CS270: LAB #18

Gray Code

You may work in teams of ideally one or two people (three is acceptable in the event of an unscheduled absence). Unless stated otherwise, the lab is due to be submitted into Gradescope at the end of the class day (11:59pm)
In order to receive credit, follow these instructions:

[a] Every team member should be discussing simultaneously the same problem – do NOT try to divvy up the labor and assign different problems to different students since the material is cumulative.

[b] Directly edit this lab PDF using Sedja/PDFescape with your answers (extra pages can be added in the rare event you need more than the allotted space)

[c] Each lab, rotate which member has the responsibility of being the Scribe. This is the person that is typing the answers and uploading the final PDF – note that only a single copy of the filled in PDF is turned into Gradescope. Only one lab needs to be submitted for the entire team, and all members receive the same score. Make sure to use a font that your PDF editor is compatible with (otherwise you might find your answers appear as weird shapes/sizes or simply disappear entirely!)

[d] The Gradescope submission must have each answer properly tagged with the appropriate question. Moreover, every member of the team must be listed as a submitter. Although it is the Scribe which executes these actions, it is still the responsibility of the entire team to make certain this is done properly (thus it is highly recommended that the Scribe share their screen so the entire team can witness it). Answers which are improperly tagged cannot be seen by the grader and thus cannot be scored.

[e] **FOR REMOTE ONLY**: Each lab, rotate which member has the responsibility of being the Recorder. This is the person who hits the Zoom Record button (once the technical permission is granted by the TA/RCF/Professor) and ensures that everyone has their camera/microphone on. They are also the member that is responsible to make sure the DrexelStream video is marked as viewable and entered into the https://tinyurl.com/VidLinkForm webform before 11:59pm (they should also email the rest of their team as confirmation.) Note that the video file doesn't get created/processed until after the Recorder has quit Zoom.

[f] Each lab, rotate which member has the responsibility of being the Manager. This is the person that ensures that everyone is participating equally and honestly, keeps the group on task, ensures that all team members understand a solution before going on to the next question, and presses the "hand up" button in Zoom to summon a TA or the professor (but they only do so after surveying the group to make sure everyone has the same question).

Team Name (CS pioneer): Tim Berners-Lee

Scribe name: Brendan Hoag

Recorder name: Jackson Masterson

Manager name: Cole Bardin

Other team member (if any): Jeremy Matthews

Question 1: 8 points

Gray Codes are an important method for representing sequential values.

Read the following description of Gray Codes and answer the following questions.

`https://www.allaboutcircuits.com/technical-articles/gray-code-basics/`

(a) (1 point) What researcher at Bell Labs submitted the patent Pulse Code Communication?

Frank Gray

(b) (1 points) Are Gray Codes well suited for mathematical operations?

No because they are a numeric representation of a cyclic encoding scheme, where it will roll over and repeat

(c) (1 points) How many bits of the Gray Code sequence change as the number count progresses?

Only one bit in the sequence changes as the count progresses

(d) (1 points) How can summing the bits of a Gray Code Sequence be used for error detection?

The sum of the current number and the next number should only change by 1 each time.
If this does not happen, then there has been an error.

(e) (1 points) What was the problem cased by the standard binary system going from 7 to 8?

The issue with the standard binary system going from 7 to 8 is that all the bits must flip
because we're going from 0111 to 1000, and a read operation during this flip could give bad data.

(f) (1 points) Was the problem with mechanical or digit systems changing the bits?

It was specifically with mechanical systems, where the timing of bit flips may be off somewhat
due to mechanical switching rather than seemingly instant switching of digital systems.

(g) (1 points) Why does a Gray Code reduce the chances of an error when going from 7 to 8?

A gray code reduces this error because the code will be at most 1 digit off because only 1 digit needs to change

(h) (1 points) In the Gray Code image, what do the red and white boxes represent?

The red boxes represent 1's and the white boxes represent 0's.

(i) (1 points) How are Gray Codes used in modern aircraft?

Gray codes are used in the encoders on the altimeters of an aircraft

Question 2 : 6 points

The smallest Gray Code is 1-bit.

$$G_1 = [0, 1]$$

Any larger Gray Code can be created using a recursive formula.

$$G_n = [0G_{n-1}, 1G'_{n-1}]$$

To create G , the following algorithm is followed:

1. Add a 0 to the beginning of all values in the previous Gray Code.
   $0G_{2-1} = 0G_1 = [00, 01]$
2. Reverse the previous Gray Code
   $G'_{2-1} = G'_1 = \text{reverse}([0, 1]) = [1, 0]$
3. Add a 1 to the beginning of the reversed code.
   $1G'_{2-1} = 1G'_1 = [11, 10]$
4. Append the two sequences created together.
   $[0G_{2-1}, 1G'_{2-1}] = [0G_1, 1G'_1] = [00, 01] + [11, 10] = [00, 01, 11, 10]$

Now let's create G 3 by performing those four steps from above.

(a) (1 point) Create $0G_2$

0G2 = 000, 001, 011, 010

(b) (1 point) Create G'_2 (Just reverse the elements of the list, not the individual bits)

G'2 = 10, 11, 01, 00

(c) (1 point) Create $1G'_2$

1G'2 = 110, 111, 101, 100

(d) (1 point) Create G_3

G3 = 000, 001, 011, 010, 110, 111, 101, 100

(e) (2 points) Verify you have created a Gray Code. Draw an arrow between each pair of binary values to show which bit changes. (if you do not have a drawing tool in your pdf editor, you can use color coding)
Remember: Gray Codes are Cycles, so remember to draw a line from the last element to the first.

G2 → 0G2
00 → 000
01 → 001
11 → 011
10 → 010

G'2 → 1G'2
10 → 110
11 → 111
01 → 101
00 → 100

G3 = 0G2, 1G'2

Question 3 : 25 points

(a) (15 points) Develop a Racket Function that takes a value and adds it to the front of every element in a list. Include a screenshot of your implementation. Give both a recursive version and non-recursive version

```
(define (prepend x L) ...)
```

Make sure your code passes all the below tests.

```
(equal? (prepend 0 '())
        '())
(equal? (prepend 0 '((0) (1)))
        '((0 0) (0 1)))
(equal? (prepend 0 '((0 0) (0 1) (1 1) (1 0)))
        '((0 0 0) (0 0 1) (0 1 1) (0 1 0)))
(equal? (prepend 1 '((0) (1)))
        '((1 0) (1 1)))
(equal? (prepend 1 '((0 0) (0 1) (1 1) (1 0)))
        '((1 0 0) (1 0 1) (1 1 1) (1 1 0)))
```

```
; non recursive
(define (prepend x L)
  (map (λ (l) (cons x l)) L))

; recursive
(define (prependR x L)
  (if (null? L) null (cons (cons x (first L)) (prependR x (rest L)))))
```

(b) (10 points) Develop a Racket Function to make Gray Codes and include a screenshot below

```
(define (gray_code n) ...)
```

Make sure your code passes all the below tests.

```
(equal? (gray_code 1)
        '((0) (1)) )
(equal? (gray_code 2)
        '((0 0) (0 1) (1 1) (1 0)))
(equal? (gray_code 3)
        '((0 0 0) (0 0 1) (0 1 1) (0 1 0)
         (1 1 0) (1 1 1) (1 0 1) (1 0 0)))
```

```
(define (prepend x L)
  (map (lambda (l) (cons x l)) L)
)

(define (prependR x L)
  (if (null? L) null (cons (cons x (first L)) (prependR x (rest L))))
)

(define (gray_code n)
  (cond
      [(< n 1) '()]
      [(equal? n 1) '((0) (1))]
      [else (append (prepend 0 (gray_code (- n 1))) (prepend 1 (reverse (gray_code (- n 1)))))]
  )
)
```

Question 4 : 10 points

Two steps in creating a Gray Code require prepending a value to a Gray Code.

Assume that $G_n = [g_0, g_1, \cdots, g_x]$ is a Gray Code.

The first step to making G_n is prepending a 0 to each value.

$$0G_n = [0g_0, 0g_1, \cdots, 0g_x]$$

(a) (2 points) Explain (in plain English) why only one bits changes between $0g_y$ and $0g_{y+1}$ for all $0 \leq y < n$.

Only one bit changes because it is a gray code, and that is how it is defined. By prepending a 0 to each code, we have not changed that property, since all elements had only one bit different before the 0 was prepended. 01 and 11 are off by only one bit, as are 001 and 011, the only difference being the prepended 0.

(b) (2 points) A Gray Code must cycle. Explain why only one bit changes between $0g_x$ and $0g_0$.

Only one bit changes between 0gx and 0g0 because only one bit changes at each n in Gn, so the difference between G0 and Gx is only one. This means that the difference between the first and last term is also the same.
This property will also hold when prepending the 0's (in the case of 0G0 and 0Gx).

(c) (2 points) Do the same arguments hold true if a 1 is prepended instead of 0? Why?

It does not hold true because the difference between gray codes is the 1's, so prepending 1s would change the code.

(d) (4 points) **Theorem 1**:

Prepending ___a 0___ to a Gray Code keeps the property that only ___1 bit___ changes between elements.

Question 5 : 12 points

Two steps in creating a Gray Code requires reversing all elements in a Gray Code.

Assume that $G_n = [g_0, g_1, \cdots, g_x]$ is a Gray Code.

$$G'_n = [g_x, g_{x-1}, \cdots, g_0]$$

(a) (4 points) Explain (in plain English) why only one bits changes between g_y and g_{y-1} for all $0 < y \leq n$.

Only one bit changes between gy and g(y-1) when inversing,
because inverting a gray code results in a gray code, by definition.
so the same rules apply to the inverted grey code.

(b) (4 points) A Gray Code must cycle. Explain why only one bit changes between the first and last element.

When inverting/reflecting a gray code, the order of the elements is preserved,
but reflected such that the first element becomes the last and vice versa.
Therefore, the same cycling logic for gray codes holds true for the reflected gray codes.

(c) (4 points) **Theorem 2**:

Reversing a Gray Code keeps the property that only one bit changes between elements.

Question 6 : 39 points
Prove by Induction that the following recursive formula creates a Gray Code.

To be a Gray Code, the list must satisfy each of the four requirements in the red box here:
You may use Theorems from previous questions.

The 4 requirements to be a GrayCode:

[a] each member is n bits [i.e. 0s and 1s]

[b] there are exactly 2^n members

[c] each value 0 to $2^n - 1$ is represented exactly once (no repeats, no duplicates)

[d] exactly one bit changes from one member to the next (including the wrap around from the last back to the first)

$$G_1 = [0, 1]$$
$$G_n = [0G_{n-1}, 1G'_{n-1}]$$

(a) (3 points) Base Case: Explain why G_1 is a Gray Code

G1 is a gray code because each member is only 1 bit, there are exactly 2 members, no member is repeated, exactly one bit changes between each member

(b) (4 points) **Inductive Hypothesis:** What should be assume about $G_n = [g_1, g_2, \cdots, g_x]$?

we should assume it meets the 4 requirements of gray codes, meaning we assume Gn is a valid gray code.

(c) (8 points) We need to append two lists together to make $G_{n+1} = [0G_x, 1G'_x]$.
Explain why the last element in $0G_{n-1}$ only differs by 1 bit from the first element in $1G'_{n-1}$

We can assume that Gn-1 and G'n-1 by themselves only differ by one bit since they are already valid gray codes.
Appending a 0 to each element of Gn-1 would not affect the property that each element only changes by one bit; likewise with appending a 1 to each element of G'n-1.
The last element of Gn-1 is the same as the first element of G'n-1.
Therefore appending a 0 to Gn-1 and a 1 to G'n-1 would make it so that the last element of 0Gn-1 and first element of 1G'n-1 would differ by one bit.

(d) (8 points) Explain why the first element in $0G_{n-1}$ only differs by 1 bit from the last in $1G'_{n-1}$

Using the same premise assumptions from 6c, the first element in Gn-1 is the same as the last element in G'n-1.
Therefore appending a 0 to Gn-1 and a 1 to G'n-1 would cause the first element in 0Gn-1 and the last element in 1G'n-1 to only differ by one bit

(e) (8 points) Explain why G_{n+1} must have 2^{n+1} elements in it.

Gn+1 must have 2^(n+1) elements since Gn has 2^n elements and the next sequence of gray codes is composed of 2 of the previous sequence of gray codes. Therefore Gn+1 would be 2*2^n or 2^(n+1) elements

(f) (8 points) Explain why $G_{n+1} = [0G_n, 1G'_n]$ fulfills are four requirement to be a Gray Code (see Red box above).

Gn+1 = [0Gn, 1G'n] fulfills the 4 requirements of a gray code. there are exactly 2^(n+1) elements because of the reasoning in 6e.
All the elements differ by 1 bit because of the reasoning in 6c and 6d.
Each element is only repeated once due to the unique appending.
Lastly, each member would have n+1 bits since Gn and G'n have n bits so 0Gn and 1G'n would have n+1 bits.