CSE 21 HW 7

Brian Masse

March 5, 2025

1. A slow-growing sequence of length n (with n $\geq$ 1) is a non-decreasing sequence of integers that start with 1 and each pair of entries differ by at most 1.

a) (for n $\geq$ 1), How many slow-growing sequence of length n are there?

Creating a slow growing sequence of length n from a slow growing sequence of length n - 1, where a_n is the last element in the n - 1 sequence, there are two two options: a_n and $a_n + 1$. Thus,

$$A(n) = 2 \cdot A(n-1), A(1) = 1$$

$$\begin{array}{rl} 1) & A(n) = 2 \cdot A(n-1) \\ 2) & A(n) = 2^2 \cdot A(n-1) \\ & \vdots \\ k) & A(n) = 2^k \cdot A(n-1) \\ & \vdots \\ n-1) & A(n) = 2^{n-1} \end{array}$$

So, there are 2^{n-1} slow-growing sequences of length n.

b) How many bits would the most efficient fixed-length encoding of sequenes use?

$$\begin{aligned} n &= \lceil log_2(2^{n-1}) \rceil \\ &= n-1 \end{aligned}$$

c) Develop your own encoding / decoding algorithm where the code uses this number of bits

Create an encoding such that each bit in the encoded binary string represents whether to add 1 or 0 to the previous number in the string.

d) Use your encoding to encode the follow slow-growing sequences

- *(1,2,3,3,3,4,4,5)* = 1100101
- *(1,1,1,2,2,3,4,4)* = 0010110
- *(1,2,2,3,3,4,5,6)* = 1010111

e) Use your decoding to decode the following strings

- 01010100 = (1,1,2,2,3,3,4,4,4)
- 11100011 = (1,2,3,4,4,4,4,5,6)
- 10111110 = (1,2,2,3,4,5,6,7,7)

2. Image files can be encoded using binary strings. In the most simple version, you can encode an nxm black and white image using nm bits with black corresponding to 0 and white corresopnding to 1.

We can use hexadecimal to encode each of the chunks of 4 pixels into a single hexadecimal character.

a) How many bits are required to encode this image by encoding each pixel with 1 bit?

$$nm = (96)(96) = 9216$$

b) How many hexadecimal characters are needed for this image?

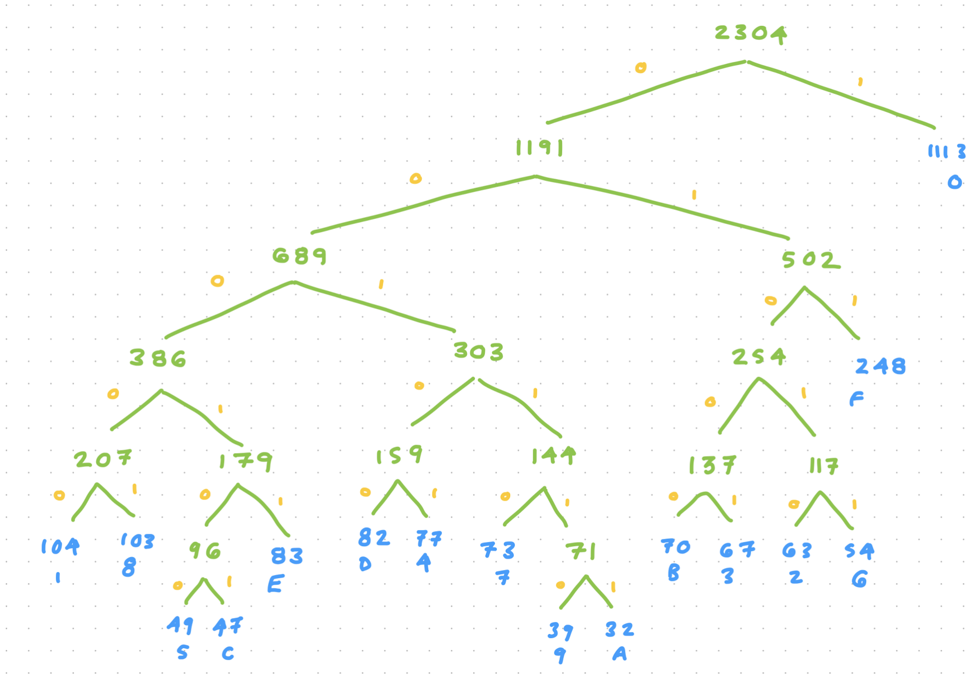
$$\left(\frac{96}{4}\right)(96) = 2304$$

- $\frac{96}{4}$: Groups per row
- 96 : Number of Rows

c) Huffman encoding is actually used in image compression. What we can do is compute the frequency table of the hexadecimal chracters and build a Huffman code based on that. Then encode the hexadecimal string using the Huffman code. Here is the frequncy table for this particular image:

Character	*0*	*1*	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*	*A*	*B*	*C*	*D*	*E*	*F*
Weight	*1113*	*104*	*63*	*67*	*77*	*49*	*54*	*73*	*103*	*39*	*32*	*70*	*47*	*82*	*83*	*248*

- *Draw the huffman tree for the set of frequencies.*



- *Give the code for each character*

Number	*Code*
0	1
1	00000
2	01010
3	01001
4	00101
5	000100
6	01011
7	00110
8	00001
9	001110
A	001111
B	01000
C	000101
D	00100
E	00011
F	011

- *Calculate the total number of bits needed to encode this particular image of the moon using this coding.*

$$
\begin{aligned}
T(n) = 1113(1) + 104(5) + 63(5) + 67(5) + 77(5) + 49(6) + 54(5) + 73(5) + 103(5) \\
+39(6) + 32(6) + 70(5) + 47(6) + 82(5) + 83(5) + 248(3) \\
= 6739
\end{aligned}
$$

3. Consider the set of 26 capital letters of the Roman Alphabet

a) Using a fixed length character-by-character encoding, what is the minimum number of bits required to encode each character?

$$n = \lceil log_2(26) \rceil = 5$$

b) Develop a fixed length character-by-character encoding for this alphabet using the number of bits from the previous part and use it to encode / decode the following strings

Assign each letter in the alphabet a number, starting with A=1 and going to Z=26. Then, for each character in a string, encode its corresponding value with 5 bits.

- *Encode: "MATH"* $\implies$ (13)(1)(20)(8) $\implies$ *01101 00001 10100 01000*
- *Encode: "BYTE"* $\implies$ (2)(25)(20)(5) $\implies$ *00010 11001 10100 00101*
- *Decode: 10010 01110 10001 10011* $\implies$ (18)(14)(17)(19) $\implies$ *RNQS*
- *Decode: 10001 01110 01110 10011* $\implies$ (17)(14)(14)(19) $\implies$ *QNNS*

c) Using fixed length encoding. What is the minimum number of bits required to encode each 4 letter string over this alphabet.

$$n = \lceil log_2 26^4 \rceil = 19$$

d) Develop a fixed length encoding for 4 letter strings using the number of bits from the previous part and use it to encode / decode the following strings:

Assign each letter in the alphabet a number, starting with A=1 and going to Z=26.

To encode, convert the 4-character string into a base-26 number. (Replace each letter with its corresponding number, then multiply it by a power of 26 depending on its position in the string). Convert this number into base 10. Encode the result.

To decode, convert the binary string into a base 10 number, and that into a base 26 number. The coefficients in front of the powers of 26 correspond to the characters in the string.

- *Encode: "MATH"* $\implies (13 \cdot 26^3 + 1 \cdot 26^2 + 20 \cdot 26 + 8)_{26} \implies 229692$
$\implies$ *011 1000 0001 0011 1100*

- *Encode: "BYTE"* $\implies (2 \cdot 26^3 + 25 \cdot 26^2 + 20 \cdot 26 + 5)_{26} \implies 52577$
$\implies$ *000 1100 1101 0110 0001*

- *Decode: 100 1111 1010 1001 0101* $\implies (326293)_{10}$
$\implies (18 \cdot 26^3 + 14 \cdot 26^2 + 17 \cdot 26 + 19)_{26} = RNQS$

- *Decode: 100 1011 0101 1001 1111* $\implies (308639)_{10}$
$\implies (17 \cdot 26^3 + 14 \cdot 26^2 + 14 \cdot 26 + 19)_{26} = QNNS$

4. Suppose you are traveling from the bottom left corner of a 9 by 6 grid of city blocks and you wish to get to the top right corner only using up and right movements

a) What is the minimum number of bits necessary to encode these paths with a fixed length encoding?

There are $\binom{6+9}{6} = \binom{15}{6} = 5005$ number of paths from the bottom left to the top right.

$$n = \lceil log_2 5005 \rceil = 13$$

b) Develop an encoding strategy and encode the path given in the example

Using the ranking / unranking algorithm proposed in class can store all possible path sequences in 13 bits.

First translate the path into a fixed length binary string of length 15 with 6 1s, where 1 represents a movement to the right. Instead of encoding this 15-bit string, list out each path in lexicographic order, and assign each an index. Encode this index.

To determine the index from a 15 bit binary string, use the ranking algorithm.

$$\text{rank: Path} \rightarrow \text{index}$$

$$Rank(x,n,k) = \begin{cases} Rank(x', n-1, k) & \text{if x starts with 0} \\ Rank(x', n-1, k) + \binom{n-1}{k} & \text{if x starts with 1} \end{cases}$$

Where x' is the current path minus the first character.

To decode this 15 bit binary string, use the deranking algorithm.

$$\text{Derank: Index} \rightarrow \text{Path}$$

$$Derank(x,n,k) = \begin{cases} 0 \circ Derank(d, n-1, k) & \text{if d} < \binom{n-1}{k} \\ 1 \circ Derank(d - \binom{n-1}{k}, n-1, k-1) & \text{else} \end{cases}$$

Using this algorithm to encode the example:

$$Path \implies 111\ 1000\ 0100\ 0100, n = 15, k = 6$$

$$\begin{aligned} &\implies \binom{14}{6} + \binom{13}{5} + \binom{12}{4} + \binom{11}{3} + \binom{6}{2} + \binom{2}{1} \\ &= (4967)_{10} \\ &= 1\ 0011\ 0110\ 0111 \end{aligned}$$

c) Use your encoding strategy to decode the following string: 0 1010 1100 1100

$$(0\ 1010\ 1100\ 1100)_2 = (2764)_{10}$$

Using the Derank Algorithm:

$$Derank(2764, 15, 6) = \ 011\ 0100\ 1010\ 0010$$

Position	*d*	*n*	*k*	*Decoded Binary Value*
1	2764	15	6	0
2	2764	14	6	1
3	1084	13	5	1
4	256	12	4	0
5	256	11	4	1
6	46	10	3	0
7	46	9	3	0
8	46	8	3	1
9	11	7	2	0
10	11	6	2	1
11	1	5	1	0
12	1	4	1	0
13	1	3	1	0
14	1	2	1	1
15	0	1	0	0

Translated into a path