.UBAfiuba
FACULTAD DE INGENIERÍA

Greedy: Mochila fraccionaria y cambio en monedas

Teoría de Algoritmos I (75.29 / 95.06)

Ing. Víctor Daniel Podberezski
vpodberezski@fi.uba.ar

Algoritmos greedy

Paradigma de resolución de problemas

Utiliza una heurística de selección

En cada paso elige la mejor solución local (optimo local)

El criterio de selección depende únicamente de elecciones anteriores y/o de estado actual.

Con el objetivo de encontrar el optimo global

En general se puede crear para cualquier problema un algoritmo greedy

No todos los problemas admite soluciona optimas utilizándolos

Subestructura óptima

Un problema tiene subestructura optima

Si una solución optima al problema contiene dentro soluciones optimas a sus subproblemas

La solución greedy

Toma decisiones heurísticas para ir solucionando subproblemas de forma óptima para llegar a la solución óptima general

Mochila fraccionaria

Contamos con

un contenedor (mochila) con capacidad W

conjunto de n elementos fraccionables de valor v_i y peso w_i

Queremos

Seleccionar un subconjunto de elementos o fracciones de ellos de modo de maximizar el valor almacenado y sin superar la capacidad de la mochila

Solución Greedy

Priorizar los elementos mas valiosos por unidad

Llenar el contenedor con la mayor cantidad posible de unidades del elemento disponible mas valioso por unidad

Repetir el proceso mientras quede espacio en el contenedor y elementos disponibles

Complejidad

Ordenar los elementos según valor por unidad

O(nlogn)

Iterar mientras haya lugar

O(n)

La complejidad algorítmica de la solución es O(nlogn)

```
Lugar = W
Valor = 0
Mientras existan elementos disponibles y
                                    lugar > 0

    Sea x el elemento disponible con mayor
                                valor por unidad

    Cantidad = min(Wx , Lugar)

    Valor += Cantidad * Vx
    Lugar -= Cantidad
    Quitar x de disponible

Retornar Valor
```

Optimalidad

El algoritmos greedy es siempre optimo?

Llamaremos a la solución greedy G, y a los elementos elegidos $g_1, g_2, \ldots g_r$

Imaginemos que existe una solución optima O, con elementos $o_1, o_2, \ldots, o_s$

Si valor(O) > valor(G) entonces existe al menos un elemento o_i

que no esta en G con mayor valor por unidad que algún elemento gi en G

O esta con menor cantidad en G y esa diferencia esta en al menos un elemento en O en menor valor

Esto implica que Greedy no seleccionó un elemento disponible según su programación

Es absurdo! Por lo tanto greedy es optimo

Cambio mínimo

Contamos con

Un conjunto de monedas de diferente denominación sin restricción de cantidad

$\$ = (c_1, c_2, c_3, \ldots, C_n)$

Un importe X de cambio a dar

Queremos

Entregar la menor cantidad posible de monedas como cambio

Ejemplos

Denominaciones de las monedas

(1,5,10,25,50,100)

(!) Asumiremos que existe siempre la moneda unidad

Cambio mínimo a retornar

80 → (0,1,0,1,1,0)

24 → (4,0,2,0,0,0)

101 → (1,0,0,0,0,1)

Solución Greedy

Conocida como “solución del cajero”

Tomar la mayor denominación posible menor al cambio a dar

Dar tantas monedas de esa denominación como sea posible

Con el resto, repetir el procedimiento hasta dar todo el cambio

```
Cambio = X
#Monedas = 0
Mientras X > 0

     Sea Ci la denominación con mayor valor
                    en $ tal que Ci<=Cambio

          Cantidad = piso(Cambio / Ci)
          Cambio -= Ci * cantidad
          #Monedas += Cantidad


Retornar #Monedas
```

Complejidad O(n)

Optimalidad

Contra ejemplo

$ = (1,2,4,5,10)

X= 8

Greedy = (1,1,0,1,0) → 3

Optimo = (0,0,2,0,0) → 2

Greedy no es optimo!

Casos especiales

Existen casos donde el algoritmo greedy funciona

La base (1,5,10,25,50,100) es un caso

También funciona $=(1,5,25), pero no $=(1,10,25)

Se conoce a un sistema $ como “canonico”, “standard”, “ordenado” o “greedy”

a aquel en el que para todo x, greedy($,x) = optimo($,x)

(Existen diferentes papers que usan equivalentemente estos nombres)

Basta con un contraejemplo para determinar que un sistema no lo es

Cuando funciona greedy?

No es necesario probar todos los valores posibles de x

En “Optimal Bounds for the ChangeMaking Problem”

https://www.cs.cornell.edu/~kozen/Papers/change.pdf

Si existe un contraejemplo el mismo estará entre $c_3 + 1 < x < c_n + c_{n-1}$

En “A Polynomial-time Algorithm for the Change-Making Problem”

https://ecommons.cornell.edu/handle/1813/6219

Se presenta un algoritmo $O(n^3)$ para determinar si un $ es canónico

Se puede construir un algoritmo siempre optimo para resolver este problema utilizando “programación dinámica”

.UBAfiuba

FACULTAD DE INGENIERÍA

Presentación realizada en Abril de 2020