**Institute of Engineering & Management**

**Department of Computer Science & Engineering**

**Design & Analysis of Algorithm Lab for 3rd year 5th semester 2018**

**Code: CS 591**

**Date:** 29/08/18

**WEEK-3**

**Assignment-1**

**Problem Statement:** KMP String Matching: Given a text txt[0..n-1] and a pattern pat[0..m-1], write a function search(char pat[], char txt[]) that prints all occurrences of pat[] in txt[]. You may assume that n > m.

**Algorithm:**

**Source code:**

#include <iostream>

#include <vector>

void preprocess(std::string &pat, std::vector<int> &vect)

{

vect[0]=0;

int i=1, j=0, n = pat.length();

while(n>i)

{

if(pat[j] == pat[i])

{

vect[i] = j+1;

i++; j++;

}

else{

if(j>0)

j = vect[j-1];

else{

vect[i] = 0;

i++;

}

}

}

}

void kmp(std::string &str, std::string &pat)

{

int i=0, j=0, n=str.length();

std::vector<int> lps(pat.length(), 0);

preprocess(pat, lps);

std::cout<<"The locations of the pattern: ";

while(i<n)

{

if(pat[j] == str[i])

{

if(j == pat.length()-1)

{

std::cout<<i-pat.length()+1<<" ";

j = lps[j];

i++;

}

else{

i++; j++;

}

}

else

{

if(j == 0)

i++;

else{

j = lps[j-1];

}

}

}

std::cout<<std::endl;

}

int main()

{

std::string str, pat;

std::cout<<"Enter the String: ";

std::cin>>str;

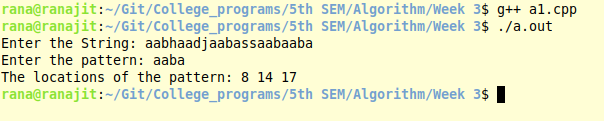
std::cout<<"Enter the pattern: ";

std::cin>>pat;

kmp(str, pat);

}

**Screen-Shot:**

****

**Time Complexity:**

**Source code:**

#include <iostream>

#include <vector>

int main()

{

int n, coin\_no;

std::cout<<"Enter the no. of coin types: ";

std::cin>>coin\_no;

std::vector<int> coins(coin\_no);

std::cout<<"Enter the coin types (space separated): ";

for(auto &i: coins)

std::cin>>i;

std::cout<<"Enter the total change: ";

std::cin>>n;

std::vector<int> mem(n+1, 0);

for(int i=1;i<=n;i++)

{

int minimum = (i<coins[0])? i : mem[i-coins[0]];

for(int j=1;j<coin\_no;j++)

minimum = std::min(minimum, ((i<coins[j])? i : mem[i- coins[j]]));

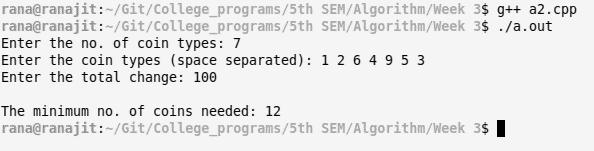
mem[i] = minimum+1;

}

std::cout<<"\nThe minimum no. of coins needed: "<<mem[n]<<std::endl;

}

**Screen-Shot:**

****

**Time Complexity:**

**Source code:**

#include <iostream>

#include <vector>

int main()

{

int n, card;

std::cout<<"Enter cardinality of the set: ";

std::cin>>card;

std::vector<int> set(card);

std::cout<<"Enter elements of the set: ";

for(auto &i: set)

std::cin>>i;

std::cout<<"Enter the sum: ";

std::cin>>n;

std::vector<std::vector<bool>> mem(n+1, std::vector<bool>(card+1));

for(auto &i: mem)

i[0] = false;

mem[0] = std::vector<bool>(card+1, true);

for(int i=1;i<=n;i++)

{

for(int j=1;j<=card;j++)

{

mem[i][j] = mem[i][j-1] || ((i<set[j-1])? false : mem[i-set[j-1]][j-1]);

}

}

std::cout<<"\nResult: ";

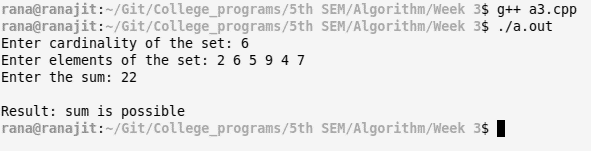
if(mem[n][card]==true)

std::cout<<"sum is possible"<<std::endl;

else std::cout<<"sum is not possible"<<std::endl;

}

**Screen-Shot:**

****

**Time Complexity:**

**Source code:**

#include <iostream>

#include <vector>

int main()

{

int m, n, minimum;

std::cout<<"Enter the size of Matrix: ";

std::cin>>m>>n;

std::vector<std::vector<int>> matrix(m, std::vector<int>(n));

std::cout<<"Enter the Matrix:"<<std::endl;

for(auto &i: matrix)

for(auto &j: i)

std::cin>>j;

for(int i=0;i<m;i++)

for(int j=0;j<n;j++)

{

if(i == 0)

{

if(j == 0)

minimum = 0;

else minimum = matrix[i][j-1];

}

else{

if(j == 0)

minimum = matrix[i-1][j];

else minimum = std::min(matrix[i-1][j-1], std::min(matrix[i][j-1],matrix[i-1][j]));

}

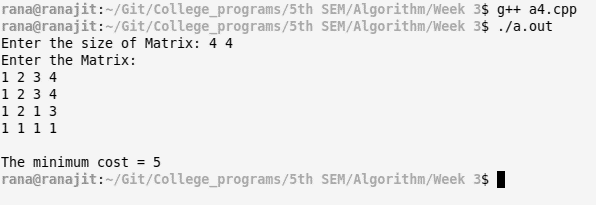
matrix[i][j] += minimum;

}

std::cout<<"\nThe minimum cost = "<<matrix[m-1][n-1]<<std::endl;

}

**Screen-Shot:**

****

**Time Complexity:**