minPrefixGen.py

Description: This module takes the range of query and the number of bits to represent the max element in that range and returns the minimum prefix set which can represent all the values in the given range.

Input: Data (Range of the Query from the data user's end) and Bits (# bits to represent the max value)

Complexity: $O(n^2)$ (n = the number of elements in the range)

Output: minPrefixSet which will represent all the elements in the specified range.

Code:

```
preSet = [] # list which stores the prefix Set
unionPreSet = [] # Takes only the commons and removes redundancies from preSet[]
starList, numList = [], [] # These are the lists that store elements with/without Stars respectively
finList = [] # Stores the intermediate result

# This function joins the elements in the list
def joinList(l):
    return ''.join(l)

# This function prints the starlist
def retList(starList):
    print(starList)

# This fucntion removes the element(s) from the starlist
def remFromList(starList, tempArray):
    if joinList(tempArray) in starList:
        starList.remove(joinList(tempArray))

# This function creates two separate lists 1. with "*" and 2. without "*"
def separateLists(unionPreSet):
    for x in unionPreSet:
        if '*' in list(x):
            starList.append(x)
        else:
            numList.append(x)
    return starList, numList

# This is the function which generates the minimum Prefix set
# used in the tree building process!
def minPrefix(unionPreSet):
    finList = []
    unionPreSet = sorted(unionPreSet)
    starList, numList = separateLists(unionPreSet)
    # print(starList, numList)

# This function takes the binary values of numbers in the range
# and then converts it to prefix set and appends all the values to preSet List!
def createPrefix(s, preSet):
    temp = []  # temp list to generate the Prefix Set of a binary value
    temp.append(s)
    s = list(s)
    for x in range(1, len(s) + 1):
        s[-x] = '*'
```

```
            temp.append(''.join(s))
        preSet += temp

# This function will replace the first occuring star with 1 and 0 and will
# restrict the elements in the starlist
def rangeRestrict(starList):
    for s in starList[:]:
        s = list(s)
        t0 = s[:]
        t1 = s[:]
        t0 = [x.replace("*", '0') for x in list(t0)]
        t1 = [x.replace("*", '1') for x in list(t1)]
        t0 = joinList(t0)
        t1 = joinList(t1)

        if t0 in numList:
            if t1 in numList:
                pass
            else:
                remFromList(starList, s)
        else:
            remFromList(starList, s)

    return starList


sttt = ""
# This function checks the element which has more number of stars in it.
def checkMaxStars(temp):
    res = ""
    global sttt

    num = 0
    for x in temp:
        x = list(x)

        if num < x.count("*"):
            num = x.count("*")
            res = joinList(x)
            sttt = res
    return res

# This is used to extract the prefix of the maxStarElement
def removeStars(maxStarElement):
    return joinList(list(filter(("*").__ne__, maxStarElement)))

what = []

# This will over lap the modified maxStarElement with other replaceElements
# if there is an overlap then it removes that element from the list as the
# maxStarElement can represent it when permuted with '0' or '1'
def replaceElements(listt):
    temp = listt[:]
    global what
```

```
    ele = checkMaxStars(temp) # returns element with max stars in the list
    maxStarElement = list(ele)
    if ele == "":
        what = finList + temp

    else:
        finList.append(joinList(maxStarElement))
        maxStarElement = removeStars(maxStarElement)
        length = len(list(maxStarElement))

        for element in listt[:]:
            if len(listt) == 0:
                print("Jasdhhkasdjhaskm")
                break
            else:
                if maxStarElement[:] in element[:length]:
                    listt.remove(element)

# Main point of execution which takes the range of data and the number of bits
# and returns the minPrefixSet of the range of data
def main(data, Bits):
    r = data
    global what
    global finList

    string = '{0:0' + str(Bits) + 'b}'
    if r[0] == "0" and r[1] == "1":
        what = ['*']
    else:
        for n in range(r[0], r[-1]+1):
            ## This function call creates the prefix set and stores in preSet list[]
            createPrefix(string.format(n), preSet)
        unionPreSet = list(set(preSet))
        j = list(sorted(unionPreSet))
        minPrefix(j)
        newList = rangeRestrict(starList)
        newList = starList + numList
        lenStar = len(starList)

    for x in range(lenStar):
        replaceElements(newList)
    return finList + what + newList # this combination has the minPrefixSet

""" CAN BE USED TO TEST THIS CODE!"""
# r = [1,10]
# print(main(r, 4))
```