Tarea 3 Problemas conceptuales

Iván David Valderrama Corredor
Ingeniería de Sistemas y Ciencias de la Computación
Pontificia Universidad Javeriana, Cali

12 de marzo de 2019

Índice

1. Problemas conceptuales

1.1. Problema 16-1 (a-c):Greedy Coin Change(Cormen et al., página 446).

Considere el problema de "Making Change"para n centavos, utilizando el menor número de monedas. Supongamos que el valor de cada moneda es un número entero.

A) Describe un algoritmo greedy para hacer cambios que consistan en monedas de un cuarto de dolar (25 centavos), monedas de diez centavos, monedas de cinco centavos y monedas de un centavo. Demuestre que su algoritmo produce una solución óptima.

Respuesta

Una estrategia para resolver el problema de Making Change por medio de un algoritmo greedy, seria empezar a contar las monedas de mayor valor hasta las monedas de menor valor, llegando al cambio deseado.
En este caso tenemos monedas de cuarto(25 centavos) por lo que seria x(monto a cambiar) // 25. nos daria las monedas actuales y el sobrante = x-m(monedasNoAcumuladas)*25 siguiendo:
sobrante // 10 ; sobrante = x-m*10 ; monedasAcumuladas += monedasNoAcumuladas
sobrante // 5 ; sobrante = x-m*5 ; monedasAcumuladas += monedasNoAcumuladas
sobrante // 1 ; monedasAcumuladas += monedasNoAcumuladas.
La demostracion seria la siguiente:
primero, para un x=0 la cantidad de monedas devueltas serian 0, para un x >0 se tendria una cantidad de monedas c tal que en cada caso se encuentre la menor cantidad de monedas que completen x.
si x esta entre [1-5) c equivale a 1. Lo que nos daria un optimo local.
si x esta entre [5-10) c equivale a 5. Debido a que si juntamos 5 centavos la podemos remplazar por una moneda de 5 centavos.
si x esta entre [10-25) c equivale a 10. Debido a que si juntamos 2 monedas de 5 centavos la podemos remplazar por una moneda de 10 centavos o si tenemos 5 monedas de 1 centavo y 1 moneda de 5 centavos, las podemos reemplazar por una de 10 centavos.
si x esta entre [25-n) c equivale a 25. Debido a que si tenemos 5 monedas de un centavo, 2 monedas de cinco centavos y 1 moneda de 10 centavos, las podemos reemplazar por una moneda de 25 centavos. dando como resultado una moneda mas optima.
Por lo que podemos confirmar que la estrategia de pagar con las monedas de mayor precio, nos permite entregar la minima cantidad de monedas de cambio.
[1]

B) Supongamos que las monedas disponibles están en las denominaciones que son potencias de c, es decir, las denominaciones son c0, c1, ..., ck para algunos enteros c >1 y k>= 1. Demuestre que el algoritmo greedy siempre produce una solución óptima.

Respuesta

El razonamiento en este problema es muy similar al del anterior problema, consideremos 2 tipos de monedas, los centavos y monedas que equivalen a c^, podremos emplear 1 -c centavos, debido a que cualquier cantidad mayor a c centavos, seria remplazada por almenos 1 moneda de c^. Debido a esto, esta operacion reducira el numero de monedas en 1-c es decir, cuando el resto es mayor que c, utilizara tantas monedas c^ que necesite antes de emplear los c centavos. Lo mismo ocurre para monedas mas grandes. [1]

C) Ofrezca un conjunto de denominaciones de monedas para las cuales el algoritmo greedy no produce una solución óptima. Su conjunto debe incluir un centavo para que haya una solución para cada valor de n.

Respuesta

Un conjunto de denominacion podria ser: [1,10,25] donde el valor de cambio seria 30, la solucion producida por el algoritmo de greedy daria como resultado (25,1,1,1,1,1) con un total de 6 monedas, pero un resoltado no greedy nos daria (10,10,10) con un total de 3 monedas. Lo que nos da una solucion mas optima que la del algoritmo greedy.

D) Proporcione un algoritmo en tiempo de O(nk), que realice cambios para cualquier conjunto de diferentes denominaciones de monedas, suponiendo que una de las monedas es un centavo.

Respuesta

```
INF = 99999999999

def CoinChange(D,x):
    ans,i=0, len(D)-1
    while(x>0):
```

```
        if D[i]<= x:
            monedas= x//D[i]
            x-= monedas*D[i]
            ans+=monedas
        else:
            i-=1
    return (ans)
```

-Complejidad temporal: $\theta(nk)$

1.2. Ejercicio 4.3:Trucking Company(Kleinberg and Tardos, página 189).

Usted está consultando para una compañía de camiones que realiza una gran cantidad de paquetes comerciales entre Nueva York y Boston. El volumen es lo suficientemente alto como para que tengan que enviar varios camiones cada día entre las dos ubicaciones. Los camiones tienen un límite fijo W en la cantidad máxima de peso que pueden llevar. Las cajas llegan a la estación de Nueva York una por una, y cada paquete i tiene un peso w_i. La estación de camiones es bastante pequeña, por lo que, como máximo, un camión puede estar en la estación en cualquier momento. La política de la compañía requiere que las cajas se envíen en el orden en que llegan; De lo contrario, un cliente podría molestarse. En este momento, la compañía está utilizando un algoritmo greedy simple para empacar: empacan las cajas en el orden en que llegan, y siempre que la siguiente no encaja, envían el camión en su camino. Pero se preguntan si podrían estar utilizando demasiados camiones y quieren saber su opinión sobre si la situación puede mejorar. Así es como están pensando. Tal vez uno podría disminuir la cantidad de camiones que se necesitan al enviar a veces un camión que está menos lleno, y de esta manera permitir que los siguientes camiones estén mejor embalados. Demostrar que, para un conjunto dado de cajas con pesos específicos, el algoritmo greedy actualmente en uso, en realidad minimiza la cantidad de camiones que se necesitan.

Respuesta

Teorema:
Sea A un conjunto de paquetes ordenados en orden de llegada, y 'a' $\in$ A un paquete cuyo tiempo de llegada es minimo entre todos los demas paquetes en A. Entonces 'a' hace parte de una solucion optima.

Demostracion:
Sea B $\subseteq$ A una solucion optima al problema de Trucking Company. Notamos que B $\neq \emptyset$ porque A $\neq \emptyset$. Sea 'b' $\in$ B un paquete cuyo tiempo de llegada es minimo entre todos los demas paquetes en B, procedemos por casos:
Caso (1) a = b: 'a' hace parte de una solucion optima.

Caso (2) a $\neq$ b: Considere el conjunto $B' = (B \setminus \{b\}) \cup \{a\}$ Notamos que |B| = |B'|. Tambien notamos que ningun paquete en $B \setminus \{b\}$ tiene conflicto con 'a' dado que el tiempo de llegada no es menor que el de 'a'. Entonces B' es una solucion optima.

Referencias

[1] CLRS Solutions, *16-1 Coin changing*.
https://walkccc.github.io/CLRS/Chap16/Problems/16-1/