DM551 – Algorithms and Probability

Assignment 1, Fall 2015

Mads Petersen
mpet006

30-10-2015

1

a

For n teams, the number of matches can be viewed like this:
Team 1 will have to play against all other teams, that is team 2,3,4...n, If we write it in form of a list of matches that each team has to play, it could look like:
Team 1: $(1,2),(1,3),(1,4),...,(1,n)$. Team two will also have to play against all other teams, but their match with team 1, is already listed, so that list looks like:
Team 2: $(2,3),(2,4),...,(2,n)$. for team 3, we have:
Team 3: $(3,4),...,(3,n)$. As can be seen there are exactly $n-1$ matches listed for Team 1 in this manner, and $n-2$ matches for team 2, $n-3$ for team 3, 1 match for team $n-1$ and 0 matches for team n. So the total number of matches can be calculated as

$$matches_{total} = \sum_{1}^{n-1} i$$

It is also equivalent to counting subsets of size 2, from a set with 5 elements, without repetition:

$$matches_{total} = \frac{n!}{2!(n-2)!}$$

For $n = 5$ we have:

$$matches_{total} = \frac{5!}{2!(5-2)!} = \sum_{1}^{4} i = 10$$

b

Each team plays $n-1$ matches, since no ties are possible, there must be a winner and a loser of each match. Therefore there are as many wins as there are total matches, and as many losses as there are total matches. So the least amount of wins that some team must have can be found if we divide the wins out evenly among the teams:

$$k = \frac{matches_{total}}{n}$$

But e.g. for n = 4, this would give:

$$k = \frac{6}{4} = 1,5$$

Obviously a team can't have 1,5 wins, this means that there is one team with 2 wins, and 1 with 1 win. So if the average is higher than 1, then it means there is a team that has 2 wins. So to get the correct k-value, we have to do a ceiling division:

$$k = \lceil \frac{matches_{total}}{n} \rceil$$

So for 5 teams, there must be some team, that has at least :

$$k = \lceil \frac{\frac{5!}{2!(5-2)!}}{5} \rceil = \lceil \frac{5!}{2!(5-2)!5} \rceil = \lceil \frac{5 \cdot 4}{2 \cdot 5} \rceil = 2$$

c

If no team scores more than 5 goals, that means there is 6^2 possible scores, those are:
(0,0), (0,1, (0,2), (0,3), (0,4), (0,5)
(1,0) ... (1,5)
(2,0) ... (2,5)
(3,0) ... (3,5)
(4,0) ... (4,5)
(5,0) ... (5,5)
That's 36 possible scores, therefore by the pigeon hole principle there has to be 37 matches at least before 2 of them must have the same result. This gives:

$$37 \leq \frac{n!}{2!(n-2)!}$$

And we know that for $n = 5, matches_{total} = 10 \Rightarrow n > 5$. Since n is an integer, it should be fairly simple to move forward by guessing.
First try, $n = 10 \Rightarrow \frac{10!}{2!(8-2)!} = 45$.
Second try, $n = 9 \Rightarrow \frac{9!}{2!(9-2)!} = 36$.
Based on this it can be concluded that the minimum number of teams, $n = 10$.

d

It will be assumed that the matches are evenly, or as close as can be, distributed across each of those 4 months. Based on the observation that Ramsey numbers says something about mutual friends at a party. These friends have a binary relationship, they are either friends or they are enemies. If friends are replaced by teams, and the binary relationship is replaced with having either played, or not played. When we then know that the Ramsey number $R(3,3) = 6$, we can then say that among 6 teams, there will be 3 mutual "friends", amongst those 6 teams, that have either played each other, or have not. Based on this we can say that there must be at least 6 teams represented in one of the 2-month periods. It is only necessary in one of them, since if we know it is true for 1 of those 2-month periods, then we know that in the period the teams has either played, in which case we don't care about the second period, or they will have NOT played, in which case they have to play in the second period, if all teams have to play each other. Knowing that 6 teams have to be represented, after how many matches can we be certain that is true.
After 1 match, there will always be 2 teams represented.
After 2 matches, there may be a recurrence from the first match, in which case there may only be 3 teams represented.
After 3 matches, there could be 1 recurrence from each of the 2 previous matches, but if that is the case, then those 3 teams will all have played each other, so it has to be assumed that this doesn't happen, therefore there can be only 1 recurrence, therefore 4 teams will be represented.
After 4 matches, again there can only be 1 recurrence from the previous 3 for the same reasons as above, therefore at least 5 teams will have been represented. After 5 matches, continuing the logic from above, we have at least 6 teams represented.
So after 5 matches there will be 6 teams represented, and if 6 teams are represented there will be 3

among them, that all have either played each other, or have not. There must then be at least 5 matches in one of the 2-month periods, and there must obviously be at least 6 teams in the tournament. Based on the assumption that the matches are evenly distributed across all 4 months, and that there must be 5 matches in a 2-month period, we can say that there must be at least 10 matches total. The total number of matches for 6 teams is calculated as:

$$\frac{6!}{2!(6-2)!} = 15$$

Therefore it is concluded that if there are 6 teams in the tournament, then there will be more than 10 matches, and therefore 5 or more matches in one of the 2-month periods, and among those 5+ matches, all 6 teams will be represented, and among the 6 teams there will be 3 that have either all played each other, or have all not played, and therefore must play in the second 2-month period.

2

a

```
>>> allPermutations(['a','b','c','d'])
['a', 'b', 'c', 'd'] 0
['a', 'b', 'd', 'c'] 1
['a', 'c', 'b', 'd'] 2
['a', 'c', 'd', 'b'] 3
['a', 'd', 'b', 'c'] 4
['a', 'd', 'c', 'b'] 5
['b', 'a', 'c', 'd'] 6
['b', 'a', 'd', 'c'] 7
['b', 'c', 'a', 'd'] 8
['b', 'c', 'd', 'a'] 9
['b', 'd', 'a', 'c'] 10
['b', 'd', 'c', 'a'] 11
['c', 'a', 'b', 'd'] 12
['c', 'a', 'd', 'b'] 13
['c', 'b', 'a', 'd'] 14
['c', 'b', 'd', 'a'] 15
['c', 'd', 'a', 'b'] 16
['c', 'd', 'b', 'a'] 17
['d', 'a', 'b', 'c'] 18
['d', 'a', 'c', 'b'] 19
['d', 'b', 'a', 'c'] 20
['d', 'b', 'c', 'a'] 21
['d', 'c', 'a', 'b'] 22
['d', 'c', 'b', 'a'] 23
```

b

```
>>> allSubsetsInRange(['a','b','c','d','e'],1,5)
['a'] 0
['b'] 1
['c'] 2
['d'] 3
['e'] 4
['a', 'b'] 5
['a', 'c'] 6
['a', 'd'] 7
['a', 'e'] 8
['b', 'c'] 9
['b', 'd'] 10
['b', 'e'] 11
['c', 'd'] 12
['c', 'e'] 13
['d', 'e'] 14
['a', 'b', 'c'] 15
['a', 'b', 'd'] 16
['a', 'b', 'e'] 17
['a', 'c', 'd'] 18
['a', 'c', 'e'] 19
['a', 'd', 'e'] 20
['b', 'c', 'd'] 21
['b', 'c', 'e'] 22
['b', 'd', 'e'] 23
['c', 'd', 'e'] 24
['a', 'b', 'c', 'd'] 25
['a', 'b', 'c', 'e'] 26
['a', 'b', 'd', 'e'] 27
['a', 'c', 'd', 'e'] 28
['b', 'c', 'd', 'e'] 29
['a', 'b', 'c', 'd', 'e'] 30
```

c

```
>>> allSubsetsInRange(['a','b','c','d','e'],3,3)
['a', 'b', 'c'] 0
['a', 'b', 'd'] 1
['a', 'b', 'e'] 2
['a', 'c', 'd'] 3
['a', 'c', 'e'] 4
['a', 'd', 'e'] 5
['b', 'c', 'd'] 6
['b', 'c', 'e'] 7
['b', 'd', 'e'] 8
['c', 'd', 'e'] 9
```

d

```
>>> allSubsetsInRange(['a','b','c','d','e'],4,4)
['a', 'b', 'c', 'd'] 0
['a', 'b', 'c', 'e'] 1
['a', 'b', 'd', 'e'] 2
['a', 'c', 'd', 'e'] 3
['b', 'c', 'd', 'e'] 4
```

It can be seen that after $[a, b, d, e]$ comes $[a, c, d, e]$.

3

Not done

4

(i)

The number of distinct 2-partitions is also the number of ways that $|V|$ balls can be distributed into 2 identical containers. Therefore

$$|S| = S2(|V|,1) + S2(|V|,2)$$

where $S2$ denotes Stirling numbers of the second kind. $S2(|V|,1) = 1$ in all cases since there is only 1 way to distribute any number of objects into 1 box. This leads to:

$$|S| = 1 + S2(|V|,2)$$

From Rosen p.418.

$$S2(n,j) = \frac{1}{j!}\sum_{i=0}^{j-1}(-1)^i\binom{j}{i}(j-i)^n$$

with $j = 2$:

$$S2(n,2) = \frac{1}{2}\sum_{i=0}^{1}(-1)^i\binom{2}{i}(j-i)^n = \frac{1}{2}\cdot 1\cdot 1\cdot 2^n + \frac{1}{2}\cdot(-1)\cdot 2\cdot 1^n = 2^{n-1}-1$$

Finally this gives the total number of possible 2-partitions:

$$|S| = 1 + 2^{|V|-1} - 1 = 2^{|V|-1}$$

(ii)

Let's look at the distribution of elements over 2 sets if there's $1/2$ chance that an element ends up in either set. Let $V = \{1,2,3,4\}$ the 2-partition of $V = A_1 \cup B_1$. After distributing the first element, there's a $1/2$ chance of having $A_1 = \{1\}, B_1 = \{\}$ and there's $1/2$ chance of having $A_2 = \{\}, B_2 = \{1\}$. Since these 2 options are identical (from the definition of a 2-partition, they are identical if $A_1 = A_2$ or $A_1 = B_2$), there's a $1/1$ chance of having the 2 partition $\{\{1\},\{\}\}$. From here there's a $1/2$ chance that the next element is added to either A or B, so there's $1/2$ chance of having $\{\{1,2\},\{\}\}$ and $1/2$ of $\{\{2\},\{1\}\}$. Therefore after the last element is distributed we have $2^{|V|}$ possibilities. But since precisely half of them are identical, we end up with $2^{|V|}/2 = 2^{|V|-1}$ and each possibility has the chance $1/2^{|V|-1}$. We know from (i) that $|S| = 2^{|V|-1}$ thus $1/2^{|V|-1}$ = 1 / $|S|$.

(iii)

The expected size of U:

$$p(|U| = n) = \frac{\binom{|V|}{n}}{2}\cdot n\cdot\frac{1}{2^{|V|-1}} = \binom{|V|}{n}\cdot n\cdot\frac{1}{2^{|V|}}$$

therefore:

$$E(|U|) = \sum_{n=0}^{|V|}\binom{|V|}{n}\cdot n\cdot\frac{1}{2^{|V|}}$$

And via mathematica it can be shown that:

$$\sum_{n=0}^{|V|} \binom{|V|}{n} \cdot n \cdot \frac{1}{2^{|V|}} = \frac{|V|}{2}$$

So it can be concluded that:

$$E(|U|) = \frac{|V|}{2}$$

And since in and 2-partition of V $|W| = |V| - |U|$, therefore:

$$E(|W|) = |V| - \frac{|V|}{2} = \frac{|V|}{2}$$

a

For uv to be in E' precisely 1 vertex of uv must be in U, so what are the chances of that?. Looking at the distribution of just the 2 vertices in uv let them be $V1, V2$: There is $1/2$ chance after distributing 1 element if either:
$U = \{V1\}, W = \{\}$
$U = \{\}, W = \{V1\}$
after the second element $V2$, there's $1/4$ chance of either:
$U = \{V1, V2\}, W = \{\}$
$U = \{\}, W = \{V1, V2\}$
$U = \{V1\}, W = \{V2\}$
$U = \{V2\}, W = \{V1\}$
Out of the 4 there are 2 cases where precisely 1 vertex of uv belongs to U. Therefore there is 2/4 = $1/2$ chances that $uv\ inE'$.

b

The chance that $e \in U$ is the chance that $uv \in E'$, therefore we know that there is precisely $1/2$ chance that $e \in U$. That means that for each edge e there's precisely $1/2$ chance that $X_e(s) = 1$ and $1 - 1/2 = 1/2$ chance that $X_e(s) = 0$ therefore

$$E(X_e) = \frac{1/2 \cdot |E|}{|E|} \cdot 1 + \frac{1/2 \cdot |E|}{|E|} \cdot 0 = 1/2 \cdot 1 = 1/2$$

c-e

not done.

5

a

The number of non-negative, integer solutions to $x_1 + x_2 + x_3 = 10$ can be solved like the stars and bars problem with 10 stars and 2 bars. The resulting 3 groupings of stars are then equivalent to the values of x_1, x_2, x_3. Thus the number of solutions:

$$\frac{(10+3-1)!}{10!(12-10)!} = \frac{12!}{10!2!} = \frac{12 \cdot 11}{2} = 6 \cdot 11 = 66$$

b

This is equivalent to already having picked 5 items i.e. there's already 2 stars in one container and 3 in another. This leaves $10 - 5$ items to be picked. The number of solutions to this is the same as the number of solutions to $x_1 + x_2 + x_3 = 5$ which by following the same procedure as for a gives:

$$\frac{7!}{5!2!} = \frac{7 \cdot 6}{2} = 7 \cdot 3 = 21$$

c

This is equivalent to a stars and bars problem with 2 bars and 10 stars. And each container holding no more than 5 balls. That is then the same as:

$$x_1 + x_2 + x_3 = 10; x_1 \leq 5, x_2 \leq 5, x_3 \leq 5$$

This is equivalent to the number of solutions to $x_1 + x_2 + x_3 = 10$ minus the number of solutions where $x_1 \geq 6$ minus the solutions where $x_2 \geq 6$ and minus the solutions where $x_3 \geq 6$. The number of solutions, by same method as the two previous assignments are:

$$\binom{12}{10}$$

The number of solutions where $x_1 \geq 6$ is

$$\binom{6}{4}$$

The number of solutions where $x_2 \geq 6$ is the same, and for x_3 also. So the total number of ways to do this is then:

$$\binom{12}{10} - \binom{6}{4} \cdot 3 = 66 - 45 = 21$$

. Since that is a very small number of solutions it can be checked by writing them all out:

```
Solutions =
    [[0,5,5],[1,4,5],[1,5,4],[2,3,5],[2,4,4],[2,5,3],[3,2,5],[3,3,4],[3,4,3],[3,5,2],[4,1,5],

    [4,2,4],[4,3,3],[4,4,2],[4,5,1],[5,0,5],[5,1,4],[5,2,3],[5,3,2],[5,4,1],[5,5,0]],
Size = 21.
```

6

a

There are $S(n,m)$, where S denotes Stirling numbers of the second kind, ways to distribute n distinct objects into m identical boxes with a minimum of 1 object in each. Now if we consider that as a set $D = \{x_1, x_2, x_3, x_4\}$ where each variable is a number of balls in containers 1 through 4. When the containers are identical, then one permutation of this set, is equivalent to another. But since the containers are distinct, then the number of possible distributions becomes $S(n,m)$ times the number of permutations of D :

$$S(n,m) \cdot |D|! = S(n,m) \cdot m! = S(12,4) \cdot 4! = 611.501 \cdot 24 = 14.676.024$$

b

This is equivalent to the number of ways to distribute 12 balls among 3 boxes multiplied by the number of boxes. Since it can be done while leaving either box 1, box 2, box 3 or box 4 empty. From the previous assignment, the number of ways to distribute 12 distinct balls among 3 distinct boxes is:

$$S(12,3) \cdot 3! = 519.156$$

And to obtain the total number of ways to distribute 12 distinct balls among 4 distinct boxes while leaving 1 box empty is:

$$519.156 \cdot 4 = 2.076.624$$

c

Then chance of there being a blue ball in each container can be stated as the the number of ways to distribute balls between the 4 containers with at least 1 ball in each, divided by the total number of ways to distribute the balls in the 4 containers. From task a above the number of ways to distribute the 12 balls between the 4 containers with at least 1 ball in each is $S(12,4)$. The total number of ways the balls can be distributed can then be written as

$$S(12,4) + S(12,3) + S(12,2) + S(12,1)$$

i.e. the number of ways to distribute the balls between 4 containers with at least 1 ball in each container. Plus the number of ways to distribute the balls between 3 containers with at least 1 ball in each. And so forth till we reach 1 container. Here by we get the chance that there is at least 1 ball in each container as:

$$\frac{S(12,4)}{S(12,4) + S(12,3) + S(12,2) + S(12,1)} = \frac{611.501}{700.075} \approx 0,873479$$

This is the chance of there being at least 1 blue ball in each container. The chance that there is at least 1 red ball in each container is exactly the same as there are the same number of red balls. The chance of there being at least 1 red ball AND 1 blue ball in each container after 12 red balls and 12 blue balls has been evenly distributed is:

$$\left(\frac{S(12,4)}{S(12,4) + S(12,3) + S(12,2) + S(12,1)}\right)^2 \approx 0,762966$$

7 Appendix

```
import math
translator = {}

def allPermutations(Set): #Generates all permutations of Set
    n = len(Set)
    maxPerms = math.factorial(n)
    Sub = substitute(Set)
    for i in range(maxPerms):
        Ori = reSub(Sub)
        print(Ori,i)
        algorithm1(Sub)
    translator = {}

def allSubsetsInRange(Set,Min,Max): #Generates all subsets of length Min to Max of Set.
    Max = Max + 1           #if only r combinations are wanted use r = Min = Max.
    subtitute = substitute(Set)
    counter = 0
    for i in range(Min,Max):
        subset = firstX(i,subtitute)
        MaxCs = binomial(len(Set),i)
        for c in range(MaxCs):
            ori = reSub(subset)
            print(ori, counter)
            counter = counter + 1
            algorithm3(subtitute,subset)
    translator = {}

def allBitstrings(n): #Generates all bitstring of length n
    String = []
    for i in range(n):
        String.append(0)
    Max = 2 ** n
    for i in range(Max):
        print(String)
        algorithm2(String)

def algorithm1(List): #Generate next permutation in lexicographic order.
    n = len(List)-1
    j = n-1
    while List[j] > List[j+1]:
        j = j - 1
    k = n
    while List[j] > List[k]:
        k = k - 1
    temp = List[j]
    List[j] = List[k]
    List[k] = temp
    r = n
    s = j + 1
    while r > s:
        temp = List[r]
        List[r] = List[s]
        List[s] = temp
        r = r - 1
        s = s + 1
    return List

def algorithm2(BitString): #Generate the next larger bit string.
    i = len(BitString) - 1
    while (BitString[i] == 1) & (i > 0):
        BitString[i] = 0
        i = i - 1
    BitString[i] = 1
    return BitString

def algorithm3(List,After): #Generate the next r-Combination after After in
    lexicographic order.
    r = len(After)-1
    i = r
    while After[i] == List[len(List)-1] - r + i:
        i = i - 1
    After[i] = After[i] + 1
    for j in range (i+1,r+1):
        After[j] = After[i] + (j-i)

def substitute(List): #substitutes members of a list with numbers 1 - len(list). For
    use with algorithm3.
    i = 1
    Sub = []
    for entry in List:
        translator[i] = entry
        Sub.append(i)
        i = i + 1
    return Sub

def reSub(List): #back substitutes the List previously replaced with numbers.
    Original = []
    for entry in List:
        Original.append(translator[entry])
    return Original

def firstX(X,Set): #Gets the first X members of Set.
    subset = []
    for i in range(X):
        subset.append(Set[i])
    return subset

def binomial(n,r): #Calculates the binomial of (n,r).
    bi = (math.factorial(n) / (math.factorial(r) * math.factorial(n - r)))
    return int(bi)
```