Quiz 2

October 11, 2015

Math/Stat

2) Given a 2 by 3 grid (which has 6 blocks and 17 edges), shortest route to visit all edges (assuming edge length is 1).

Solution 1 *Recall a well known fact that one can visit each edge of a (connected) graph G* **exactly once** *if and only if the number of vertices of odd degree is either* 0 *or* 2 *(the degree of a vertex v of G is the number of edges passing through that vertex in question). Our graph G in question has* 6 *vertices of degree* 3*, namely T_1, T_2 on the top side, B_1, B_2 one the bottom side, L on the left side and R on the right side. Thus there is no way to find a path to visit all edges exactly once. However, if we make two auxiliary edges connecting T_1, T_2 and B_1, B_2, then there are only two vertices of degree* 3 *left, namely L and R. One can find a path to visiting each edge of the new graph G' exactly once. This path has length 19 since the new graph has* 19 *edges. This means on the original graph G, there exists a path of length* 19 *to visit all edges (the path will visit edges T_1T_2 and B_1B_2 twice and all other edges once). It is easy to show this is the shortest path possible.*

5) Given a stick, if randomly cut into 3 pieces, what's the average size of the smallest, of the middle-sized, and of the largest pieces?

Solution 2 *Let X and Y be the random variable representing two of pieces, the thrid piece has length $1 - X - Y$. Thus X and Y are i.i.d with $\mathcal{U}(0, 1)$ distribution. We compute the c.d.f $F_{\max}$ of $\max X, Y, 1 - X - Y$ as follows:*

$$\begin{aligned} F_{\max}(t) : &= \mathbb{P}(\max\{X, Y, 1 - X - Y\} \leq t \mid X + Y \leq 1) \\ &= \mathbb{P}(X \leq t,\ Y \leq t,\ 1 - X - Y \leq t \mid X + Y \leq 1) \\ &= \frac{\mathbb{P}(X \leq t, Y \leq t, 1 - t \leq X + Y \leq 1)}{\mathbb{P}(X + Y \leq 1)} \end{aligned}$$

We easily compute that

$$F_{\max}(t) = \begin{cases} 0 & t \leq \frac{1}{3}, \\ (3t - 1)^2 & \frac{1}{3} < t \leq \frac{1}{2}, \\ 1 - 3(1 - t)^2 & \frac{1}{2} < t \leq 1. \end{cases}$$

We further compute the density function and then the expectation

$$\mathbb{E}_{\max} = \frac{11}{18}.$$

One can similarly compute the c.m.f of $\min\{X, Y, 1 - X - Y\}$*:*

$$F_{\min}(t) = \begin{cases} 1 - (1 - 3t)^2 & 0 \leq t \leq \frac{1}{3} \\ 1 & t > \frac{1}{3}. \end{cases}$$

The expectation is $\mathbb{E}_{\min} = \frac{1}{9}$*. The middle piece therefore has expectation* $\frac{5}{18}$*.*

C++ Coding Question

12) Implement the interface for matrix class in C++.

```
#include <iostream>
#include<vector>
using namespace std;

template<typename T>

class Matrix
{
public:
    Matrix(int A, int B, T t):RowNumber(A),ColNumber(B) //constructor innitialize A*B
        matrix with entries t.
    {
        MyMatrix.resize(A);
        for(int i=0;i<A;i++)
            MyMatrix[i].resize(B,t);


    }
    Matrix(const Matrix&
        OneMatrix):RowNumber(OneMatrix.RowNumber),ColNumber(OneMatrix.ColNumber)
        //copy constructor
   {
       MyMatrix.resize(RowNumber);
    for(int i=0;i<RowNumber;i++)
            MyMatrix[i].resize(ColNumber);
        for(int i;i<RowNumber;i++)
            for(int j=0;j<ColNumber;j++)
                MyMatrix[i][j]=OneMatrix.MyMatrix[i][j];
    }

    Matrix& operator=(const Matrix& OneMatrix) //overloading assignment operator
    {
        if(RowNumber!=OneMatrix.RowNumber||ColNumber!=OneMatrix.ColNumber)
        {cout<<"The matrices do not match."<<endl;
            throw -1;
        }
        for(int i=0;i<RowNumber;i++)
            for(int j=0;j<ColNumber;j++)
                MyMatrix[i][j]=OneMatrix.MyMatrix[i][j];
        return *this;


    }

    Matrix operator+(const Matrix& AnotherMatrix) //overloading +
    {
        if(RowNumber!=AnotherMatrix.RowNumber||ColNumber!=AnotherMatrix.ColNumber)
        {cout<<"The matrices do not match."<<endl;
            throw -1 ;
        }
        else
        {Matrix A(RowNumber,ColNumber,0);
            for(int i=0;i<RowNumber;i++)
                for(int j=0;j<ColNumber;j++)
                    A.MyMatrix[i][j]=MyMatrix[i][j]+AnotherMatrix.MyMatrix[i][j];
            return A;
        }

    }

    Matrix operator-(const Matrix& AnotherMatrix) //overloading -
    {
        if(RowNumber!=AnotherMatrix.RowNumber||ColNumber!=AnotherMatrix.ColNumber)
        {cout<<"The matrices do not match."<<endl;
            throw -1 ;
        }
        else
        {Matrix A(RowNumber,ColNumber,0);
            for(int i=0;i<RowNumber;i++)
            { for(int j=0;j<ColNumber;j++)
                    A.MyMatrix[i][j]=MyMatrix[i][j]-AnotherMatrix.MyMatrix[i][j];
            }
            return A;
        }

    }
    Matrix operator*(const Matrix& AnotherMatrix) //overloading matrix multiplication
    {
        if(ColNumber!=AnotherMatrix.RowNumber)
        {throw "The matrices can not be multiplied";

        }
        Matrix A(RowNumber,AnotherMatrix.ColNumber,0);
        for(int i=0;i<RowNumber;i++)
            for(int j=0;j<AnotherMatrix.ColNumber;j++)
            { for(int k=0;k<ColNumber;k++)
                    A.MyMatrix[i][j]+=MyMatrix[i][k]*AnotherMatrix.MyMatrix[k][j];
            }
        return A;

    }

    T& operator()(int A, int B)
    {   if(A>=RowNumber||B>=ColNumber||A<0||B<0)

        throw "Position not available";


        return MyMatrix[A][B];
    }

    void print()
    {
        for(int i=0;i<RowNumber;i++)
        { cout<<"\n";
            for(int j=0;j<ColNumber;j++)
                cout<<MyMatrix[i][j]<<'\t';}
        cout<<'\n';
    }
private:
    int RowNumber;
    int ColNumber;
    vector<vector<T>> MyMatrix;
};
```