An Algorithm to Count onto Functions

Rinku Kumar
M.M.UNIVERSITY
Mullana, Haryana
India

Rakesh Kamboj
M.M. University
Mullana, Haryana
India

Chetan Pahwa
M.M. University
Mullana, Haryana
India

ABSTRACT
This paper proposes an algorithm to derive a general formula to count the total number of onto functions feasible from a set A with cardinality n to a set B with cardinality m. Let f:A→B is a function such that │A│=n and │B│=m, where A and B are finite and non-empty sets, n and m are finite integer values. To count the total number of onto functions feasible till now we have to design all of the feasible mappings in an onto manner, this paper will help in counting the same without designing all possible mappings and will provide the direct count on onto functions using the formula derived in it.

General Terms
Onto Function counting Algorithm.

Keywords
Function, Onto, Cardinality [3], Mappings, transformations, Stirling number [6].

1. INTRODUCTION
The concept of a function is extremely important in mathematics and computer science. For example, in discrete mathematics functions are used in the definition of such discrete structures as sequences and strings. Functions are also used to represent how long it takes a computer to solve problems of a given size. Many computer programs and subroutines are designed to calculate values of functions.

In many instances we assign to each element of a set a particular element of a second set For example, suppose that each student in a discrete mathematics class is assigned a letter grade from the set {A, B, C, D}, and suppose the grades are A for ST1, B for ST2, C for ST3, A for ST4, C for ST5. This assignment of grades is shown below:

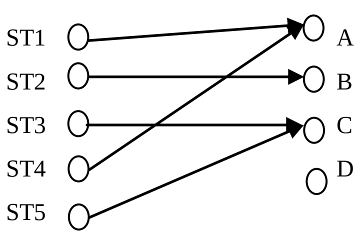

Figure 1: Assignment of grades to students

Here

f(ST1)=A, f(ST2)=B, f(ST3)=C, f(ST4)=A, f(ST5)=C

Definition: Let *A* and *B* be two nonempty sets. A *function f* from *A* to *B* is an assignment of exactly one element of *B* to each element of *A.* We write *f (a)* = *b* if *b* is the unique element of *B* assigned by the function *f* to the element *a* of *A.* If *f* is a function from *A* to *B,* we write

f: A→B. In general f (a) =b, ∇ a∈A, b∈ B

Functions are sometimes called mappings or transformations [2] also.

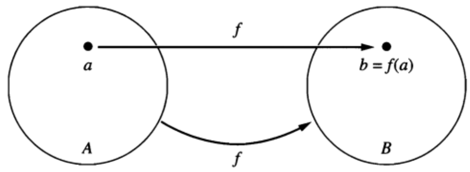

Figure 2: Function f maps A to B

There are number of functions feasible which can be applied to different realistic situations based on the criteria used for mapping the objects to other objects. Some of the types of function mostly used in computing systems are into, onto, one-one, bijective and the inverse functions. We are hereby specifying a method to count all of the object mapping available in an onto manner.

2. RELATED WORK
Onto functions are used mostly in situations where we have to map all of the available objects of a set to objects in other sets, like in job assignment to processors we have to assign each and every job to one of the processor if the number of processors is less than the number of jobs available then no assignment is feasible in an onto manner since we have to assign more than one jobs to single processor to allocate processor to each and every job but a single processor can execute one job at a time, so we must have number of processors to be at least equal to the number of jobs.

2.1 Onto Function
Here every element of set B is assigned to at least one of the element of set A.

A function f from A to B is called onto, or surjective [2], if and only if for every element b ∈ B there is an element a ∈ A with f (a) = b. A function f is called a surjection if it is onto.

Let A and B are two finite sets with cardinality n and 2 respectively i.e. │A│=n and │ B │=2, where n is a finite integer number and f be a function from set A to B i.e.

$$f: A \rightarrow B$$

Suppose TNOF be the total number of onto functions feasible from A to B, so our aim is to calculate the integer value TNOF.

For function f: A→B to be onto, the inequality │A│≥2 must hold, since no onto function can be designed from a set with cardinality less than 2 where 2 is the cardinality of set B.

Case 1: if │A│=│B│=2
A= {a, b} and B= {1, 2}
Following onto functions can be designed

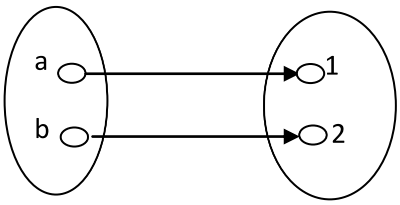

Figure 3: onto function

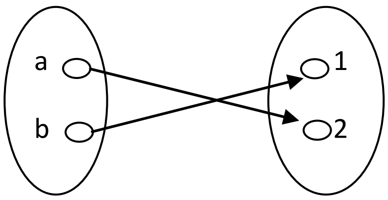

Figure 4: onto function

Besides these two functions any other onto functions can't be designed on the given sets, so for case 1

TNOF=2

Case2: if │A│=3 and │B│=2
Let A= {a, b, c} B= {1, 2}
Here onto functions can be designed as given below:

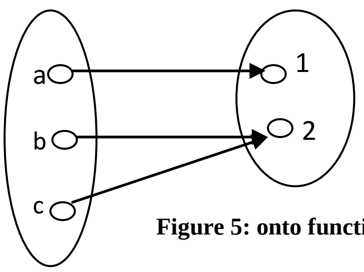

Figure 5: onto function

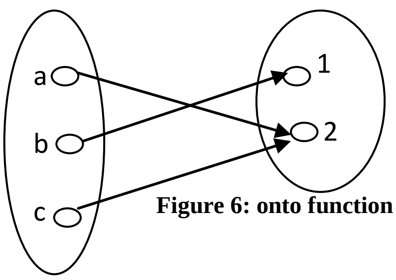

Figure 6: onto function

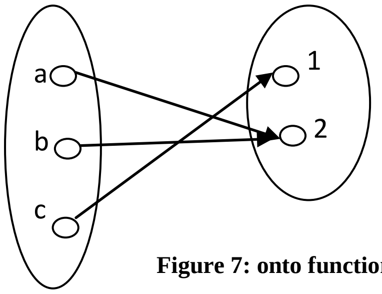

Figure 7: onto function

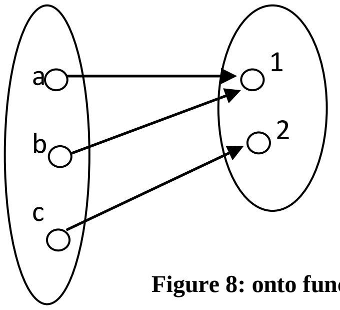

Figure 8: onto function

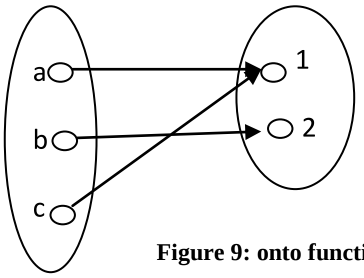

Figure 9: onto function

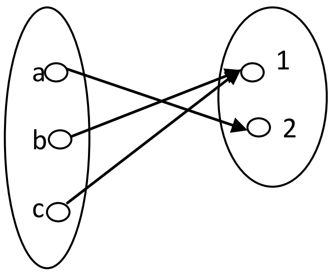

Figure 10: onto function

So, TNOF=6.

similarly, we can find onto functions feasible between two finite and non-empty sets by designing all of the feasible mappings in an onto manner defined above, this method is simple enough if the values of n and m are small what if values of n and m are much large, then it is not feasible to map all such mappings by hand, so we propose an algorithm to find all such feasible functions i.e. to count total number of onto functions feasible.

3. PROPOSED ALGORITHM

Because it is not feasible all the time to perform all such mappings and count them to calculate the count for feasible onto functions, following algorithm is proposed to count TNOF.

3.1 ALGORITHM STEPS

Let $f: A \rightarrow B$ *such that* |A|=n and |B|=m

Check whether

1. $n < m$, if yes then no onto function can be designed.
2. Elseif
   $n = m$ then onto function is feasible and TNOF is calculated as
   - 1st element of A can be mapped to m elements of B
   - 2nd element of A can be mapped to m-1 elements of B
   - 3rd element of A can be mapped to m-3 elements of B, and so on

   So total number of feasible mappings is $m \cdot (m-1) \cdot (m-2) \cdot (m-3) \dots \cdot 1 = m!$
   ….......... **(1)**

   Hence $TNOF = m!$, so, $m!$ onto functions can be designed form A to B

3. Else if
   $n > m$, then we have to map n elements of A to m elements of B,
   - Create partition of set A into m blocks, and consider each block as a single element in A.
   - Now the problem will be simple to map m blocks of A to m elements of B such that all elements in a single block will be assigned the same element from set B

- So TNOF=$m! \cdot P$ …………………………................ **(2)**

Where P is the no of ways of partitioning (i.e. total number of feasible partitioning) of A into m non-empty blocks.

P can be calculated by simply using Stirling number of second kind, i.e. Number of ways of partitioning of A with n elements into m nonempty blocks are given by

$$P = S\left\{\begin{matrix} n \\ m \end{matrix}\right\} \; or \; \left\{\begin{matrix} n \\ m \end{matrix}\right\},$$

Where S is Stirling number of second kind such that

$$S\left\{\begin{matrix} n \\ m \end{matrix}\right\} = \sum_{j=1}^{m} (-1)^{m-j} \frac{j^{n-1}}{(j-1)! \cdot (m-j)!} \quad \dots \textbf{(3)}$$

4. So total number of onto functions are:

$$TNOF = m! \cdot \sum_{j=1}^{m} (-1)^{m-j} \frac{j^{n-1}}{(j-1)! \cdot (m-j)!}$$

.... **(4)**

Hence the value of TNOF will give the count for all feasible onto functions from A to B.

4. RESULT DISCUSSION

Here we show some of the calculated values form the proposed algorithm and compare them with the actual feasible mappings.
Let
$f: A \rightarrow B$ Such that |A|=n and |B|=m such that $n \geq m$

Case 1: let n=m=2

Feasible mappings or onto functions are shown in figure3 and figure 4, i.e. TNOF=2

Now using equation (4) for above used values of n and m, we have

$$TNOF = 2! \cdot \sum_{j=1}^{2} (-1)^{2-j} \frac{j^{2-1}}{(j-1)! \cdot (2-j)!}$$

$$TNOF = 2! \cdot 1 = 2$$

So the result matches with actual mappings.

Note: Since if n=m then 2nd part of calculation i.e. $S\left\{\begin{matrix} n \\ m \end{matrix}\right\}$=1, so, $TNOF = m!$

Case 2: if n>m,

Let n=3, m=2, form actual mapping created in section 2 and we have TNOF=6. Now using equation 4 we have :

m → n ↓	**1**	**2**	**3**	**4**	**5**
1	1	0	0	0	0
2	1	2	0	0	0
3	1	6	6	0	0
4	1	14	36	24	0
5	1	30	150	240	120

$$TNOF = 2! \cdot \sum_{j=1}^{2} (-1)^{2-j} \frac{j^{3-1}}{(j-1)! \cdot (2-j)!}$$

$$TNOF = 2! \cdot 3 = 6$$

Similarly Let n=4 m=3, we have

$$TNOF = 3! \cdot \sum_{j=1}^{3} (-1)^{3-j} \frac{j^{4-1}}{(j-1)! \cdot (3-j)!}$$

$$TNOF = 3! \cdot 6 = 36$$

So using the above algorithm we can easily find the total number of onto functions from A to B, without performing actual mappings between these two sets. TNOF will infer the actual mappings feasible in onto manner i.e. total number of onto functions feasible. Based on the above algorithm we are hereby giving TNOF values for some values of n and m.

Table 1: TNOF values for function, such that │A│=n and │B│=m

This proposed algorithm works well for finite and not null Values of n and m, also the results are verified with actual Mappings and their count. Now for any combination of n and m we can easily state that how many onto functions are feasible from A to B.

5. CONCLUSION

This paper proposed an algorithm to count the total number of onto functions feasible form a set A with n elements to a set B with m elements. So now we do not have to perform all mappings and then count them, simply put the values for n and m in equation 4 and we will get the count required. The proposed algorithm is valid for all finite and non-empty values of n and m, this can be used in any scenario of association of objects.

6. REFERENCES

[1] Rinku Kumar, Rakesh Kamboj, Chetan Pahwa Functions Feasibility Analysis: Based on Cardinality of Sets”, volume 2, issue 3(Mar. 2012), IJARCSSE.

[2] K. Rosen, Discrete Mathematics and its Applications (6th ed), New York: McGraw-Hill, 2007.

[3] L. Gerstein, Discrete Mathematics and Algebraic Structures, New York: Freeman and Co., 1987.

[4] C L Liu, D P Mohapatra, Elements of Discrete Mathematics, TMH, 2008.

[5] Richard Johnsonbaugh, *Discrete Mathematics*, Prentice Hall, 2008

[6] Ronald L. Graham, Donald E. Knuth, Oren Patashnik (1988) *Concrete Mathematics*, Addison–Wesley, Reading MA. ISBN 0-201-14236-8, p. 244