Experiment No. 8

Aim - Implementation of sum of subjects.

Theory -

Algo -

- So if we take example as int [] A = {3,2,7,1}, s = 6
- If we consider another int array with the same size as A
- If we include the element in subset we will put 1 in that particular index else put 0.
- So we need to make every possible subsets & check if ony of the subsets makes sum s.

Example -

Input - set[] = {3,34,4,12,5,2}, sum = 9

Output - True

There is a subset (4,5) with sum 9.

Time Complexity -

The above soln may try as subsets of given set in wo case. Therefore, time complexity of the above soln is exponential.

Conclusion -

We have successfully implemented the algo & analyse the complexity.

pra8.py ●

pra8.py > isSubsetSum

```python
def isSubsetSum(set, n, sum) :

    if (sum == 0) :
        return True
    if (n == 0 and sum != 0) :
        return False

    if (set[n - 1] > sum) :
        return isSubsetSum(set, n - 1, sum);

    return isSubsetSum(set, n-1, sum) or isSubsetSum(set, n-1, sum-set[n-1])

set = [3, 34, 4, 12, 5, 2]
sum = 9
n = len(set)
if (isSubsetSum(set, n, sum) == True) :
    print("Found a subset with given sum")
else :
    print("No subset with given sum")

```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

Python

```
/usr/bin/python3.9 "/home/computer/Desktop/aoa pra/pra8.py"
(base) computer@computer:~/Desktop/aoa pra$ /usr/bin/python3.9 "/home/computer/Desktop/aoa pra/pra8.py"
Found a subset with given sum
(base) computer@computer:~/Desktop/aoa pra$
```

LTCE

Experiment No. 9

Graph Colouring

Aim - To implement graph colouring.

Theory -

Algo -

1) Colour 1st vertex with first color.
2) Do following for remaining v - vertices.
a) Consider the following picked vertex & color it with the lowest numbered color that has not been used on any previously colored vertices adjacent to it. If all previously used colors appear on vertices adjacent to V, assign a new color to it.

Time complexity -

$O(v^2 + E)$ is the time complexity in wrost case.

Conclusion -

We have successfully implemented the algo & analyzed the complexity.

Activities Visual Studio Code Tue 1:51 PM

pra9.py - aoa pra - Visual Studio Code

File Edit Selection View Go Run Terminal Help

pra8.py pra9.py

pra9.py > greedyColoring

```
def addEdge(adj, v, w):

    adj[v].append(w)
    adj[w].append(v)
    return adj

def greedyColoring(adj, V):
    result = [-1] * V
    result[0] = 0;
    available = [False] * V
    for u in range(1, V):

        for i in adj[u]:
            if (result[i] != -1):
                available[result[i]] = True
        cr = 0
        while cr < V:
            if (available[cr] == False):
                break

            cr += 1
        result[u] = cr
        for i in adj[u]:
            if (result[i] != -1):
                available[result[i]] = False
    for u in range(V):
        print("Vertex", u, " ---> Color", result[u])
if __name__ == '__main__':
    g1 = [[] for i in range(5)]
    g1 = addEdge(g1, 0, 1)
    g1 = addEdge(g1, 0, 2)
    g1 = addEdge(g1, 1, 2)
    g1 = addEdge(g1, 1, 3)
    g1 = addEdge(g1, 2, 3)
    g1 = addEdge(g1, 3, 4)
    print("Coloring of graph 1 ")
    greedyColoring(g1, 5)
    g2 = [[] for i in range(5)]
    g2 = addEdge(g2, 0, 1)
    g2 = addEdge(g2, 0, 2)
    g2 = addEdge(g2, 1, 2)
    g2 = addEdge(g2, 1, 4)
    g2 = addEdge(g2, 2, 4)
    g2 = addEdge(g2, 4, 3)
    print("\nColoring of graph 2")
    greedyColoring(g2, 5)
```

Ln 8, Col 28 Spaces: 4 UTF-8 LF Python 3.9.16 64-bit

LTCE

Experiment No. 10

Rabin-Karp Algo

Aim - To implement Rabin-Karp Algo

Theory -

Algo -

Rabin-Karp Matcher (T, P, d, q)

1. $n \leftarrow$ length [T]
2. $m \leftarrow$ length [P]
3. $h \leftarrow d^{m-1} \bmod [q]$
4. $p \leftarrow 0$
5. $t_0 \leftarrow 0$
6. for $i \leftarrow 1$ to m
7. $t_0 \leftarrow (d t_0 + P[i]) \bmod q$
8. $t_0 \leftarrow (d t_0 + T[i]) \bmod q$
9. for $s \leftarrow 0$ to $n-m$
10. ~~then if~~ do if $p = t_3$
11. Then if $P[1 \ldots m] = T[s+1 \ldots s+m]$
12. then "Pattern occurs with shift" s
13. If $s < n-m$
14. then $t_{s+1} \leftarrow (d(t_s - T[s+1]h) + T[s+m+1]) \bmod q$.

Example -

Input - txt [] = "THIS IS A TEST TEXT"

pat [] = "TEST"

Output - Pattern found at index 10.

Time Complexity -

The average & best-case running time of the algo is $O(n+m)$, but in worst case is $O(nm)$.

Conclusion -

We have successfully implemented algo & analysed the complexity.

Activities Visual Studio Code Tue 2:02 PM

pra10.py - aoa pra - Visual Studio Code

File Edit Selection View Go Run Terminal Help

pra10.py

```python
d = 256
def search(pat, txt, q):
    M = len(pat)
    N = len(txt)
    i = 0
    j = 0
    p = 0
    t = 0
    h = 1
    for i in range(M-1):
        h = (h * d)% q
    for i in range(M):
        p = (d * p + ord(pat[i]))% q
        t = (d * t + ord(txt[i]))% q
    for i in range(N-M + 1):

        if p == t:
            for j in range(M):
                if txt[i + j] != pat[j]:
                    break

            j+= 1
            if j == M:
                print ("Pattern found at index " + str(i))

        if i < N-M:
            t = (d*(t-ord(txt[i])*h) + ord(txt[i + M]))% q
            if t < 0:
                t = t + q

txt = "GEEKS FOR GEEKS"
pat = "GEEK"
q = 101 # A prime number
search(pat, txt, q)
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```
/usr/bin/python3.9 "/home/computer/Desktop/aoa pra/pra10.py"
(base) computer@computer:~/Desktop/aoa pra$ /usr/bin/python3.9 "/home/computer/Desktop/aoa pra/pra10.py"
Pattern found at index 0
Pattern found at index 10
(base) computer@computer:~/Desktop/aoa pra$
```

Ln 35, Col 20 Spaces: 4 UTF-8 LF Python 3.9.16 64-