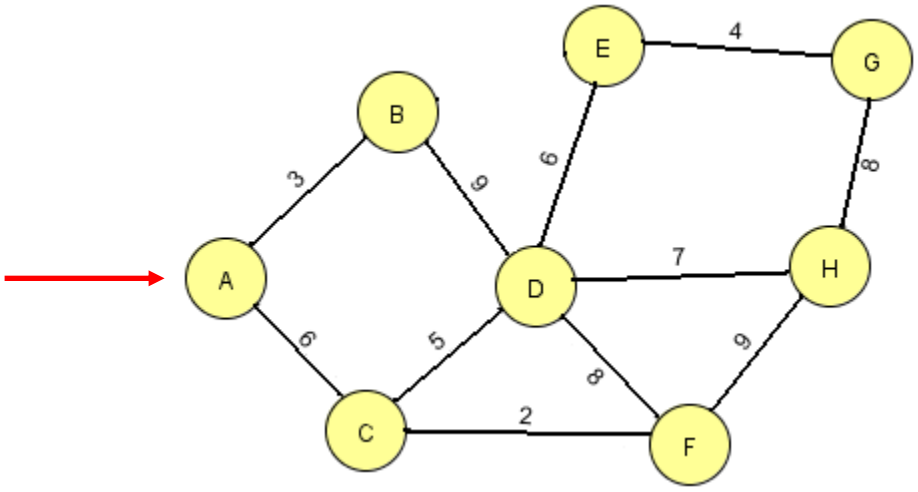
CS211 ALGORITHMS & DATA STRUCTURES II

LAB 10

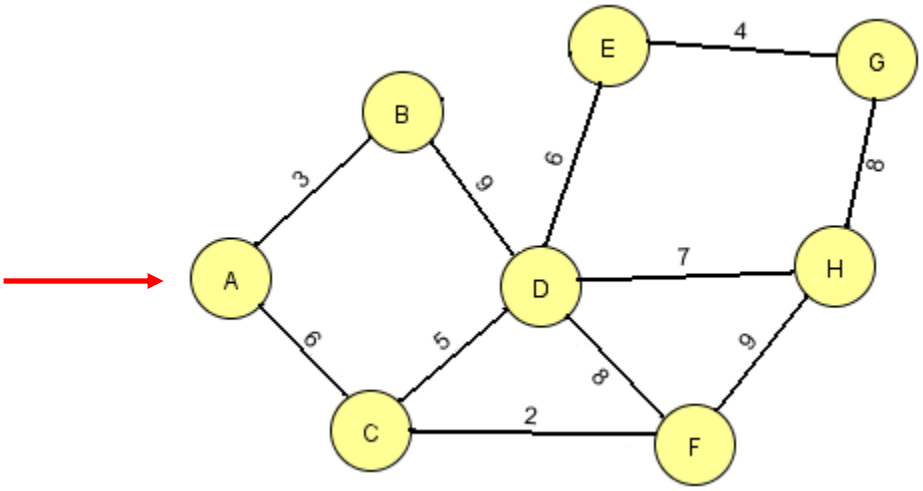
Dr. Phil Maguire

WEIGHTED GRAPHS



Vertices Visited	Edge Priority Queue	Edge Selected
A	AB3, AC6	AB3
A B	AC6, BD9	AC6
A B C	CF2, CD5, BD9	CF2
A B C F	CD5, FD8, BD9, FH9	CD5
A B C F D	DE6, DH7, FH9	DE6
A B C F D E	EG4, DH7, FH9	EG4
A B C F D E G	DH7, GH8, FH9	DH7

Find the shortest distance between Node A and Node G in the following graph using Dijkstra's algorithm.



Vertices Visited	A	B	C	D	E	F	G	H
A	0	3 (A)	6 (A)	-	-	-	-	-
A B	0	3 (A)	6 (A)	12 (B)	-	-	-	-
A B C	0	3 (A)	6 (A)	11 (C)	-	8 (C)	-	-
A B C F	0	3 (A)	6 (A)	11 (C)	-	8 (C)	-	17 (F)
A B C F D	0	3 (A)	6 (A)	11 (C)	17 (D)	8 (C)	-	17 (F)
A B C F D H	0	3 (A)	6 (A)	11 (C)	17 (D)	8 (C)	25 (H)	17 (F)
A B C F D H E	0	3 (A)	6 (A)	11 (C)	17 (D)	8 (C)	21 (E)	17 (F)
A B C F D H E G	0	3 (A)	6 (A)	11 (C)	17 (D)	8 (C)	**21 (E)**	17 (F)

PART II: Programming exercise

```
public class DiceRolling{

    public static void main(String[] args){

        double total=0;
        int montecarlo=100000;
        for(int i=0;i<montecarlo;){
            int sneeze=0;
            int counter=0;
            int dice=0;
            do{
                counter++;
                dice = (int)(Math.random()*6)+1;
                if((int)(Math.random()*1000)+1==1000){
                    sneeze++;
                }
            }while(dice!=6);
            if(sneeze==1){
                total+=counter;
                i++;
                System.out.println(counter+"    "+total/i);
            }
        }
        System.out.println(total/montecarlo);
    }
}
```