INTRO COMP ROBOTICS

CS 560 Homework #1

Xuenan Wang(xw336) - September 23, 2019

InMotion ARM Neurorehabilitation Robot

Problem 1

The Cosmic Engine, a 10-metre (33 ft) clock tower built by Su Song in Kaifeng, China, in 1088, featured mechanical mannequins that chimed the hours, ringing gongs or bells among other devices.

Among the first verifiable automation is a humanoid drawn by Leonardo da Vinci (1452–1519) in around 1495. Leonardo's notebooks, rediscovered in the 1950s, contain detailed drawings of a mechanical knight in armour which was able to sit up, wave its arms and move its head and jaw.

The Turk was a fake chess-playing machine constructed in the late 18th century. Constructed and unveiled in 1770 by Wolfgang von Kempelen to impress the Empress Maria Theresa of Austria, the mechanism appeared to be able to play a strong game of chess against a human opponent, as well as perform the knight's tour, a puzzle that requires the player to move a knight to occupy every square of a chessboard exactly once.

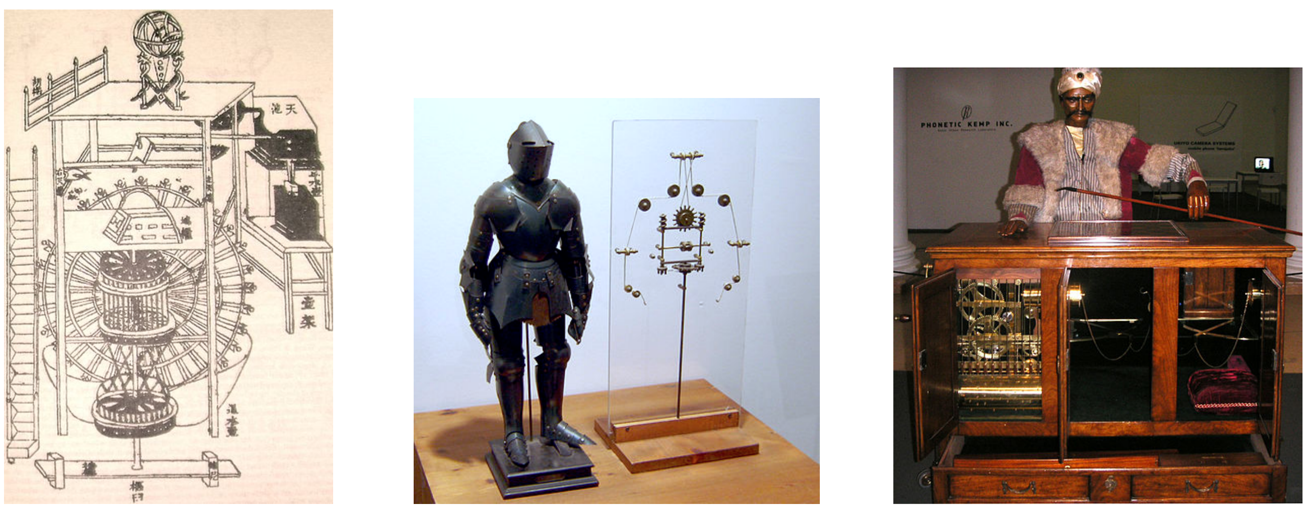


Figure 1. Clocktower by Su Song(left); Leonardo's robot(middle); The Turk(right).

Problem 2

Soccer and dice are convex, because according to the definition of convex, a convex polygon is defined as a polygon with all its interior angles less than 180°, which means that all vertices of the polygon will point outwards.

Problem 3

$P(\varnothing) = \{\varnothing\},\ P^2(\varnothing) = \{\varnothing,\ \{\varnothing\}\}$.
$|P^n(\varnothing)| = 2^{n-1}$.
If $|S| = k\ and\ S \neq \varnothing$, then $|P^n(S)| = 2^{nk}$.

Problem 4

1. Assume that there are two different identity elements $e_1,\ e_2$ in group G. Based on axioms 3, we have $e_1 \cdot e_2 = e_2 \cdot e_1 = e_1 = e_2$, which obviously contradict to our assumption. Therefore, group G has a unique identity element.
2. Based on the knowledge of question 1, we know that group G has a unique identity element. According to axioms 4, $a \cdot b = b \cdot a = e$. Since for each group, we have a unique e, we can say that for each a here, there are going to be only one possible b satisfying $a \cdot b = b \cdot a = e$. So we have the conclusion of for each $a \in G$, a has a unique inverse.

Problem 5

If we assume that these two robot arms will not hit each others and they won't hit the ground, the whole possible positions of the second segment tip would be equal to
$N_{possible\ positions\ of\ segment\ 1} \times N_{possible\ positions\ of\ segment\ 2} - N_{repeat\ positions}$
$= 2^{10} \times 2^{15} - [(2^{10} - 1) \times 2^{10}]$
$= 1024 \times 32768 - 1047552$
$= 32506880$

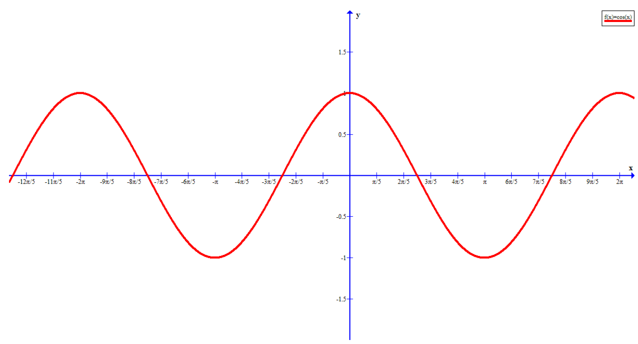
Problem 6

$$N_{(10\ sides\ expected)} = \sum_{m=1}^{10} \frac{10}{m} \approx 29.3 = 30\ times$$

$$N_{(n\ sides\ expected)} = \sum_{m=1}^{n} \frac{n}{m}$$

Problem 7

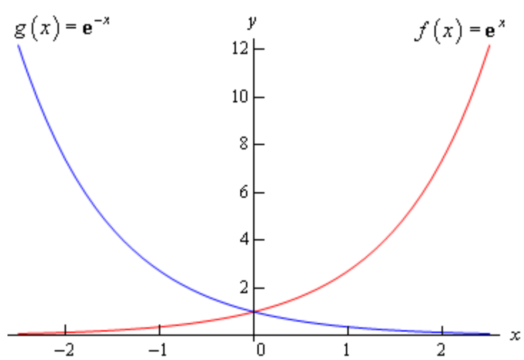
1. $f : [-\frac{\pi}{2}, \frac{\pi}{2}] \rightarrow [0,1],\ x \mapsto \cos x$ is surjective.



According to the plot of cos(x) above, we can see that for every y in [0, 1], there are at least one x values such that $x \mapsto y$ and x is not unique for each y.

To make it bijective, $[0,\frac{\pi}{2}] \rightarrow [0,1]$.

2. $f : \mathbb{R} \rightarrow \mathbb{R},\ x \mapsto e^x$ is injective.



According to the plot of e^x above, we can see that all $x \in \mathbb{R}$ have a corresponding y, but not all y has a corresponding x.

To make it bijective, $\mathbb{R} \rightarrow (0, +\infty]$.

Problem 8

1. $(-1,1) \mapsto \{x^2 + (y-1)^2 = 1 \,|\, (x,y) \neq (0,2)\} \mapsto \{x^2 + (y-1)^2 = +\infty \,|\, (x,y) \neq (0,+\infty)\}$

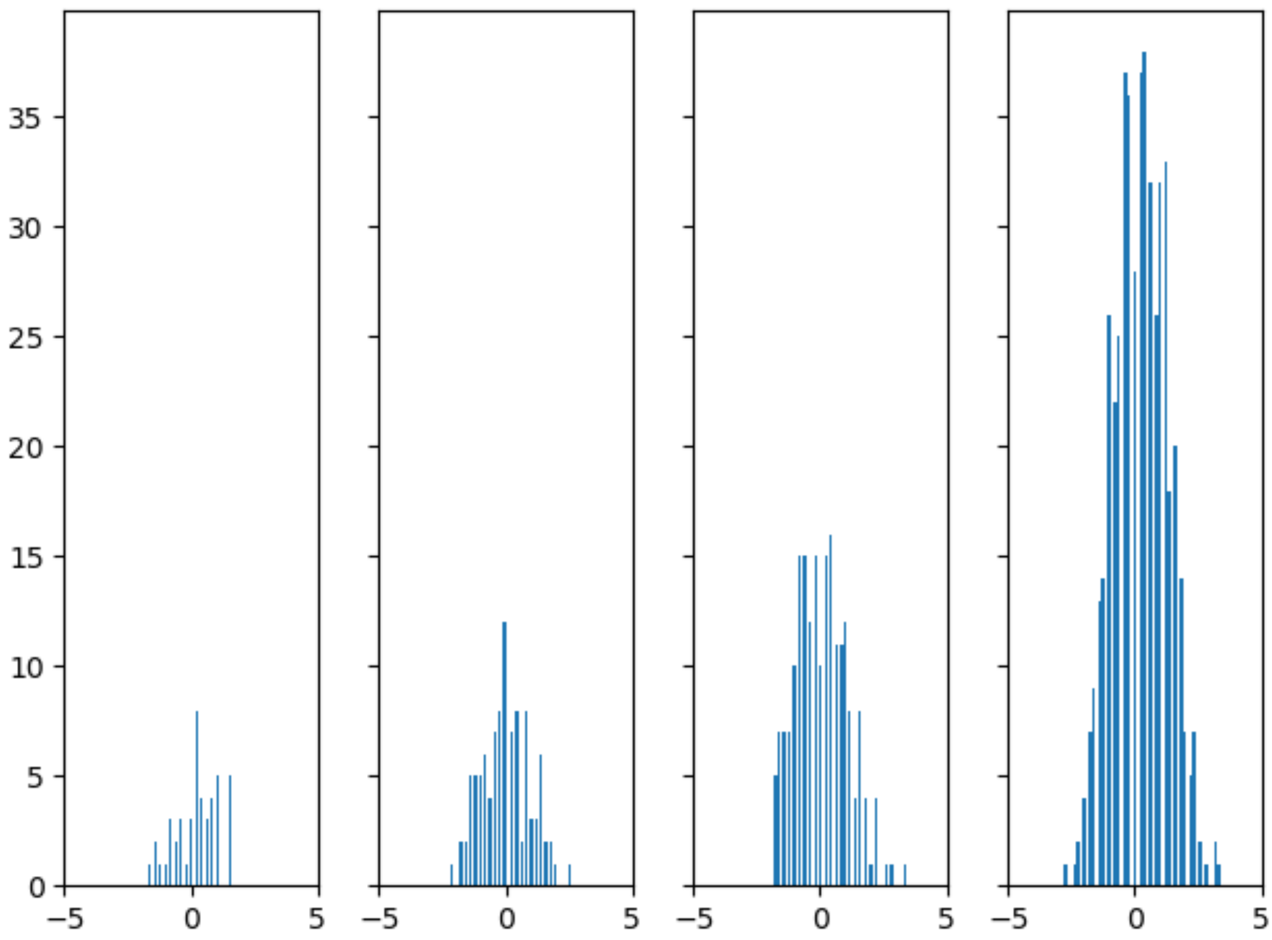
The real line can be compactified by adding a point at infinity.

2. $\{(x,y) \,|\, x^2 + y^2 = 1\} \mapsto \{(x,y) \,|\, \| (x,y) \|_\infty = 1\}$

Problem 9

X is a 1-dimensional manifold.

Problem 10



Bonus Problem

The binary-reflected Gray code list for n bits can be generated recursively from the list for n − 1 bits by reflecting the list (i.e. listing the entries in reverse order), prefixing the entries in the original list with a binary 0, prefixing the entries in the reflected list with a binary 1, and then concatenating the original list with the reversed list.

We need to prove that for consecutive two bits, they are absolute different.

Consider n and n+1 bit. If we add 1 to n, we are converting all ones in the tail to zeros and make lowest bit which was zero to one. As follows.

$$(n)_2 = \cdots 011 \cdots 11$$
$$(n+1)_2 = \cdots 100 \cdots 00$$

Therefore when we compute $g(n)$ and $g(n+1)$, k bit on the tail will become $100 \cdots 00$ and k+1 bit is different. Because for n and n+1 bits, everything except k+1 bit is all the same. So for k+1 bit, they either be $XOR(0)$ or $XOR(1)$, which won't change the fact that k and k+1 bits are different.

Above all, we can see that there will always be a way to encode in the way of gray code.

Appendix: source code for Q10

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
# @Date    : 2019-10-05 23:23:24
# @Author  : Xuenan(Roderick) Wang
# @Email   : roderick_wang@outlook.com
# @Github  : https://github.com/hello-rodericknwang

import math
import numpy as np
import matplotlib.pyplot as plt
from matplotlib import colors

def phi(x):
    return 0.5+(np.sign(x)/2)*math.sqrt(1-math.exp(-(2*x*x)/math.pi))

list_y = np.zeros(50)
list_x = np.arange(-5, 5, 0.2)
for i in range(50):
    list_y[i] = phi(list_x[i])

def get_sample(y_array):
    x_array = np.zeros(len(y_array))
    for i in range(len(y_array)):
        for j in range(len(list_y)):
            if y_array[i]<list_y[j]:
                x_array[i] = list_x[j]
                break
    return x_array

if __name__ == '__main__':
    sample50 = np.random.rand(50)
    sample100 = np.random.rand(100)
    sample200 = np.random.rand(200)
    sample500 = np.random.rand(500)
    result50 = np.zeros(50)
    result100 = np.zeros(100)
    result200 = np.zeros(200)
    result500 = np.zeros(500)
    result50 = get_sample(sample50)
    result100 = get_sample(sample100)
    result200 = get_sample(sample200)
    result500 = get_sample(sample500)
    fig, axs = plt.subplots(1, 4, sharey=True, tight_layout=True)
    axs[0].hist(result50, bins=50)
    axs[0].set_xlim(-5, 5, 0.2)
    axs[1].hist(result100, bins=50)
    axs[1].set_xlim(-5, 5, 0.2)
    axs[2].hist(result200, bins=50)
    axs[2].set_xlim(-5, 5, 0.2)
    axs[3].hist(result500, bins=50)
    axs[3].set_xlim(-5, 5, 0.2)
    plt.show()
```