Maximum subarray sum

```
import sys
def MaxSum(a,l=None,r=None):
    global lind,rind
    if not a:
        return 0
    if l is None and r is None:
        l=0
        r=len(a)-1
    if(l==r):
        return a[l]
    mid=(l+r)//2
    lMax = -sys.maxsize
    total = 0
    for i in range(mid,l-1,-1):
        total += a[i]
        if total > lMax:
            lMax = total
            l1=i
            r1=m
    rMax = -sys.maxsize
    total = 0
    for i in range(mid+1, r+1):
        total += a[i]
        if total > rMax:
            rMax=total
            l2=m+1
            r2=i
   #lsum=maxsum(a,l,m)
   #rsum=maxsum(a,m+1,r)
   #if(lsum>rsum):
   #    res=lsum
   #    lind=l
   #    rind=m
   #else:
   #    res=rsum
   #    lind=m+1
   #    rind=r
   #if(res>lMax+rMax):
   #    return res
   #else:
   #    lind=l1
   #    rind=r2
   #    return(lmax+rmax)
    LRMax=max(MaxSum(a,l,mid),MaxSum(a,mid+1,r))
    return max(LRMax,lMax+rMax)
a=[int(i) for i in input().split(',')]
p=MaxSum(a)
print("The maximum subarray sum is "+str(p))
```

Marc's cake walk

```
def marcsCakewalk(calorie):
    s=0
    calorie.sort(reverse=True)
    for i in range(len(calorie)):
        s=s+(2**i)*calorie[i]
    return s
print("Enter the numnber of cupcakes")
n=int(input())
print("Enter the calories")
calorie=[int(i) for i in input().split()]
p=marcsCakewalk(calorie)
print("The minimum number of miles marc should walk to maintain his weight is "+str(p))
```

Palindromic Partitioning

```
def ispalin(x):
    if(x==x[::-1]):
        return 1
    return 0
def minparti(s,i,j):
    if(i>=j or ispalin(s[i:j+1])):
        return 0
    a=float('inf')
    for k in range(i,j):
        c=(1+minparti(s,i,k)+minparti(s,k+1,j))
        a=min(c,a)
    return a
l=input()
n=len(l)
print("The maximum number of cuts are "+str(minparti(l,0,n-1)))
```

Length of Longest Arithmetic Progression

```
l=[int(i) for i in input().split()]
d=[]
n=len(l)
for i in range(n-1):
    for j in range(i+1,n):
        d.append(l[j]-l[i])
s=set(d)
c=[]
for i in s:
    c.append(d.count(i))
print(max(c)+1)
```

Bank subarrays of equal positive and negative

```
def check(a):
    p = 0
    n = 0
    for i in range(len(a)):
        if a[i]>0:
            p+=1
        elif a[i]<0:
            n+=1
    return p,n
n = int(input("test cases: "))
for p in range(n):
    a = [int(i) for i in input("enter elements: ").split()]
    count = 0
    e = []
    for i in range(len(a)):
        for j in range(i,len(a)+1):
            b = a[i:j]
            c,d = check(b)
            if c==d and c!=0 and d!=0:
                count+=1
                e.append(b)
    print("The sub arrays are: ",e)
    print("Number of subarrays: ",count)
```

Travelling Swamp Thing Problem

```
# include <bits/stdc++.h>
using namespace std;
int n, m, e;
int dp[20][101][1<<15];
struct edge
{
   int v, d, e;
};
vector < edge > V[20];
int recursion(int last, int energy, int visited)
{
   if(energy < 0)
      return 1e9;
   else if((visited == ((1<<n)-1)) && energy >= 0)
      return 0;
   int answer = 1e9;
   if(dp[last][energy][visited] != -1)
      return dp[last][energy][visited];
   for(int i=0; i<V[last].size(); i++)
   {
      if((visited & (1 << V[last][i].v)))
         continue;
      answer = min(answer, V[last][i].d +
recursion(V[last][i].v, energy - V[last][i].e, (visited |
(1<<V[last][i].v)))) ;
   }
   dp[last][energy][visited] = answer;
   return answer;
}
int main(void)
{
         cin>>n>>m>>e;
   //scanf("%d %d %d",&n, &m, &e);
   for(int i=0; i<20; i++)
      for(int j=0; j<101; j++)
         for(int k=0; k<(1<<15); k++)
            dp[i][j][k] = -1;
   struct edge temp, temp2;
   while(m--)
   {
      int a, b, d, e;
      cin>>a>>b>>d>>e;
      //scanf("%d %d %d %d", &a, &b, &d, &e);
      a--, b--;
      temp.v = b, temp.d = d, temp.e = e;
      temp2.v = a, temp2.d = d, temp2.e = e;
      V[a].push_back(temp);
      V[b].push_back(temp2);
   }
   int answer = 1e9;
   //for(int i=0;i<n;++i)
   answer = min(answer, recursion(0, e, 1));
   if(answer == 1e9)
      cout<<"-1\n";
   else
      cout<<answer<<endl;
   return 0;
}
/*3 3 25
1 2 4 20
2 3 1 20
1 3 10 5
Output:11*/
```