Universidad del Caribe
2000 CANCUN, QUINTANA ROO, MÉXICO
CONOCIMIENTO Y CULTURA PARA EL DESARROLLO HUMANO

Técnicas Algoritmicas - IT0107

Tarea 01 del Tercer Parcial
Profesor: **Dr. David Israel Flores Granados**.
Fecha de publicación: 12 de Noviembre de 2018.
Fecha de entrega: 12 de Noviembre de 2018.

Problema de Cambio del Cajero usando una Estrategia Voraz

0.1 Problema

Los algoritmos voraces (greedy algorithms en inglés) son unas rutinas muy eficientes (O(n), O(n2)) aunque no suelen proporcionar la mejor solución a un problema. Existen algoritmos voraces muy conocidos, como el Algoritmo de Dijkstra o el Algoritmo de Kruskal.
Una de las aplicaciones de los algoritmos voraces es el conocido problema del cambio de monedas, en el cual dicho problem se presenta de la siguiente forma:

Dado un sistema monetario C de longitud K y una cantidad de cambio M, devolver una solución (si existe) que nos indique el menor número de monedas de C equivalente al monto M, es decir, que nos muestre el menor cambio posible para M a partir de las monedas de C.

Listing 1: Programa en Python para calcular los la menor cantidad de Monedas de una maquina.

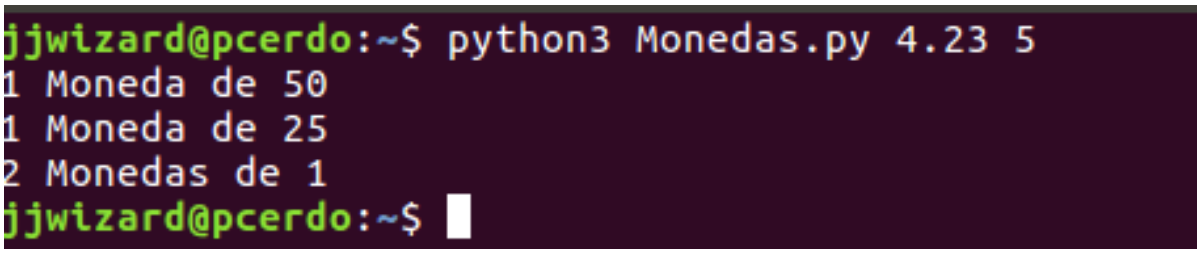
```
/*
Descripcion: Implementar con estrategia Voraz la solucion al problema de
    la maquina de cambio de monedas en Python para leer Monto y Pago y
    posteriomente entregar un vector de cambio con la minima cantidad de
    monedas.
Programador: Joan de Jesus Mendez Pool
Fecha de creacion: 12/11/2018
Entradas: Monto y Pago de la transaccion
Salida: Vector de la cantidad minima de monedas respecto al Monto y Pago
    efectuado
*/
import sys

def verifica(Monto, Pago, Monedas):
   val=round(Pago-Monto,2)
   D={0:0, 1:0, 2:0, 3:0, 4:0, 5:0}
   if(val<1):
      val=int(val*100)
   n=len(Monedas)-1

   while (val>0):
      if(val>=Monedas[n]):
         D[n]=int(val/Monedas[n])
         val=int(val - (Monedas[n]*D[n]))
      n=n-1
   return D
def imprime(D, C):
   for i in reversed(range(len(D))):
      if( D[i] != 0 ):
         if(D[i] > 1):
            print(D[i], "Monedas de", C[i], sep=" ")
         else:
            print(D[i], "Moneda de", C[i], sep=" ")
def main():
   C={0:1, 1:5, 2:10, 3:20, 4:25, 5:50}
   D=[]
   monto=sys.argv[1]
   pago=sys.argv[2]
   D=verifica(float(monto) , float(pago), C)
   imprime(D,C)
if __name__ == "__main__":
   sys.exit(int(main() or 0))
```

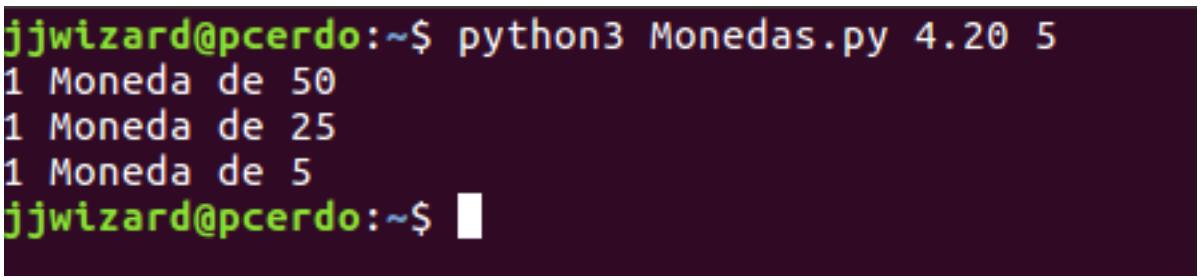
Output:

Para el primer caso tenemos un monto de 4.23 dolares y pagamos con la cantidad de 5 dolares, por lo que la cantidad de centavos a regresar es de 0.77, ejecuntando el código obtenemos el siguiente resultado:



Por lo que obtenemos que la mínima cantidad de monedas para el cambio de 0.77 es 1 moneda de 50 centavos, 1 moneda de 25 centavos, y 2 monedas de 1 centavo.

Para el segundo caso tenemos un monto de 4.20 dolares y pagamos con la cantidad de 5 dolares, por lo que la cantidad de centavos a regresar es de 0.80, ejecuntando el código obtenemos el siguiente resultado:



Por lo que obtenemos que la mínima cantidad de monedas para el cambio de 0.80 es 1 moneda de 50 centavos, 1 moneda de 25 centavos, y 1 moneda de 5 centavos.

Para el tercer caso se agregó la cantidad de 20 centavos al vector C correspondiente a la monedas, ahora tenemos un monto de 4.60 dolares y pagamos con la cantidad de 5 dolares, por lo que la cantidad de centavos a regresar es de 0.40, ejecuntando el código obtenemos el siguiente resultado:

```
jjwizard@pcerdo:~$ python3 Monedas.py 4.60 5
1 Moneda de 25
1 Moneda de 10
1 Moneda de 5
jjwizard@pcerdo:~$
```

Ahora no obtenemos la mínima cantidad de monedas porque la respuesta correcta debería dar con 2 monedas de 20 centavos pero el programa nos arroja el cambio de 0.40 es 1 moneda de 50 centavos, 1 moneda de 10 centavos, y 1 moneda de 5 centavos. Esto nos demuestra que la aplicación de los algoritmos voraces no siempre es optima para encontrar la respuesta correcta a los problemas planteados