Roll: 32

Jahir Sadik Monon

Fundamentals of Genomics and Proteomics Lab (Task 4)

Department of Computer Science and Engineering, University of Dhaka

Problem statement: Write and algorithm to solve partial digest problem.

Find X such that ΔX = L

For example L = {2, 2, 3, 3, 4, 5, 6, 7, 8, 10}.

Solve for L (i.e. find X such that ΔX = L).

Problem solution:

```
# returns the absolute differences from LMax to elements of set X
def CalculateDifferences(LMax, X):
    difference = []
    for point in X:
        difference.append(abs(LMax-point))
    return difference

# remove from L, the new distances created after puting point nPoint
def removeElements(nPoint, X, L):
    for point in X:
        if abs(nPoint - point) in L:
            L.remove(abs(nPoint - point))

# checks if the set {abs(LMax - {elements of X})} is a subset of L
def isSubsetofL(LMax, X, L):
        for point in X:
            if abs(LMax-point) not in L:
                return False
        return True

def insertPoint(L, X, MAX_VAL):
    # if L has no elements, we have found a potential solution
    if len(L) == 0:
        print ("One possible position of the points is", end = ":\t")
        print(X)
        return
    LMax = max(L) # take the maximum value in the list L
    # see if placing the maximum value in right side works by checking if
    #       {abs(LMax - {elements of X})} is a subset of L
    if isSubsetofL(LMax, X, L):
        X.append(LMax)
        # remove elements from
        removeElements(LMax, X, L)
        insertPoint(L, X, MAX_VAL)
        # this subtree is checked, so add the points back & remove LMax from X
        if LMax in X:
            X.remove(LMax)
        # add back to L, the elements removed in removeElements()
        L.extend(CalculateDifferences(LMax, X))

    # Same approach as before but after placing the cut on the left
    #       by taking abs(MAX_VAL-LMax)
    if isSubsetofL(abs(MAX_VAL-LMax), X, L):
        X.append(abs(MAX_VAL-LMax))
        removeElements(abs(MAX_VAL-LMax), X, L)
        insertPoint(L, X, MAX_VAL)
        if abs(MAX_VAL-LMax) in X:
```

```
            X.remove(abs(MAX_VAL-LMax))
        L.extend(CalculateDifferences(abs(MAX_VAL-LMax), X))
    return

# the core function to solve the partial digest problem
def partialDigest(L, X, MAX_VAL):
    MAX_VAL = max(L)
    # we can initially remove the maximum value without changing anything
    L.remove(MAX_VAL)
    # initialize the X array with two values, 0 and max val
    X = [0, MAX_VAL]
    insertPoint(L, X, MAX_VAL)


# set the variables to default values and call the function
X = []
MAX_VAL = 0
# input
L = [2, 2, 3, 3, 4, 5, 6, 7, 8, 10]
partialDigest(L, X, MAX_VAL)
```