Problem-set 01

Mostafa Touny

Contents

Lectures: Sipser 1: Ch. 0.2. 2: Ch. 3.1, 3.3.

Ex. 1

skipped

Ex. 2

a

As the recipe is deterministic, i.e generates only one output given the same input, The assignment is valid and the function is well-defined.

For injectivity, we prove the contrapositive; Namely, if $x_0 \neq x_1 \to f(x_0) \neq f(x_1)$. Since, $x_0 + 1 \neq x_1 + 1$, Their binary representation differs in either the number of bits or in some bit not matching. That unmatching bit cannot be the most significant bit as it's always 1. Therefore, resulting strings, $f(x_0)$ and $f(x_1)$ are not the same.

For surjectivity, Pick-up any string $w \in \{0,1\}^*$ and reverse the recipe to obtain natural number x. Namely, add 1 as the most significant bit, interpret string as a binary number, convert to base 10, and finally subtract 1. Clearly $f(x) = w$.

b

Here's a very illustrative example that achieves an encoding in $\lg a + \lg b$.

(2, 5), converted to binary, (10, 101); add leading zeros so both have the same number of bits, (010, 101). Finally, Concatenate bit by bit, i.e add the first bit of the first number then first bit of the second then second bit of the first, ..etc, Yielding 011001.

The recipe can easily be rolled back.

Ex. 3

a

Binary search on range (a, a+1, .., b-1, b), where at each step algorithm D is queries on both the first and second halves of the array, Then recursively call the binary search on both halves.

```
primeFactor( X = array(a, a+1, .., b-1, b) )

  if X.length == 1
    if D(x)
      return X[0]
```

```
    return FALSE

  if X.length == 0
    return FALSE

  halfIndex = floor(X.length/2)
  firstHalf = X[: halfIndex]
  secondHalf = X[halfIndex+1 :]

  if D(firstHalf)
    return primeFactor(firstHalf)
  if D(secondHalf)
    return primeFactor(secondHalf)
  return FALSE

main( X = array (a, a+1, .., b-1, b) )
  res = primeFactor(X)
  if res == FALSE
    print 'no'
  else
    print 'yes'
```

Since $m \geq$ length of X, Complexity is $O(\lg m)$.

b

Solving decision $D by f. Compute $f(x) = y$ and check whether $y_i = b$.

```
y = f(x)
if y[i] == b
  return YES
return FALSE
```

Computing f by $D. on each bit x_i of x, Call D on string x, bit 0, and position i. if result is YES, let $y_i = 0; if result is NO, let $y_i = 1$. assign $f(x)$ to y.

```
y = []
for i in x.length
  res = D(x, 0, i)
  if res == YES
    y[i] = 0
  else
    y[i] = 1
return y
```

Ex. 4

a

Trivially we can transform the input to the form *input#.

Initially the machine is on state q_0.

q_0 : move right until # is reached, then q_1.

q_1 : move left until a non-# is reached, then q_2.

q_2 : if * halt; if 0 print # and q_3; if 1 print $ and q_4.

q_3 : move right until a blank space is reached, then print 0 and q_5.

q_4 : move right until a blank space is reached, then print 1 and q_5.

q_5 : move left until # is reached, then q_1.

b

skipped

c

Assume andrew id is ac12. f(ac12) = 10000110.

*10000110# is our assumed input. on step 8, machine is on 0 and state q_0. On step 9, machine is on # and state q_0. On step 10, machine is on state q_1