NAME: SURIYAKUMAR J
IT-C

Design Analysis Algorithm

ASSIGNMENT

1. Explain the subset sum problem and provide an algorithm in C++ to solve it using backtracking. Discuss the time and space complexity of your solution

SOL

The subset sum problem is a classic computer science problem that falls under the category of combinatorial optimization

problem statement

Given a set of positive integer and target sum, can you find the subset of integer in set such sum is equal to target sum?

coding

```
# include <iostream>
# include <vector>
using namespace std;
int c=0
int subset (vector<int>& list, int st sum, int idx, int target su
{
 if(st sum == target sum)
   {
    c++;
   }
for (int i=idx ; i< list.size(); i++)
 {
  subset ( list, st sum + list[i], i+1, larger sum);
 }
}
```

```
int main()
{
 int n, target sum;
 cin>>n
vector<int> list(n)
for (int i=0; i<n; i++)
{
  cin>>list[i];
}
cin >> (targetsum;
Subset(list, 0, 0, target sum);
cout<<c;
}
```

1.) Time complexcity

=> In worst case, the algorithm explores all possible subsets of the given set
There are 2^n possible subsets for set of n elements
=> For each subset, the algorithm checks the sum, which is an O(n) operation.
=> Therefore, the worst case time complexity of this algorithm is $O(2^n * n)$

2.) Space complexity

The space complexity is determined by the recursive call stack. In the work case, the depth of the call number of element in the set

=> Therefore, the worst-case complexity is O(n)

2.) Implement a C++ program to find a knight's tours on an 'n×n' chessboard starting from a given initial position. Ensure that the knight visits each square exactly once. Display the solution if possible or indicate if its not solvable.

Sol

```
#include <iostream>
using namespace std;
bool is safe (int a, int b, int, chess[][100], int n)
{
return true
}
return false
}
bool Knight (int a, int b, int n, int count, int chess[][100]
int x[], int y[])
```

```
{if(count ==n*n)
{
return true;
}
for(int i=0; i<8; i++)
{
int p=a+x[i];
int q= b+x[i];
if (is safe(p,q,chess,n))
{chess[p][q]=count;
if (knight(p,q,n,count+1,chess,x,y))
{
return true;
}
else
{
chess[p][q]=-1
}}}
return false
}
int main()
Chess[0][0]=0
int x[]={2,1,-1,-2,-2,-1,1,2}
int y[]={1,2,2,1,-1,-2,-2,-1}
if(knight(0,0,n,1,chess,x,y))
{
cout<<"solution exists:"<<endl
for(int i=0;i<n;i++)
{for(int j=0;j<n;j++)
{
cout<<chess[i][j]<<" "
}
cout<<endl
}else
}} {cout<<"No solution exists
```

Develop C++ program that solves the m- coloring problem for given graph.

sol

```
# include <bits/stdc++.h>
using namespace std;
bool isSafe (int v, int arr [][4], int color [], int index, int c)
{
for ( int i=0; i<4; i++)
  {
  if (arr [index [i] && color [i] == c)
    return false;
  } return true;
}
for (int c=1; c<=m; c++)
{ if (isSafe (v, arr, color, index, c))
    {
     color [index] = c
     if (m color (v, arr, m, color, index+1))
      return true;
      color [index] = 0;
     }}
     return false;
    int main ()
    {
     int arr [4][4];
     for (int i=0 ; i<4 ; i++)
     {for (int j=0; j<4; j++)
      {  cin >> arr [i][j]
      }
    }
```

```
int m;
cin>>m
int color (m)
for (int i=0; i<m; i++)
    color(i)=0
if (m color (4, arr, m, color, 0))
{cout<< "Solution Exists:\n";
for (int i=0; i<4; i++)
  {cout << color(i)<< " ";
}
else
  cout <<"Solution does not exist";
}
```

4.) Design a algorithm to find a Hamitonnian cycle in given graph:

```
#include<iostream>
using namespace std
const int max=0
int graph (max)(max)
vector<int> path;
int v
bool isSafe(int v, int p){
 if (graph (path (p-1)(v)=0)
  {
    return false
  }
 for (int i=0; i<p; i++)
    {
     if (path (i) == v)
     {
       return false
       }
     return true
       }
```

```
if (upde t)){
for (int i=0; i<v; i++)
{
  cout << path(i) << " ";
}
cout << path[0] << endl;
}
else
{
  cout << "path does not exist";
}
```

5.) Create a C++ program to slove sudoku puzzle using a backtracking algorithm

```
#include<iostream>
using namespace
const int v=a;
void print Grid (int grid[N][N]
{
for(int i=0; int j=0; i<N; j<N; i++; j++)
    {
     cout << grid[i][j] << " "
     cout << endl;
  }}

bool (isSafe (int grid [N][N], int row, int col, int
  {
   return false;
}}
```

```
bool hamiltononian (int p)
{
 int if (p==v)
 {
  if(graph[path(p-1)][path(0)]==1)
   return true;
  else
   return false;
 }
 for (int i=0; i<v; i++)
    {if (isSafe(i,p))
     {
      path[p]=i;
     if (hamiltonian(p+1))
      return true;
      path[p]=-1;
     }
    }
 return false
 }
 bool cycle()
 {
  path resize(v,1);
  path(0)=0;
  if (hamiltonian(1)))}
   return false;
  }
  return true;
 int main()
 {
  cin>>v;
  for (int i=0; i<v; i++)
  {
   for (int j=0; j<v; j++)
    {
     cin>> graph(i)(j);
  }}
```

```
(solve sudoku (grid)))
{
cout << " solved" << endl
print Grid (grid);
}
else
{
cout << " No solution" << endl
}
```

6.) Write a c++ program that uses the sieve of sundaram algorithm to find all prime numbers less than or equal to 'n'.

```
#include <iostream>
#include <vector>
using namespace std;
int sieve of sundaram (int n)
int no=(n-1)/2
vector<bool> a (no+1)
for (int i=1 ;i<=no;i++)
  {
    for(int j=1 ; (i+j+2*i*j ) <= no ; j++)
    {
      a[i+j+2*i*j]= true;
    }}
  cout<<"2"
  for (int i=1;i<=no;i++)
    {if(a[i]== false)
      {
        cout<<i+1<<" ";
      }}
int main()
  {int n
  cin>>n
  } sieve of Sundaram(n);
```

7.) Develop a C++ function to find a prime number greater than value 'p' digits its is equal to 's'. Implement using backtracking

```
#include<iostream>
using namespace std;
bool isPrime(int n)
{
if(n<=1)
  return false;
if(n<=3)
   return true;

if(n%2==0|| n%3==0)
    return false;
for(int i=5; i*i<=n; i+=6
{if(n%i==0|| n%(i+2)==0)
  return false
}return true
for(int num=p+1 ; num++)
{   int temp=num
    int digit sum=0
   while(temp>0)
    {
     digit += temp=num
     int digit sum=0
     while (temp>0)
     {
       digit += temp%10
       temp /= 10
     }
```

```
int main()
{
 int p, s
 cin >> p >> s;
  int prime = find primeNumbers (p, s)
  if (prime != -1)
  cout << p << s << " " prime << endl;
 }
}
```