|  |  |  |
| --- | --- | --- |
| Hasil carian imej untuk UTEM LOGO | FACULTY OF ENGINEERING TECHNOLOGY, UNIVERSITI TEKNIKAL MALAYSIA MELAKA | |
| DISCRETE MATHEMATICS | | |
| BEEC 3413 | SEMESTER 1 | SESI 2021/2022 |
| LAB 5: PERMUTATION | | |
| NAME AND MATRIX NUMBER | 1.RAHMAN KAZI ASHIKUR | B081910450 |
|  |  |
|  |  |
| COURSE | BEEC | |
| DATE | December 24, 2021 | |
| NAME OF INSTRUCTOR | Azman | |
| EXAMINER’S COMMENT | | VERIFICATION STAMP |
| TOTAL MARKS |

1. OBJECTIVES 1.To understand the representation permutation. 2.Able to generate permutation from a set of data.

## EQUIPMENTS

1. Personal Computer.
2. R Software.
3. SYNOPSIS & THEORY
   1. PERMUTATION

Permutation of a set of distinct objects is an ordered arrangement of these objects. for example permutation of *S =* {1, 2, 3}, is the ordered arrangement of 1, 2, 3:.

{1, 2, 3}, {1, 3, 2}, {2, 1, 3}, {2, 3, 1}, {3, 1, 2} and {3, 2, 1}.

**Theorem:**

If *n* is a positive integer and *r* is an integer with 1 *≤ r ≤ n*, then there are

*P* (*n*, *r* ) *= n*(*n −* 1)(*n −* 2) *· · ·* (*n − r +* 1)

*r* -permutations of a set with *n* distinct elements.

* 1. GENERATING PERMUTATION

Any set with *n* elements can be placed in one-to-one correspondence with the set {1, 2, 3, *· · ·* , *n*}. We can list the permutations of any set of *n* elements by generating the permutations of the *n* smallest positive integers and then replacing these integers with the corresponding elements.

One of the algorithm of generating next permutation is using **lexicographic** (or **dictionary**) **ordering**, as shown on the algorithm below:

# **procedure** next permutation(a1 a2 . . . an: permutation of

{1, 2, . . . , n} not equal to n n - 1 . . . 2 1) j := n - 1

**while** aj > aj+1 j := j - 1

# {j is the largest subscript with aj < aj + 1}

k := n

**while** aj > ak

# k := k - 1

{ak is the smallest integer greater than aj to the right of aj} interchange aj and ak

r := n

s := j + 1

**while** r > s

# interchange ar and as r := r - 1

s := s + 1

{this puts the tail end of the permutation after the jth position in increasing order}

{a1 a2 . . . an is now the next permutation}

To generate all the possible permutation can use the following algorithm:

# **procedure** permutation(a1 a2 . . . an: permutation of

{1, 2, . . . , n} not equal to n n - 1 . . . 2 1)

{Let x as an array dim(r , n), where r := n!} x[1,] := a1 a2 . . . an

**for** i = 2 to r

# x[i,] := nextpermutation(x[i-1,])

{x1 x2 . . . xr is now all the generated permutation}

## PROCEDURE

* 1. PERMUTATION

1. Open R Gui, and create a new script.
2. Use knowledge from previous chapter write the factorial function.
3. Create new function name **nextpermutation** based on the algorithm above. 4.Create new function name **permutation** based on the algorithm above.

5.Create two sets *A =* {1, 2, 3} and *B =* {3, 4, 5, 6}.

6.Run the function permutation(A) and permutation(B).

## RESULT

1. Follow all the procedures
2. copy paste all your source codes and output as a result.

#2.Use knowledge from previous chapter write the factorial function

Graphical user interface, text

Description automatically generated

Graphical user interface, text, application

Description automatically generated

Graphical user interface, text, application, email

Description automatically generated

Output

Table

Description automatically generated with medium confidence

## DISCUSSION

**A permutation is an ordered arrangement of objects from a group without repetitions. ... Use the Permutation function to find the number of permutations of n items chosen k at a time. Permutations are used to calculate the probability of an event in an experiment with only two possible outcomes (binomial experiment).**

**Permutation is about how many ways to arrange certain set of number. The total of arrangement can be determined by simply take the number of element inside the set and factorial it. For example**

**A = (1 ,2,3)**

**Total number permutation of A is 3! = 6 (1,2,3), (1,3,2), (2,1,3), (2,3,1), (3,1,2), (3,2,1)**

**Using the coding, we can automate the permutation process by filling the matrices that we initiate with the permutation array each row. So we can see the number of row is equal to factorial of the number of element inside the set**

## CONCLUSION

**From the lab we able to understand the number of permutation by taking the factorial number of element inside the set. We also learn how to automate the permutation process to further understanding of the permutation topic.**

## REFERENCE

* The R Manuals, <http://www.r-project.org/>

**Code of file**

|  |
| --- |
| #Factorial fuction  fact<-function(n)  {  if (n==0)    {  return(1)    }  else    {  return(n\*fact(n-1))    }  }  #3.Create new function name nextpermutation based on the algorithm above  nextpem<-function(x)  {  n = length(x)  j = n-1  while (x[j] > x[j+1]) {  j = j- 1    }  k = n  while (x[j]>x[k]) {  k = k -1    }  b = x[j]  x[j] = x[k]  x[k] = b    r = n  s = j+1  while (r>s) {  e = x[r]  x[r] = x[s]  x[s] = e  r = r -1  s = s+1    }  return(x) }  #4.Create new function name permutation based on the algorithm above.  pem<-function(a)  {  n = length(a)  r = fact(n)  x <- array(dim = c(r,n))  for (i in 1:n) {  x[1,i] = a[i]    }  for (i in 2:r) {  x[i,] = nextpem(x[i-1,])  }  return(x)  }  # 5.Create two sets A = {1, 2, 3} and B = {3, 4, 5, 6}.  A = c(1, 2, 3)  B = c(3, 4, 5, 6)  #6.Run the function permutation(A) and permutation(B).  pem(A)  pem(B) |