Assignment 5

Design: Factoring

Linhao Chen
CSE 13S - Fall 2019

Date:11/02/2019

1 Introduction

This program will run a program to print all numbers from 2 to 100,000 and indicates which one is Prime Number or Composite Number. And if the number is a Composite Number, the program will also do prime factorization for them and print out its prime factors. User will have an optional option that indicating the range of numbers. By providing a maximum number n, the program will print the number from 2 to n and do same operations above. Expect to use getopt, bit vectors, Sieve of Eratosthenes, loops, inttypes, etc.

2 Factor

```
Main {
  Set integer variables n
  Read the command lines
  If argc is one, the default maximum number will set to 100,000
  If argc is two, shows error and provide instruction, then exit
  If argc is three
    Switch to case depend on the command-line option
      In case ‘n’:
      Set int variable n equals to parameter
      if (parameter is not a number) print instruction for error and exit
  If argc is more than three, print instruction for error and exit
  Create a bit vector by using parameter n
  Use sieve to set the bit vector
  Do a for loop (i from 2 to maximum n, use 1 as increment)
    Print the number(i)
    Gets the value of No.(i) bit from bit vector
    If the value equals 1
      Print char ‘P’ to indicates it is a prime number
    Else
      Print char ‘C’ to indicates it is a composite number
      Call function prime_factor to do prime factorization by using parameter(i)
  Delete the bit vector to prevent memory leaks
```

```
    Ends
}

Function prime_factor (Parameter i) {
    Set a bool variable(print_check) and its default value will be false
    Do a for loop (o from 2 to i, use 1 as increment)
        If i is divisible by o
            Print number o
            Set print_check to true
            Parameter i divide by number o
    If (Parameter i equals to 1)
        return to main
    Else
        If (have not print yet: print_check is false)
            Print number i and return to main
        Else
            Recall Function prime_factor by using parameter i
}
```

3 Bit Vector

```
bv_create (Parameter bit_len) {
    Allocate a memory for BitVector v
    Length of byte vector equals to (bit_len/8) + 1
    Set length of BitVector v equals to bit_len
    Allocate a memory space for byte vector
    Return BitVector v
}

bv_create (Parameter BitVector *v) {
    Free memory for vector of byte
    Free memory for BitVector v
    Set the pointer to NULL
}

bv_get_len (Parameter BitVector *v) {
    return the length of BitVector v
}

bv_set_bit (Parameter BitVector *v, number i) {
    Block equals to i divide by 8
    Index equals to remainder of i divide by 8
    Set the bit to ByteVector [Block]'s index bit to 1
    (e.g. if i is 9, then Block is 1 and Index is 1, so the location will be vector [1]'s second bit)
    (if vector [1] in bit is 00100000, then we should use OR operation with 01000000 to set,
    And then it will be 01100000)
}

bv_clr_bit (Parameter BitVector *v, number i) {
    Block equals to i divide by 8
    Index equals to remainder of i divide by 8
    Set the bit to ByteVector [Block]'s index bit to 0
    (e.g. if i is 8, then Block is 1 and Index is 0, so the location will be vector [1]'s first bit)
    (if vector [1] in bit is 10100000, then we should use AND and NOT operation to clear
    e.g. 10100000 & ~(10000000) = 10100000 & 01111111 = 00100000)
}

bv_get_bit (Parameter BitVector *v, number i) {
    Block equals to i divide by 8
    Index equals to remainder of i divide by 8
    Get the value of specific bit by using short division
    (e.g. if i is 7, then Block is 0 and Index is 7, so the location will be vector [0]'s last bit)
    Pseudocode for short division (by using i equals 7):
        Temp equals to ByteVector [0], 00110101 in bit, and the value should be 53
        Do a for loop (o from 1 to index, use 1 as increment)
            Temp equals temp divide by 2
        /* (Version 1.2: Delete this part)
            If (temp equals 1)
                Exit the loop
        */ (Reason: sometimes the value in temp is 1 but may still needs to calculate again)
        Return the value of the remainder of temp divide by 2
}
    Diagram to explain the short division:
        2 |_53_        1
        2 |_26_        1
        2 |_13_        0
        2 |__6_        1
        2 |__3_        0
        2 |__1_        1      ← return 1

bv_set_all_bit (Parameter BitVector *v) {
    Do a for loop (i from 0 to length to bit vector, use 1 as increment)
        Call bv_set_bit to set bit by using parameter i
}
```

4 Sieve

The code of this part comes from Assignment 5 PDF created by Professor. Long

```
sieve (Parameter *v) {
    set all bits by calling bv_set_all_bits(v)
    clear zero and one for exclude by calling bv_clr_bit
    set bit 2 since it is a prime by calling bv_set_bit
    Do a for loop (i from 2 to sqrt of bit length of vector v, use 1 as increment)
        If (bit(i) not equals to 0) {
            Do a for loop (k from 0 until (k + i) * i bigger than bit length of vector v, use 1 as
            increment)
                Clear all Multiples of k to zero by calling bv_clr_bit (v, (k + i) * i)
    Return to main
}
```