Qishi Quiz 2

Yongjie Xin

June 9, 2019

Abstract

Results and answers.

Contents

1 Math 2

1.1 Q1 . . . 2

1.2 Q2 . . . 2

1.3 Q3 . . . 3

1.4 Q4 . . . 3

1.5 Q5 . . . 3

1.6 Q6 . . . 3

1.7 Q7 . . . 4

1.8 Q8 . . . 4

1.9 Q9 . . . 5

2 Programming 5

2.1 Q10 . . . 5

2.2 Q11 . . . 5

2.3 Q12 . . . 6

2.4 Q13 . . . 6

2.5 Q14 . . . 6

2.6 Q15 . . . 7

2.7 Q17 . . . 8

2.8 Q18 . . . 9

1 Math

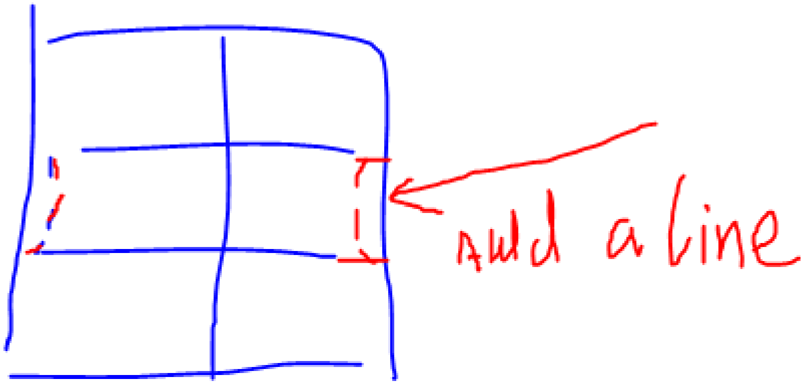
1.1 Q1

Chi-square test could be used in goodness of fit test and test of independence of the two variables. The reason that Chi-square test is used in the above two cases relating to the fact of its distribution. Chi-square distribution is defined as sum of squared iid normal variables.

If we assume the input data is from iid distribution but with a white noise. Then the data minus its expectation is just the white noise, a Gaussian variable. Sum of squared Gaussian variables should obey Chi-square distribution.

1.2 Q2

Chinese postman problem. Here we are looking for Eulerian trail. Theory: An undirected graph has an Eulerian trail **iff** exactly 0 or 2 vertices have odd degree, and all of its nonzero degree vertices belong to a single connected component. Originally we have 6 odd degree vertices. New we add two lines to have only 2 odd vertices. By theory, there should exists a Eulerian trail now. We have totally 19 edges.



1.3 Q3

1.4 Q4

(1), (2) the prob hit 36 is 1/6, with payoff zero. The next square number is 49. If we pass 36, We could reach **at least** 43 before worrying about it again. So the payoff if we pass 36 is $\geq 5/6 * 43 = 35.83$ So continue. (3) There must be a strategy because we can't play the game forever. With probability 1 that we could hit a square number if we play the game forever. If we could build a tree, and working backward from a very large number rolling, then we could get almost the perfect expected premium of this game. This would be the best strategy, since in this way we could build a "early exercise' boundary. I have seen an answer online, using a sub-optimal early exercise decision, https://math.stackexchange.com/questions/977679/toss-a-fair-die-until-the-cumulative-sum-is-a-perfect-square-expected-valueT. Maybe it is enough in a discrete case.

1.5 Q5

Referencing the answer here: { https://math.stackexchange.com/questions/13959/if-a-1-meter-rope-is-cut-at-two-uniformly-randomly-chosen-points-what-is-the-av The key part is to translate the probability space of smallest piece into the probability space of joint uniform random variables. } The expectation for smallest is 1/9, mid is 5/18, and largest is 11/18.

1.6 Q6

X_i is the event that person i picks the right hat, $X_i = 1$ when it is true, otherwise, $X_i = 0$;

$$X_i = \begin{cases} 1, & Prob = 1/N \\ 0, & Prob = (N-1)/N \end{cases}$$

As to why the prob is $1/N$: of all $N!$ combinations, there are $(N-1)!$ combinations that person i could get the right hat.

Same methodology(this is for the variance calculation):

$$X_i X_j = \begin{cases} 1, & Prob = \frac{1}{N(N-1)} \\ 0, & Prob = 1 - \frac{1}{N(N-1)} \end{cases}$$

Now $Y = \sum X_i$, expectation of Y is $\sum E[X_i] = 1$.

variance of Y is $\sum Var[X_i] + \sum_j \sum_i Cov(X_i, X_j)$; $Var[X_i] = \frac{1}{N} - \frac{1}{N^2}$. $Cov(X_i, X_j) = E[X_i X_j] - E[X_i]E[X_j] = \frac{1}{N(N-1)} - \frac{1}{N^2}$

Now the total variance is $NVar[X_i] + N^2 Cov(X_i, X_j) = 1$.

after change the rule Use conditional expectation, conditioning on what happens after 1st pick. Denote $E[R_i]$ as r_i. Denote the probability that only j persons picking the right hats of N hats as $p_j^{(N)}$.

$$E[R_N] = r_N = 1 + r_N p_0^{(N)} + r_{N-1} p_1^{(N)} + ... + r_2 p_{N-2}^{(N)} + r_0 p_N^{(N)} \tag{1}$$

in which, there is no r_1 and $r_0 = 0$. With this equation, we could work from N=2 all the way to a general case.

The probability $p_j^{(n)}$ is the prob of only j persons get the right hats out of n hats. $p_j^{(n)} = \frac{!(n-j)}{n!}$, in which $!(n-j)$ is the number of derangements.

Same methodology for $E[S(N)] = s_N$.

$$s_N = N + s_N p_0^{(N)} + s_{N-1} p_1^{(N)} + ... + s_2 p_{N-2}^{(N)} + s_0 p_N^{(N)} \tag{2}$$

With this equation, we could work from N=2 all the way to a general case.

The total expected selections is s_N. So the average selections is s_N/N for each person. The expected false selection for one person is $s_N/N - 1$.

1.7 Q7

Without loss of generosity, let's assume our variables are standardized. So that for Model-(Y, X), $R^2 = \rho_{XY}^2$. I need to think about it.

1.8 Q8

My understanding for this question is how to use a bias coin to generate a "fair coin". One way is to flip the bias coin twice, only take the HT and TH events, treating them as the "fair coin".

1.9 Q9

Starting from the 1st island, the expected bridges is a_1, starting from the $2nd$ island, the expected bridges is a_2, so on so forth.

$$
\begin{align}
a_1 &= 0.5(1 + a_2) + 0.5a_1 \tag{3}\\
a_2 &= 0.5(1 + a_3) + 0.5a_1 \tag{4}\\
\cdots & \tag{5}\\
a_9 &= 0.5 * 1 + 0.5a_1 \tag{6}
\end{align}
$$

in which, 9 linear equations with 9 variables.

2 Programming

2.1 Q10

c++ const is read from right to left, by replacing the operator to its name. Like $*$ is pointer and & is reference.

The most right const is indicating it is const function; if it is a member function of a class, it does not change data member except mutables.

const int* const & p read as p is a reference(&) of a const pointer(*) of a const int. In other words, there is a const int and it has a const pointer. p is the variable referencing this const pointer.

2.2 Q11

```
struct Node
{
    int data;
    struct Node *next;
};

void deleteNode(struct Node **head_ref, int key)
{
    // Store head node
    struct Node* temp = *head_ref, *prev;

    // If head node itself holds the key to be deleted
    if (temp != NULL && temp->data == key)
    {
        *head_ref = temp->next;   // Changed head
        free(temp);               // free old head
        return;
    }

    // Search for the key to be deleted, keep track of the
    // previous node as we need to change 'prev->next'
    while (temp != NULL && temp->data != key)
    {
        prev = temp;
        temp = temp->next;
    }

    // If key was not present in linked list
    if (temp == NULL) return;

    // Unlink the node from linked list
    prev->next = temp->next;

    free(temp);  // Free memory
}
```

2.3 Q12

2.4 Q13

Constructor itself can't be virtual. Because constructor type is determined at compile time. It has to be static. However, we could define a virtual function in the base class to constructor pointer of base or derived class given input. About this, a very helpful page I found online: https://www.geeksforgeeks.org/advanced-c-virtual-constructor/;

2.5 Q14

It is not ok.

```
Base *b = new Derived;
b -> invoke();
```

Here if the invoke method is non-virtual, but it calls a virtual function. The pointer b will use Base's invoke method, but inside of invoke will call derived class's function . https://www.geeksforgeeks.org/happens-virtual-function-called-inside-non-virtual-function/

2.6 Q15

```
#include<bits/stdc++.h>
using namespace std;

/* Returns LCS X and Y */
string lcs(string &X, string &Y)
{
    int m = X.length();
    int n = Y.length();

    int L[m+1][n+1];

    /* Following steps build L[m+1][n+1] in bottom
       up fashion. Note that L[i][j] contains
       length of LCS of X[0..i-1] and Y[0..j-1] */
    for (int i=0; i<=m; i++)
    {
        for (int j=0; j<=n; j++)
        {
            if (i == 0 || j == 0)
                L[i][j] = 0;
            else if (X[i-1] == Y[j-1])
                L[i][j] = L[i-1][j-1] + 1;
            else
                L[i][j] = max(L[i-1][j], L[i][j-1]);
        }
    }

    // Following code is used to print LCS
    int index = L[m][n];

    // Create a string length index+1 and
    // fill it with \0
    string lcs(index+1, '\0');

    // Start from the right-most-bottom-most
    // corner and one by one store characters
    // in lcs[]
    int i = m, j = n;
    while (i > 0 && j > 0)
    {
        // If current character in X[] and Y
        // are same, then current character
        // is part of LCS
        if (X[i-1] == Y[j-1])
        {
            // Put current character in result
            lcs[index-1] = X[i-1];
            i--;
            j--;

            // reduce values of i, j and index
            index--;
        }

        // If not same, then find the larger of
        // two and go in the direction of larger
        // value
        else if (L[i-1][j] > L[i][j-1])
            i--;
        else
            j--;
    }

    return lcs;
}

// Returns longest palindromic subsequence
// of str
string longestPalSubseq(string &str)
{
    // Find reverse of str
    string rev = str;
    reverse(rev.begin(), rev.end());

    // Return LCS of str and its reverse
    return lcs(str, rev);
}
```

Reference : https://www.geeksforgeeks.org/print-longest-palindromic-subsequence/

2.7 Q17

dynamic programming.

```
f(k, i) is the max payoff, after k transactions at stock i;
f(k, i) = f(k, i-1) + max( [f(k-1, j) + Stock(i) - Stock(j)], j from 0 till i );
```

2.8 Q18

Maybe DFS backtracking. haven't solved yet.