 1 < k + 1

1. **Algorithm Binary:** Count the number k of the binary digit in the binary representation of a positive decimal integer n. Running time is O (log2 n)
2. **Algorithm Addition:** If each of x and y is n bits long. Adding two n-bit numbers requires n operations, disregarding at least reading them awritingite down the answer. The sum of x and y is n+1 bits at most. The total running time for the addition algorithm is of the form c0 + c1 n, where c0 and c1 are some constants. It is linear. The running time is O(n). This is an optimal algorithm.
3. **Multiplication:** Left shifting (for multiplication) is a quick way to multiply by the base. Let x and y be both n bits. There are n intermediate rows with lengths of up to 2n bits (take the shifting into account). The total time taken to add up these rows doing two numbers (per two rows) at a time is O (n2)
4. **Integer division:** Likewise, the effect of a right shift (Integer Division for division by 2) is to divide by the base, rounding down if needed. Example: 13/16 = (((13/2)/2)/2)/2. Allows us to shift right four times and pack 0000 on the significant bits to obtain 0000, which is equal to 0. Continue from above, └ 1/2 ┘ = 0
5. **Al Khwarizmi’s Multiplication Algorithm:** y \* x = x + 2 ( x \* y/2) if y is odd or 2 (x \* y/2) if y is even. Al Khwarizmi’s Algorithm resolves the product of two integers as a process of halving one integer by 2, doubling the other integer by 2, and adding this integer if needed.

Example: x = 13, y 38 (13 = 1101, 38 = 100110)

|  |  |  |  |
| --- | --- | --- | --- |
| Y | X |  | 1101\*100110 |
| 100110 | 0000 | Str | 0000 |
| 10011 | 11010 |  | 11010 |
| 1001 | 110100 |  | 110100 |
| 100 | 0000000 | Str | 0000000 |
| 10 | 00000000 | Str | 00000000 |
| 1 | 110100000 |  | 110100000 |
| Sum except 0 | 11010  110100  110100000 |  |  |
| Total | 111101110 |  | 111101110 |
|  | (494) |  | (494) |

1. **Multiplication à la Franҫais** - A recursive algorithm that directly implements this rule (Al Khwarizmi’s Multiplication Algorithm):

**Function multiply (x, y)**

Input: Two n-bit integers x and y, where y ≥ 0

Output: Their product

if (y = 0) then return 0;

z := multiply (x, └ y/2 ┘ );

if (y is even) then return 2z

else return x + 2z;

The function for multiplying two n-bit integers terminates after n recursive calls because y is halved (𝑦 /2) at each call. total time taken is thus O(n2 ).

1. **Multiplication ẚ la Russe (Russian Peasant Method):** Let x and y be positive integers. Compute the product of x and y using x \* y = y/2 ∗ 2x if y is even, (y−1)/2∗2x + x if n is odd. The difference between the Russian Peasant Method and Al Khwarizmi’s algorithm (coded as Multiplication à la Franҫais) is that integer (even) division reduces the value of y by 1 if y is an odd number. The total time taken is thus O(n2 ).

|  |  |  |
| --- | --- | --- |
| y \* x | Al khwarizmi’s | Multiplication ẚ la Russe |
| y is even | (2x \* └ y/2 ┘) | (y/2) \* 2x |
| y is odd | x + 2(x \* └ y/2 ┘) | ((y – 1)/2) \* 2x + x |

1. Iteration version of division:

Chapter 00-05 Reviews

For any nonnegative integer a and any positive integer b, gcd(a, b) = gcd(b, a mod b).

Function Euclid (m, n)

//Compute gcd(m, n) by Euclid’s algorithm

Input: two non-negative m and n, not both zero integers

Output: the greatest common divisor of m and n

if (n == 0)

then return m;

else Euclid(n, m mod n);

* the total running time is 2n\*O(n2)= O(n3)
* The input size is the number of bits it takes to encode the numbers m and n, which are └ log2 m ┘ + 1 and └ log n ┘ + 1, respectively.
* Basic operation: One-bit manipulation in the computation of a remainder.
* For the case 1 ≤ m < n, the worst-case number of recursive calls for input size s, t is W(s, t) ∈ 𝜃 𝑡 .
* K = {max((└ log2 x ┘ + 1) , (└ log y ┘))}

function extended-Euclid(x, y)

Input: Two integers x and y with x ≥ y ≥ 0.

Output: Integers i, j, d such that d = gcd(x, y) and i\*x + j\*y = d.

if (y == 0) then return (1, 0, x);

// 1\*x + 0\*0 = x

else {(i', j', d') = extended-Euclid(y, x mod y);

return (j', i'-└ 𝑥 𝑦 ┘ \* j', d');} // r = i' mod j'.

return (i, j, d) }

* The total running time is 2n\*O(n2)= O(n3)
* The input size is the number of bits it takes to encode the numbers x and y, which are └ log2 x ┘ + 1 and └ log y ┘ + 1, respectively.
* Basic operation: One-bit manipulation in the computation of a remainder.
* For the case 1 ≤ m < n, the worst-case number of recursive calls for input size s, t is W(s, t)
* K = {max((└ log2 x ┘ + 1) , (└ log y ┘))}

Which number (x, y) controls the excecution flow number of time (who is stop)