# CSA0888. - PYTHON PROGRAMMING

**Assignment-3**

**Program 1:**

defmaxProfit(price, n): profit=[0]\*n max\_price=price[n-1] for i inrange(n-2, 0,-1):

ifprice[i] > max\_price: max\_price=price[i]

profit[i] =max(profit[i+1], max\_price- price[i])

min\_price =price[0] for i inrange(1, n):

ifprice[i] <min\_price: min\_price=price[i]

profit[i] =max(profit[i-1], profit[i]+(price[i]-min\_price)) result=profit[n-1]

return result

price=[2,7,9,56,7,4]

print("Maximumprofitis", maxProfit(price, len(price)))

# Program 2:

fromitertoolsimport permutations comb=permutations([4, 5, 6], 3) fori incomb:

print(i)

# Program. 3

defsolve(nums): count=0 n=len(nums) fori inrange(n):

for j in range(i+1,n):

ifnums[i] ==nums[j]: count+=1

returncount

nums=[5,6,7,5,5,7]

print(solve(nums))

# Program.4

defadd\_binary\_nums(x, y):

max\_len =max(len(x), len(y))

x=x.zfill(max\_len) y=y.zfill(max\_len)

result ='' carry= 0

fori inrange(max\_len - 1,-1, -1): r=carry

r+= 1 if x[i] =='1'else0

r+= 1 if y[i] =='1' else 0

result =('1' ifr %2 == 1 else '0') +result

carry= 0 if r< 2 else1

if carry !=0 : result = '1' + result return result.zfill(max\_len)

print(add\_binary\_nums('1101', '100'))

# Program 5

defminJumps(arr, n):

if (n<= 1):

return 0

if (arr[0] == 0):

return -1

jump = 1

subArrEndIndex= arr[0] i= 1

subArrFistHalfMaxSteps = 0

subArrSecondHalfMaxSteps = 0

fori inrange(1, n):

subArrEndIndex= i +subArrEndIndex if (subArrEndIndex >=n):

return jump

firstHalfMaxStepIndex= 0 j= i

forjinrange(i, subArrEndIndex): stepsCanCover=arr[j] + j

if (subArrFistHalfMaxSteps <stepsCanCover):

subArrFistHalfMaxSteps =stepsCanCover subArrSecondHalfMaxSteps = 0 firstHalfMaxStepIndex= j

elif(subArrSecondHalfMaxSteps<stepsCanCover): subArrSecondHalfMaxSteps =stepsCanCover

i= j

if (i> subArrFistHalfMaxSteps): return -1

jump += 1

subArrEndIndex= arr[firstHalfMaxStepIndex] subArrFistHalfMaxSteps =subArrSecondHalfMaxSteps

return -1

if name ==' main ':

arr=[1, 3, 5, 8, 9, 2, 6, 7, 6, 8, 9]

size =len(arr)

print("Minimum numberof jumps to reach endis", minJumps(arr, size))

# Program 6

sentence=input("Enterasentence: ") s=sentence.split()

s.sort()

result=[] n=len(s)

fori inrange(n):

for j in range(i+1,n): two\_words=s[i]+""+s[j] result.append(two\_words)

foritemin result: print(item)

# Program 7

defbacktrack():

global ans, curr, visited, nums

if (len(curr) ==len(nums)): print(\*curr)

for i inrange(len(nums)): if(visited[i]):

continue

if(i > 0 and nums[i] ==nums[i - 1] andvisited[i - 1]==False): continue

visited[i] =True

curr.append(nums[i])

backtrack()

visited[i] visited[i] =False

del curr[-1]

defpermuteDuplicates(nums): global ans, visited, curr nums=sorted(nums)

backtrack() returnans

defgetDistinctPermutations(nums): global ans, curr, visited

ans=permuteDuplicates(nums) if name ==' main ':

visited=[False]\*(5) ans,curr =[], [] nums=[1, 2, 3]

getDistinctPermutations(nums)

# Program 8

fromcollectionsimport Counter, defaultdict user\_input =["cat", "dog", "tac","edoc", "god", "tacact",

"act","code", "deno","node", "ocde", "done","catcat"]

defsolve(words: list) -> list: m=defaultdict(list)

for wordinwords:

frozenset(dict(Counter('cat')).items()):

hash(frozenset(Counter('cat')))is equal to frozenset({('c', 1),('a', 1), ('t', 1)})

m[frozenset(dict(Counter(word)).items())].append(word)

return[v fork, vinm.items()]

print(solve(user\_input))

# Program 9

deffinding(s, p, n, m):

# return 1 if nand mare negative if n< 0 andm< 0:

return 1

# return 0 ifmisnegative ifm< 0:

return 0

# returnnifnisnegative if n< 0:

# whilemispositive

while m>= 0: ifp[m]!='\*':

return 0

m-= 1

return 1

# ifdpstateisnotvisited if dp[n][m] ==-1:

ifp[m]=='\*':

dp[n][m] =finding(s, p, n- 1, m) orfinding(s, p, n, m-1) return dp[n][m]

else:

ifp[m]!=s[n] andp[m]!='?': dp[n][m] = 0

return dp[n][m] else:

dp[n][m] =finding(s, p, n- 1, m- 1)

return dp[n][m]

# returndp[n][m] if dpstate isprevisited returndp[n][m]

defisMatch(s, p): global dp

dp=[]

# resizethe dp array for i inrange(len(s) + 1):

dp.append([-1]\* (len(p) + 1))

dp[len(s)][len(p)] =finding(s, p, len(s) - 1, len(p)- 1)

returndp[len(s)][len(p)]

defmain():

s="baaabab"

p ="\*\*\*\*\*ba\*\*\*\*\*ab"

if isMatch(s, p): print("Yes")

else:

print("No")

if name ==" main ": main()

# Program. 10

def editDistance(str1, str2, m, n): ifm== 0:

return n

if n== 0: return m

if str1[m-1]==str2[n-1]:

return editDistance(str1, str2, m-1, n-1)

return 1 +min(editDistance(str1, str2, m, n-1), editDistance(str1, str2, m-1, n), editDistance(str1, str2, m-1, n-1)

)

str1 ="cut" str2 ="cat"

print(editDistance(str1, str2, len(str1), len(str2)))