DM551 – Algorithms and Probability

Exam Assignment 2, Fall 2015

Mads Nicolai Møller Petersen
mpet006

04-12-2015

Problem 1

a

1

The expected number of satisfied clauses can be calculated as the probability that any clause is satisfied and then using linearity of expectation as in 13.4 of Kleinberg and Tardos. If each variable is assigned a truth statement, true or false, each with $\frac{1}{2}$ probability. Then the chance that a clause is satisfied is $1 - \frac{1}{2}^k$ When k is the number of variables. Therefore the number of satisfied clauses:

$$R = m(1 - \frac{1}{2}^k)$$

When m is the number of clauses and k is the number of variables.

2

The bound of $8m$ for $k = 3$ found in Kleinberg and Tardos seems much too high to give any real indication. Something like:

$$1/\sum_{i=\frac{7}{8}m}^{m} C(m,i)(F(k))^i(1-F(k))^{m-i} : i \in \mathbb{Z}, F(k) = 1 - \frac{1}{2}^k$$

This is the probability that $\frac{7}{8}$ of the clauses are true when variables are assigned true or false with $p = \frac{1}{2}$. And the variables must assigned values independently of each other.

b

Randomness is achieved by assigning each variable a random boolean value using a random number generator that uniformly randomly assigns true or false to each variable.

c

This is implemented as a constructor for an instance of 3SAT that generates $n + m$ clauses. The n clauses are always of the form $3n$ variables and n clauses with each variable used in exactly 1

clause. The m clauses are generated randomly such that there are between $0-3$ recurring variables in each of the m clauses and $0-3$ new variables. if $m=0$ there will always be generated equivalent instances.

d

The following tests are performed with 3SAT instances with $3n$ variables and n clauses called Generic1, instances with $\frac{1}{2}\cdot n$ clauses of the previously mentioned kind and $\frac{1}{2}\cdot n$ randomly generated clauses of the kind mentioned above called Generic2. And of the 2 kinds mentioned in the assignment called Case1 and Case2.

```
Running 10000 tests of Case 1 with n=128
There are 512 clauses and 132 variables
Average runs per test: 0.8623, Average sated clauses: 452.155

Running 10000 tests of Case 2 with n=8
There are 512 clauses and 24 variables
Average runs per test: 0.6296, Average sated clauses: 474.671

Running 10000 tests of Generic1 test with n=512
There are 512 clauses and 1331.6959 variables
Average runs per test: 0.8813, Average sated clauses: 453.9462

Running 10000 tests of Generic2 test with n=512
There are 512 clauses and 1536 variables
Average runs per test: 0.862, Average sated clauses: 453.6082
```

Case 2 has $n+4$ variables instead of $n+2$ because $a, b, \overline{a}, \overline{b}$ are all variables.

The average number of satisfied clauses for 3SAT with 512 clauses is $\frac{7}{8} \cdot 512 = 448$ which is not far off the average in the tests, tho still a bit lower. The number of average runs before is much much lower than $8m$ this upper bound seems all but useless. When there is a $Pr = \frac{7}{8}$ that a clause with 3 variables is sated when each variable is assigned true or false with $p = \frac{1}{2}$. A number of average runs $1/\sum_{i=\frac{7}{8}m}^{m} C(m,i)\frac{7}{8}^{i}\frac{1}{8}^{m-i} : i \in \mathbb{Z}$ seems much closer.

Problem 2

Not done

Problem 3

a

$$E[X_i] = \frac{1}{4} \Rightarrow E[X] = E\left[\sum_{i=1}^{m} X_i\right] \Rightarrow E[X] = \sum_{i=1}^{m} E[X_i] \Rightarrow E[X] = \sum_{i=1}^{m} \frac{1}{4} = \frac{m}{4}$$

b

Assuming the challenged student gets the answers correct to the 40% of the test he knows. he still needs another $0.1n$ correct answers, and since there is a $\frac{3}{4}$ chance of guessing incorrectly we get the number of points resulting from the remaining $3/5$ of the questions = $X - \frac{1}{3}(m - X)$, this must be greater than or equal to 10% of n.

$$\frac{n}{10} \leq X - \frac{1}{3}(m - X) \Rightarrow \frac{5}{30}m \leq X - \frac{1}{3}(m - X) \Rightarrow X \geq \frac{3}{8}m \Rightarrow X \geq 4 \cdot \frac{3}{8}E[X] = X \geq \frac{3}{2}E[X]$$

c

We have

$$Pr(X > \frac{3}{2}E[X]) < 0,05$$

and

$$Pr(X > (1+\delta)\mu) < \left[\frac{e^{\delta}}{(1+\delta)^{(1+\delta)}}\right]^{\mu}$$

If

$$(1+\delta) = \frac{3}{2}, \mu = E[X], \delta = \frac{1}{2}, E[X] = \frac{m}{4}, m = \frac{3}{5}n$$

We get

$$Pr(X > \frac{3}{2}E[X]) < \left[\frac{e^{\frac{1}{2}}}{\frac{3}{2}^{\frac{3}{2}}}\right]^{\frac{3}{20}n}$$

Solving for n

$$\left[\frac{e^{\frac{1}{2}}}{\frac{3}{2}^{\frac{3}{2}}}\right]^{\frac{3}{20}n} = 0,05 \Rightarrow n = 184,584 \approx 185$$

Problem 4

a

Because if Jørgen is in a place with 0 umbrellas, represented by node 0, There is a 100% chance that he will move to a location that has 4 umbrellas(node 4). Whether it rains or not, since he can't take any umbrellas. If he is in a location with 4 umbrellas, then if it rains, that is with probability p, he will go to a location that has 1 umbrella(node 1), since he will bring one from his current location to the location with 0 umbrellas. If it does not rain, probability q, then he will move back to a location with 0 umbrellas since he does not bring any. If he is in a location with 3 umbrellas and it rains, then he will bring 1 umbrella to a location with 1. And will therefore be in a location with 2 umbrellas(node 2). If it does not rain he will move to a location with 1 umbrella since he does not bring 1. When he is in a location with 2 umbrellas, if it does not rain, he will move to a location that also has 2 umbrellas. If it does rain then he will bring an umbrella and move to a location where there already is 2, and thus be in a location with 3 umbrellas. (Umbrella - ella - ella).

d

If p = 1, then there is a 100% chance that it rains whenever Jørgen leaves his house/office. And he will always bring an umbrella. Thereby putting him in a stable state moving between nodes 4 and 1, or nodes 3 and 2. Even tho he might start out in any of the nodes he will end up moving between either these 2 pairs.

Problem 5

a

The characteristic equation is:

$$r^2 - 4r + 4 = 0$$

This has just 1 solution, $r = 2$ Therefore the the solution is $a_n = \alpha_1 2^n + \alpha_2 n 2^n$ with $a_0 = 1, a_1 = 4$ we get:

$$a_0 = 1 = \alpha_1, a_1 = 4 = 1 \cdot 2 + \alpha_2 \cdot 1 \cdot 2 \Rightarrow \alpha_2 = 1$$

therefore the solution is:

$$a_n = 2^n + n2^n$$

b

We know from the above solution that the associated equation's solution is $a_n^{(h)} = 2^n + n2^n$ and $F(n) = n^2$ the specific solution is found with mathematica to be:

$$a_n = n^2 + 8n + 2^n(5n + 2) + 20$$

Appendix

Source code

```
public class Main {

    public static void main(String[] args)
    {
        RunCase1(128,10000);
        RunCase2(8,10000);
        RunGeneric1(512,10000);
        RunGeneric2(512,10000);
    }

    public static void RunCase1(int n, int tests)
    {
        System.out.println("Running "+tests+" tests of Case 1 with n="+n);
        int averageRuns = 0;
        int averageSated = 0;
        ThreeSat temp = new ThreeSat("Case1",n);
        System.out.println("There are "+temp.numClauses+" clauses and "+temp.
            GetNumberOfVars()+" variables");
        for(int i = 0; i < tests; i++)
        {
            //System.out.print("Test "+i+": ");
            temp = new ThreeSat("Case1",n);
            int satedClauses = 0;
            int runs = 0;
            for(Clause clause : temp.clauses)
            {
                if(clause.isTrue())
                    satedClauses++;
            }
            while(satedClauses < (7.0/8.0)*temp.numClauses)
            {
                temp = new ThreeSat("Case1",n);
                runs++;
                satedClauses = 0;
                for(Clause clause : temp.clauses)
                {
                    if(clause.isTrue())
                        satedClauses++;
                }
            }
            averageSated = averageSated + satedClauses;
            averageRuns = averageRuns + runs;
            //System.out.println("Number of clauses: "+temp.numClauses+", Number of
                sated clauses: "+satedClauses+", Number of runs:"+runs);
        }
        System.out.println("Average runs per test: "+(float)averageRuns/(float)tests+",
             Average sated clauses: "+(float)averageSated/tests);
        System.out.println();
    }

    public static void RunCase2(int n, int tests)
    {
        System.out.println("Running "+tests+" tests of Case 2 with n="+n);
        int averageRuns = 0;
        int averageSated = 0;
        ThreeSat temp = new ThreeSat("Case2",n);
        System.out.println("There are "+temp.numClauses+" clauses and "+temp.
            GetNumberOfVars()+" variables");
        for(int i = 0; i < tests; i++)
        {
            //System.out.print("Test "+i+": ");
            temp = new ThreeSat("Case2",n);
            int satedClauses = 0;
            int runs = 0;
            for(Clause clause : temp.clauses)
            {
```

```
            if(clause.isTrue())
                satedClauses++;
        }
        while(satedClauses < (7.0/8.0)*temp.numClauses)
        {
            temp = new ThreeSat("Case2",n);
            runs++;
            satedClauses = 0;
            for(Clause clause : temp.clauses)
            {
                if(clause.isTrue())
                    satedClauses++;
            }
        }
        averageSated = averageSated + satedClauses;
        averageRuns = averageRuns + runs;
        //System.out.println("Number of clauses: "+temp.numClauses+", Number of
            sated clauses: "+satedClauses+", Number of runs:"+runs);
    }
    System.out.println("Average runs per test: "+(float)averageRuns/(float)tests+",
         Average sated clauses: "+(float)averageSated/tests);
    System.out.println();
}

public static void RunGeneric1(int n, int tests)
{
    System.out.println("Running "+tests+" tests of Generic1 test with n="+n);
    int totalRuns = 0;
    int totalSated = 0;
    int totalVars = 0;
    ThreeSat temp;
    if(n % 2 == 0)
        temp = new ThreeSat(n/2,n/2);
    else
        temp = new ThreeSat((n-1)/2,(n+1)/2);
    System.out.print("There are "+temp.numClauses+" clauses ");

    for(int i = 0; i < tests; i++)
    {
        //System.out.print("Test "+i+": ");
        if(n % 2 == 0)
            temp = new ThreeSat(n/2,n/2);
        else
            temp = new ThreeSat((n-1)/2,(n+1)/2);
        int satedClauses = 0;
        int runs = 0;
        for(Clause clause : temp.clauses)
        {
            if(clause.isTrue())
                satedClauses++;
        }
        while(satedClauses < (7.0/8.0)*temp.numClauses)
        {
            temp = new ThreeSat(n,0);
            runs++;
            satedClauses = 0;
            for(Clause clause : temp.clauses)
            {
                if(clause.isTrue())
                    satedClauses++;
            }
        }
        totalVars = totalVars + temp.GetNumberOfVars();
        totalSated = totalSated + satedClauses;
        totalRuns = totalRuns + runs;
        //System.out.println("Number of clauses: "+temp.numClauses+", Number of
            sated clauses: "+satedClauses+", Number of runs:"+runs);
    }
    System.out.println("and "+totalVars/(float)tests+" variables");
    System.out.println("Average runs per test: "+(float)totalRuns/(float)tests+",
        Average sated clauses: "+(float)totalSated/tests);
    System.out.println();
```

```
    }

    public static void RunGeneric2(int n, int tests)
    {
        System.out.println("Running "+tests+" tests of Generic2 test with n="+n);
        int totalRuns = 0;
        int totalSated = 0;
        int totalVars = 0;
        ThreeSat temp;
        temp = new ThreeSat(n,0);
        System.out.print("There are "+temp.numClauses+" clauses ");

        for(int i = 0; i < tests; i++)
        {
            //System.out.print("Test "+i+": ");
            temp = new ThreeSat(n,0);
            int satedClauses = 0;
            int runs = 0;
            for(Clause clause : temp.clauses)
            {
                if(clause.isTrue())
                    satedClauses++;
            }
            while(satedClauses < (7.0/8.0)*temp.numClauses)
            {
                temp = new ThreeSat(n,0);
                runs++;
                satedClauses = 0;
                for(Clause clause : temp.clauses)
                {
                    if(clause.isTrue())
                        satedClauses++;
                }
            }
            totalVars = totalVars + temp.GetNumberOfVars();
            totalSated = totalSated + satedClauses;
            totalRuns = totalRuns + runs;
            //System.out.println("Number of clauses: "+temp.numClauses+", Number of
                sated clauses: "+satedClauses+", Number of runs:"+runs);
        }
        System.out.println("and "+totalVars/tests+" variables");
        System.out.println("Average runs per test: "+(float)totalRuns/(float)tests+",
            Average sated clauses: "+(float)totalSated/tests);
        System.out.println();
    }
}

import java.util.Random;
import java.util.ArrayList;

public class Clause
{
    int numberOfVars = 0;
    ArrayList<Variable> vars;

    public Clause(int NumVars)
    {
        numberOfVars = numberOfVars + NumVars;
        vars = new ArrayList(numberOfVars);
        Random myRand = new Random();
        for(int i = 0; i < numberOfVars; i++)
            vars.add(i, new Variable(myRand.nextBoolean()));
    }

    public Clause(Variable[] initVars)
    {
        int size = initVars.length;
        numberOfVars = numberOfVars + size;
        vars = new ArrayList(size);
        vars.addAll(java.util.Arrays.asList(initVars));
    }
```

```
    public ArrayList<Variable> GetVars()
    {
        return vars;
    }

    public Variable GetRandomVariable()
    {
        Random myRand = new Random();
        return vars.get(myRand.nextInt(numberOfVars));
    }

    public void AddVariable(Variable var)
    {
        vars.add(var);
        numberOfVars++;
    }

    public boolean isTrue()
    {
        boolean status = false;
        for(Variable var : vars)
        {
            if(var.value == true)
                status = true;
        }
        return status;
    }
}

public final class Variable
{
    public boolean value;
    Variable inverse;

    public Variable(boolean val)
    {
        value = val;
        inverse = new Variable();
        inverse.value = !value;
    }

    public Variable()
    {

    }

    public void ChangeValue(boolean newValue)
    {
        value = newValue;
        inverse.value = !newValue;
    }

    @Override
    public String toString()
    {
        return Boolean.toString(value);
    }
}

import java.util.Random;

public class ThreeSat
{
    Clause[] clauses;
    int numClauses;
    int numVariables = 0;

    public ThreeSat(String Arg, int n)
    {
        Random myRand = new Random();
        if(Arg.equals("Case1"))
        {
```

```
            Variable a = new Variable(myRand.nextBoolean());
            Variable b = new Variable(myRand.nextBoolean());
            Variable notA = a.inverse;
            Variable notB = b.inverse;
            numVariables = 4;
            numClauses = 4*n;
            clauses = new Clause[numClauses];
            for(int i = 0; i < 4*n; )
            {
                Variable temp = new Variable(myRand.nextBoolean());
                numVariables = numVariables + 1;
                clauses[i] = new Clause(new Variable[]{temp,a,b});
                i++;
                clauses[i] = new Clause(new Variable[]{temp,notA,b});
                i++;
                clauses[i] = new Clause(new Variable[]{temp,a,notB});
                i++;
                clauses[i] = new Clause(new Variable[]{temp,notA,notB});
                i++;
            }
        }

        if(Arg.equals("Case2"))
        {
            numClauses = (int)java.lang.Math.pow(n, 3);
            clauses = new Clause[numClauses];
            Variable[] xs = new Variable[n];
            Variable[] ys = new Variable[n];
            Variable[] zs = new Variable[n];
            numVariables = numVariables + 3*n;
            for(int i = 0; i < n; i++)
            {
                xs[i] = new Variable(myRand.nextBoolean());
                ys[i] = new Variable(myRand.nextBoolean());
                zs[i] = new Variable(myRand.nextBoolean());
            }
            int i = 0;
            for(Variable x : xs)
            {
                for(Variable y : ys)
                {
                    for(Variable z : zs)
                    {
                        clauses[i] = new Clause(new Variable[]{x,y,z});
                        i++;
                    }
                }
            }
        }
    }

    public ThreeSat(int n,int m)
    {
       Random myRand = new Random();
        numClauses = n+m;
        clauses = new Clause[numClauses];
        for(int i = 0; i < n; i++)
        {
            clauses[i] = new Clause(3);
            numVariables = numVariables + 3;
        }
        for(int i = 0; i < m; i++)
        {
            int choice = myRand.nextInt(4);
            if(choice == 0)
            {
                clauses[n+i] = new Clause(new Variable[]{GetRandomClause(n).
                    GetRandomVariable(),GetRandomClause(n).GetRandomVariable(),
                    GetRandomClause(n).GetRandomVariable()});
            }
            if(choice == 1)
            {
```

```
                clauses[n+i] = new Clause(1);
                clauses[n+i].AddVariable(GetRandomClause(n).GetRandomVariable());
                clauses[n+i].AddVariable(GetRandomClause(n).GetRandomVariable());
                numVariables = numVariables + 1;
            }
            if(choice == 2)
            {
                clauses[n+i] = new Clause(2);
                clauses[n+i].AddVariable(GetRandomClause(n).GetRandomVariable());
                numVariables = numVariables + 2;
            }
            if(choice == 3)
            {
                clauses[n+i] = new Clause(3);
                numVariables = numVariables + 3;
            }

        }
    }

    public ThreeSat(Clause[] initClauses)
    {
        numClauses = initClauses.length;
        clauses = new Clause[numClauses];
        System.arraycopy(initClauses, 0, clauses, 0, numClauses);
    }

    public final Clause GetRandomClause()
    {
        Random myRand = new java.util.Random();
        return clauses[myRand.nextInt(numClauses)];
    }
    public final Clause GetRandomClause(int limit)
    {
        Random myRand = new java.util.Random();
        return clauses[myRand.nextInt(limit)];
    }

    public int GetNumberOfVars()//not working as intended.
    {
        return numVariables;
    }
}
```