ALI SALEHI
ASGN 4

BIT VECTORS + PRIMES

PRIME NUMBERS

- EVENLY DIVISIBLE BY 1 AND ITSELF

PRIME NUMBER THEOREM: FOR $m \in \mathbb{N}$, $1 < m < N$ PROBABILITY OF BEING PRIME IS CLOSE TO $1/\ln(N)$ ALSO

$$\frac{N}{\ln N - (1-\epsilon)} < \pi(N) < \frac{N}{\ln N - (1+\epsilon)}$$

- EVALUATING/TESTING FACTORS FROM $2 \to N = O(n)$
- EVALUATING/TESTING FACTORS FROM $2 \to \sqrt{n} = O(\sqrt{n})$
  - MUCH FASTER + EQUALLY VIABLE
- NON PRIME NATURAL NUMBERS = COMPOSITE
- EVERY INTEGER $m > 1$ IS PRIME OR A UNIQUE PRODUCT OF PRIMES:

$$m = p_0^{\alpha 0} \times p_1^{\alpha 1} \times \dots \times p_n^{\alpha n} = \prod_{i=0}^{\kappa} p_i^{\alpha K}$$

INTERESTING PRIMES

- FIBONACCI: $F_n = F_{n-1} + F_{n-2}$ $(F_0 = 0)$
- LUCAS NUMBER: $L_N = L_{N-1} + L_{N-2}$ $(L_0 = 2)$ $(L_1 = 1)$
- MERSENNE PRIME: $M_n = 2^n - 1$ (WHERE M_n IS PRIME)
- PALINDROMIC PRIMES: A PRIME NUMBER WHOSE CORRESPONDING STRING AT A CERTAIN BASE IS A PALINDROME.
  - PALINDROME = SAME FORWARDS AND BACKWARDS

PRE LAB PART 1

1) ASSUMING YOU HAVE A LIST OF PRIMES, WRITE PSUEDOCODE TO DETERMINE IF A # IS LUCAS, MERSENNE, OR FIBONACCI PRIME.

- fib, fib1, fib2; (variables initialized to hold fibonacci value and two previous fib #'s)

- lucas, lucas1, lucas2; (variables that hold current and two previous lucas values)

- mers, n; (variables that hold current mersenne value and exponent to use)

for (i = 0, i < n, i++) {

- if (bit[i] is set): print it is prime

- if (mers == i): if bit[i] is set print is mersenne, else update to following mersenne #

- if (lucas == i): if bit[i] is set print is lucas, else update to following lucas #

- if (fib == i): if bit[i] is set print is fibonacci, else update to following fibonacci #

2) ASSUMING YOU HAVE A LIST OF PRIMES, WRITE PSUEDOCODE TO DETERMINE IF A NUMBER IN BASE 10 IS A PRIME.

given sieve() for bit vectors sets all prime bits:

for (i = 2; i <= n (inclusive); ++i) {

- if bitvector[i] / bv-get_bit(bv, i):
  - # is prime!
- else:
  - # is composite

BV.C

* bv_create(uint32_t bit_len):
  - initialize "head" just like stack
  - initialize vector to div algo (n/8+1) uint32s
  - initialize length to number of bits (n-1) * 8 ?
  - use other edge case checks from stack create
  - return pointer to bitvector

bv_delete(BV *v):
  - free(v->vector)
  - free(v)

bv_get_len(BV *v):
  - return v->length (OPAQUENESS)

bv_set_bit(BV *v):
  - OR the bit vector with 00...1..00 @ location you want to set

bv_clr_bit(BV *v):
  - AND the bit vector with 11...0..11 @ location you want to clear - INVERT!

bv_get_bit(BV *v, uint32 i):
  - locate correct bit (->vector[i/8])
  - AND WITH 00...1...0 (CLEARS ALL BITS BUT LEAVES DESIGNATED AS IS)
  - RIGHT SHIFT SO DESIGNATED BIT -> LEAST SIGNIFICANT BIT

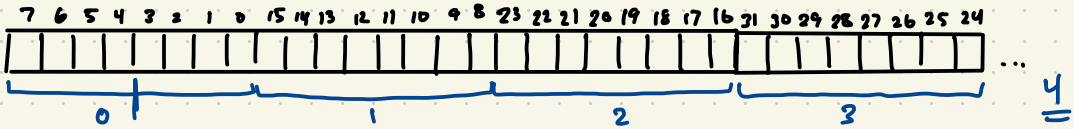
set_all_bits():
  - SET EACH uint_8 TO 0xFF

* MASK CAN BE REPRESENTED BY 1 << K (k^th bit) of
  - USE ~ TO REVERSE BITS FOR 11...0...11 FOR CLR_BIT

SIEVE

- USED SIEVE OF ERASTOTHENES FROM LAB DOC

VISUAL REPRESENTATION OF BV



- EACH BYTE IS REPRESENTED IN BINARY THEREFORE EACH BIT IS A 1 OR 0. 1 = PRIME, 0 = COMPOSITE

PRE LAB PT. 2

1) IMPLEMENT EACH bv.c FUNCTION:

   - VIEW PAGE ABOVE FOR PLANNING FOR bv.c

2) EXPLAIN HOW YOU AVOID MEMORY LEAKS WHEN YOU FREE ALLOCATED MEMORY FOR YOUR BITVECTOR ADT.

   - WHEN DYNAMICALLY CREATING MEMORY, IT IS THE RESPONSIBILITY OF THE CALLING FUNCTION TO KEEP TRACK OF AND HENCE FREE THE ALLOCATED MEMORY AFTER USE

   IN MAIN, AFTER INITIALIZING A BITVECTOR, I USE IT TO PERFORM THE NECESSARY OPERATIONS AND THEN CALL bv_delete WHEN I'M DONE. bv_delete FREES THE POINTER TO THE BitVECTOR STRUCT, AND TO THE VECTOR ARRAY ITSELF. BECAUSE THE VECTOR ARRAY IS INITIALIZED INSIDE THE STRUCT, WE MUST BE SURE TO FREE THAT FIRST IN bv_delete.

3) HOW WOULD YOU IMPROVE SIEVE()?

   - SOME SMALL BUT CONSIDERABLE TIME IMPROVEMENTS COULD COME FROM LOOPING ONLY THROUGH ODD VALUES AND CHANGING i++ -> i += 2

WITH A SIEVE/BITVECTOR COMBINATION THAT AUTOMATICALLY SETS ALL EVEN BITS TO ZERO, SO THEIR FACTORS DO NOT NEED TO BE CALCULATED. IN A SIMILAR THOUGHT PROCESS THE SIEVE COULD BE CHANGED TO FORGO CALCULATING FACTORS OF 2 AND 3 BY ONLY CONSIDERING NUMBERS THAT MODULO 6 RESULT IN A 1 OR A 5.

PRINTING PRIMES AND CHECKING

LOOP FROM 0 - N :

- IF BIT AT THAT LOCATION IS SET: PRINT THAT IT IS A PRIME

- FIBONACCI:
  - KEEP TRACK OF PREV #, PREV-PREV #, AND CURENT #
    - IF CURRENT # < i IN LOOP DO NOTHING
    - IF CURRENT # = i IN LOOP
      - IF i^{th} BIT IS SET PRINT # = FIB
      - CALCULATE NEXT FIB INTO CURRENT FIB AND UPDATE PREVIOUS VALUES
- LUCAS NUMBERS:
  - SAME PROCESS AS FIBONACCI BUT HAS ITS OWN VARIABLES AND STARTS AT DIFFERENT VALUES

- MERSENNE PRIME:
  - MERS # = $2^i - 1$
    - IF MERS # = i
      - PRINT # = MERS IF BIT IS SET
    - RECALCULATE MERSE #

BASE CHANGE + PALINDROMES

BASE_CHANGE():

- ALLOCATE MEMORY FOR LARGEST POSSIBLE # OF DIGITS FOR BIT VECTOR LENGTH
  - USED (num >>= 1) indx++; TO SIMULATE $\log_2(N)$ BECAUSE BASE 2 HAS THE MOST POTENTIAL DIGITS.

  * DON'T FORGET TO FREE MEMORY WHEN YOU'RE DONE USING *
- USE STANDARD CHANGE OF BASE APPROACH FOR STORING REMAINDERS IN NEW STRING AND CONTINOUSLY DIVIDING QUOTIENT
  - ALTER REMAINDER TO REPRESENT ASCII VALUE OF DESIRED DIGIT (INCLUDING LETTERS)
- REVERSE STRING
- RETURN STRING

IS_PALINDROME():

- SLIGHTLY ALTERED FROM LAB DOC. CREATE COPY OF PASSED STRING AND REVERSE IT.
- LOOP THROUGH CHARACTERS OF STR AND REVERSED_STR
  - IF ALL THE SAME RETURN TRUE: ELSE FALSE.

PALINDROMES():

- LOOP (i=2, i<=n, i++)
  - STORE BASE_CHANGE(i) IN A CHAR * STR
  - CHECK IF i IS PRIME AND STR IS A PALINDROME
    - IF SO PRINT PALINDROME OUTPUT
  - FREE(STR) !!!
- REPEAT FOR BASE 2, 9, 10, 29 (10 +'S')

SOURCES

- SEVERAL PIAZZA POSTS AND SUGGESTIONS FROM SECTION
- STACK.C FROM ASGN 3 FOR BV.C
- LAB DOCUMENT FOR SIEVE() AND PALINDROME
- STACK OVERFLOW "HOW TO DO AN INTEGER $\log_2$ IN C++"

  https://stackoverflow.com/questions/994593/how-to-do-an-integer-log2-in-c

- TUTORIALS POINT "C - STRINGS"

  https://www.tutorialspoint.com/cprogramming/c_strings.htm

- BASE CONVERSION GEEKS FOR GEEKS
  - USED FOR HELPING WITH ADDING CHARACTERS FOR BASES ABOVE 10 + REVERSING STRINGS

  https://www.geeksforgeeks.org/convert-base-decimal-vice-versa/