Jaden Liu

Zliu259@ucsc.edu

11/7/2020

CSE13s Fall 2020

Assignment4: Bit Vectors and Primes

Design document

Finding prime is easy to come up with its definition: iterate from 2 to itself to find whether it have any factors than 1 and itself. The execution time is O(n) for the basic method. Here is a new method, sieve, to generate the prime number. And its time is O(logn), which is faster than the original one.

Pre-lab Part 1

1. Assuming you have a list of primes to consult, write pseudo-code to determine if a number is a Fibonacci prime, a Lucas prime, and/or a Mersenne prime.
2. Assuming you have a list of primes to consult, write pseudo-code to determine if a number in base 10 is a palindrome. Note that the technique is the same in any base.

1. I answer it on page 6.
2. I answer it on both page 8 and 9.

Pre-lab Part 2

1. Implement each BitVector ADT function.
2. Explain how you avoid memory leaks when you free allocated memory for your BitVector ADT.
3. While the algorithm in sieve() is correct, it has room for improvement. What change would you make to the code in sieve() to improve the runtime?

1. I did it on page 3.
2. Use dynamic memory allocation. bitvector *b = (bitvector *)malloc(sizeof(bitvector));
And remember to destroy the bitvector after used.
3. When delete the multiples of 2 or 3, I have also deleted 6, 12 these numbers, but in sieve() it is calculated repeatedly. So create a list of prime, then deleted from the least prime dynamically.

Main function in sequence.c:

Cited from asgn4.pdf

1. -s : Print out all primes and identify whether or not they are interesting (Lucas, Mersenne, Fibonacci).

2. -p : Print out palindromic primes in bases 2, 9, 10, and first letter of your last name + 10. Your personal base must be last and match the formatting of the rest.

3. -n : Specifies the largest value to consider, inclusively, for your prime sieve. By default your program runs up through 1000.

```
int l_val = 1000;
int print_option1, print_option2 = 0;
int main(){
      while (c=getopt(etc etc)){
      switch (c){
      case n:
            l_val = atoi(argv[optind]);
      case s:
            print_option1 = 1;

      case r:
            print_option2 = 1;
      }
      }
}
```

Bitvector:

It vector is used to store the result. Each vector has 8 bit, and each bit stands for a result of number whether it is prime. For example, the first 16bit should be stored like this: vector [0]: 00110101, vector [1]: 00010100. Which means 2,3,5,7,11,13 is prime number. 1 means it is , while 0 means it is not.

```
Typedef struct bitvector{
    *vector
    length
}bitvector

BitVector * bv_create ( uint32_t bit_len );
bitvector *b = (bitvector *)malloc(sizeof(bitvector));
b->length = bit_len;
b->vector = (uint8_t *)malloc(sizeof(uint8_t) * bit_len/8);

void bv_delete ( BitVector *v) ;
free(v->vector);
free(v);

uint32_t bv_get_len ( BitVector *v);
return v->length

void bv_set_bit ( BitVector *v, uint32_t i);
v->vector[i/8] |= 1<<(1%8);

void bv_clr_bit ( BitVector *v, uint32_t i);
v->vector[i/8] &= ~(1<<(1%8));

uint8_t bv_get_bit ( BitVector *v, uint32_t i) ;
if(v->vector[i/8] & 1<<(1%8))
return 1;
else
return 0;
```

```
void bv_set_all_bits ( BitVector *v);
for(i=0;i<v.length;i++){
        bv_set_bit(v,i);
}
```

Sieve:

The sieve is just to delete the numbers that can be divided by 2,3,4,5 iteratively, until there is nothing you can delete from the list.

Cited from asgn4.pdf

```
//
// The Sieve of Erastothenes
// Sets bits in a BitVector representing prime numbers.
// Composite numbers are sieved out by clearing bits.
//
// v: The BitVector to sieve.
//
void sieve(BitVector *v) {
  bv_set_all_bits(v);
  bv_clr_bit(v, 0);
  bv_clr_bit(v, 1);
  bv_set_bit(v, 2);
  for (uint32_t i = 2; i < sqrtl(bv_get_len(v)); i += 1) {
    // Prime means bit is set
    if (bv_get_bit(v, i)) {

      for (uint32_t k = 0; (k + i) * i <= bv_get_len(v); k += 1) {
        bv_clr_bit(v, (k + i) * i);
      }
    }
  }
  return;
}
```

Interesting Primes:

There are three types of interesting prime: Fibonacci, Lucas and Mersenne prime.

Fibonacci

```
Int Fibonacci(int n):
    If n==1
        Return 0;
    If n==2
        Return 1
    If n>2
        Return Fibonacci(n-1) + Fibonacci(n-1)

Int isFibonacci(int m):
    For (int I =0; Fibonacci(i)<m; i++)
        If next_fibo == m
            Return 1
    Return 0
```

Lucas

```
Int lucas(int n):
    If n==1
        Return 2;
    If n==2
        Return 1
    If n>2
        Return lucas(n-1) + lucas(n-1)

Int islucas(int m):
    If m == 1 or m==2
        Return 1
    For (int I =0; lucas(i)<m; i++)
        If next_fibo == m
            Return 1
    Return 0
```

Mersenne:

```
Int ismersenne(int n)
    If p is odd
        P+=1
While p>1 and p % 2 == 0
    p/=2
```

```
if p==1
    return 1
return 0
```

Palindrome:

Cited from asgn4.pdf

```
def isPalindrome(s):
    f = True
    for i in range(len(s) / 2):
        if s[i] != s[-(i + 1)]:
            f = False
    return f

w = raw_input("word = ")
if isPalindrome(w):
    print w, "is a palindrome"
else:
    print w, "is not a palindrome"
```

Change base(example from 10 to 22):

```
Char base22[32];
Int i = 1;
Char *d_to_22(int n)
    If n<22
        If n<10
            Base[i] = n + '0'
        Else
            Base[i] = n + 'a' -10
    Else
        d_to_22(n/22)
        i++;
        n %= 22
        If n<10
            Base[i] = n + '0'
        Else
            Base[i] = n + 'a' -10
```

Design Process:

1. I met the first problem is that I messed up the length of the bitvector, I thought it is vectors' length instead of length of bits.
2. Second I got the variables problem when I do recursion, first I do local variable, but it did not satisfy my function. Then I change to static local variable, then it works. But only for the first time. Then I have to use global variable.

All in all, I learned how to do prime more efficiently and how to do base change and determine palindrome in C programing.