Permutation of Dynamically Sized Array

- Prafullkumar P. Tale
Integrated M.Sc. in Applied Mathematics
IIT Roorkee

Simple things should be simple…

Simple things should be simple…

Input :

- X1 = [11,22,33]
- X2 = [44,55]

Simple things should be simple…

Input :

- X1 = [11,22,33]
- X2 = [44,55]

Output

- (11,44) ;(11,55);(22,44);(22,55);(33,44);(33,55)

What’s Big deal??

- Use two ‘for’ loops

What’s Big deal??

- Use two ‘for’ loops

Problem is -

The size of arrays is not fixed nor does the number of arrays

What’s Big deal??

- Use two ‘for’ loops

Problem is -

The size of arrays is not fixed nor does the number of arrays

X1 = [11,22,33 ..]

X2 = [44,55, ..]

:

:

What’s Big deal??

- Use two ‘for’ loops

Problem is -

The size of arrays is not fixed nor does the number of arrays

X1 = [11,22,33 ..]

X2 = [44,55, ..]

:

:

Not infinity just unknown or dynamic

Little math..

Little math..

- a $\approx$ b mod (c)

  c divides (b-a)

Little math..

- a $\approx$ b mod (c)
  c divides (b-a)
- f(x) $\approx$ 0 mod(n)
  p1, p2, p3 prime factors of ‘n’

Little math..

- a $\approx$ b mod (c)
  c divides (b-a)
- f(x) $\approx$ 0 mod(n)
  p1, p2, p3 prime factors of ‘n’
- f(x) $\approx$ 0 mod (p1^e1)

  f(x) $\approx$ 0 mod (p2^e2)

  :

  :

Little math..

- a $\approx$ b mod (c)
  c divides (b-a)
- f(x) $\approx$ 0 mod(n)
  p1, p2, p3 prime factors of ‘n’
- f(x) $\approx$ 0 mod (p1^e1) --> X1

  f(x) $\approx$ 0 mod (p2^e2) -- > X2

  :

  :

The Clock…

The Clock…

- 10:58:59 am
  10:59:00 am

The Clock…

- 10:58:59 am
  10:59:00 am

- 10:59:59 am
  11:00:00 am

The Clock…

- 10:58:59 am
  10:59:00 am

- 10:59:59 am
  11:00:00 am

We can use the same concept to make permutations

- X = X1 + X2 + …

- L = saving lower limits
- U = saving upper limits

Example

- X1 = [10,20,30]   X2 = [40,50]
  X3 = [60,70,80]

Example

- X1 = [10,20,30]   X2 = [40,50]
  X3 = [60,70,80]
- X = X1 + X2 + X3
  X = [10,20,30,40,50,60,70,80]

Example

- X1 = [10,20,30]   X2 = [40,50]
  X3 = [60,70,80]
- X = X1 + X2 + X3
  X = [10,20,30,40,50,60,70,80]

- L = [1,4,6]
- U = [3,5,8]

Algorithm

L = [1,4,6]   U = [3,5,8]

Algorithm

L = [1,4,6]   U = [3,5,8]

- I = L = [1,4,6]

Algorithm

L = [1,4,6]   U = [3,5,8]

- I = L = [1,4,6]
  [1,4,7]

Algorithm

L = [1,4,6]   U = [3,5,8]

- I = L = [1,4,6]

  [1,4,7]

  [1,4,8]

Algorithm

L = [1,4,6]   U = [3,5,8]

- I = L = [1,4,6]

  [1,4,7]

  [1,4,8] ---> Sec place is over

Algorithm

L = [1,4,6]   U = [3,5,8]

- I = L = [1,4,6]

  [1,4,7]

  [1,4,8] ---> Sec place is over

  [1,5,6]

Algorithm

L = [1,4,6]   U = [3,5,8]

- I = L = [1,4,6]

  [1,4,7]

  [1,4,8] ---> Sec place is over

  [1,5,6]

  [1,5,7]

Algorithm

L = [1,4,6]   U = [3,5,8]

- I = L = [1,4,6]

    [1,4,7]

    [1,4,8] ---> Sec place is over

    [1,5,6]

    [1,5,7]

    [1,5,8]

Algorithm

L = [1,4,6]   U = [3,5,8]

- I = L = [1,4,6]

  [1,4,7]

  [1,4,8] ---> Sec place is over

  [1,5,6]

  [1,5,7]

  [1,5,8] --- > Min place is over

Algorithm

L = [1,4,6]   U = [3,5,8]

- I = L = [1,4,6]

  [1,4,7]

  [1,4,8] ---> Sec place is over

  [1,5,6]

  [1,5,7]

  [1,5,8] --- > Min place is over

  [2,4,6]

Algorithm

L = [1,4,6]   U = [3,5,8]

- I = L = [1,4,6]

  [1,4,7]

  [1,4,8] ---> Sec place is over

  [1,5,6]

  [1,5,7]

  [1,5,8] --- > Min place is over

  [2,4,6]

Algorithm

L = [1,4,6]   U = [3,5,8]

- I = L = [1,4,6]

  [1,4,7]

  [1,4,8] ---> Sec place is over

  [1,5,6]

  [1,5,7]

  [1,5,8] --- > Min place is over

  [2,4,6]

  [2,4,7]

Algorithm

L = [1,4,6]   U = [3,5,8]

- I = L = [1,4,6]

  [1,4,7]

  [1,4,8] ---> Sec place is over

  [1,5,6]

  [1,5,7]

  [1,5,8] --- > Min place is over

  [2,4,6]

  [2,4,7]

  :

  I = U