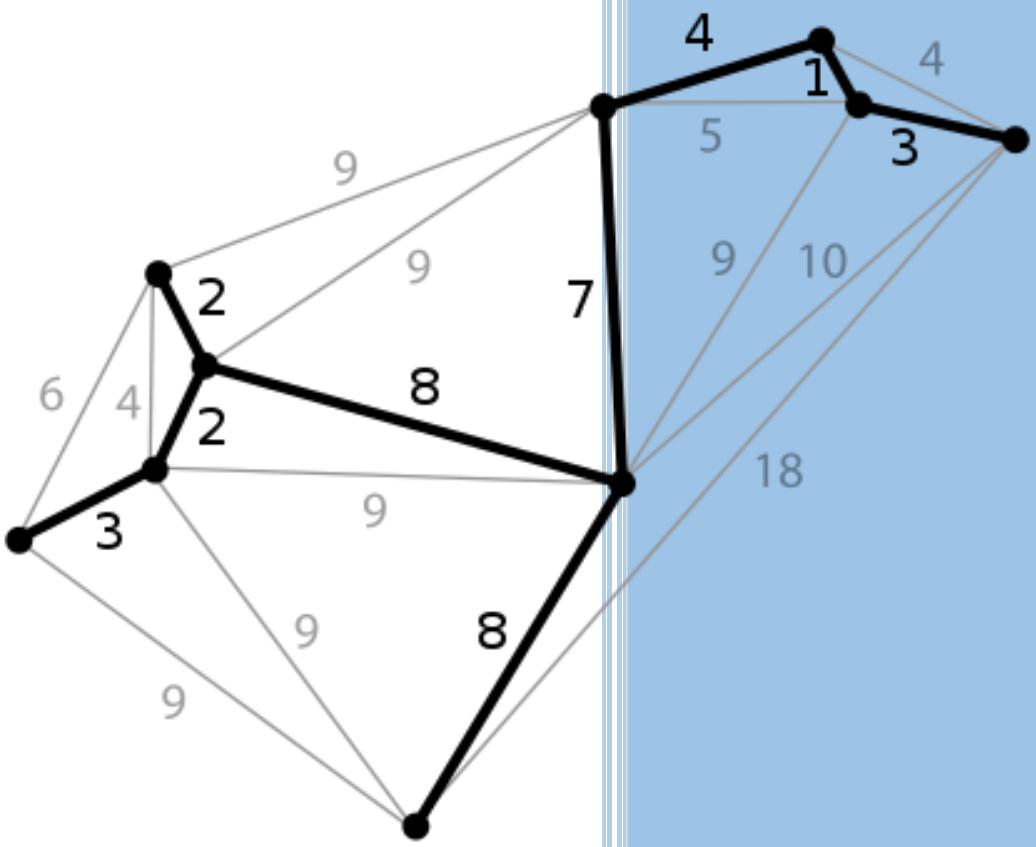
INSTITUTO POLITÉCNICO NACIONAL
ESCUELA SUPERIOR DE CÓMPUTO
ING. EN SISTEMAS COMPUTACIONALES

PRÁCTICA 3:

ALGORITMOS GREEDY



Alumnos:
Bastida González María Guadalupe
Mejía Romo Pablo Emanuel

Boletas:
2021630164
2020350970

ANÁLISIS Y DISEÑO DE ALGORITMOS
3CM3
PROFESORA: *SÁNCHEZ GARCÍA LUZ MARÍA*

12/05/2021

PRÁCTICA 3:

ALGORITMOS GREEDY

Objetivo: Programar algoritmos voraces para la solución de problemas computacionales.

Actividades:

1. Programar en ANSI C, los 2 algoritmos voraces siguientes:
   a) Mochila fraccionaria.
   b) Cambio de monedas.
2. Describir brevemente en qué consiste cada algoritmo por la estrategia voraz.
3. Explicar si existen otras formas de resolver cada algoritmo (fuerza bruta, programación dinámica, etc.)
4. Indicar la ecuación de recurrencia y la notación de orden (Big O) de cada algoritmo.

Algoritmo Mochila fraccionaria

Planteamiento del problema: se plantea que tenemos una mochila que puede cargar un peso máximo P y que tenemos n objetos fraccionables con un peso p_i y un valor v_i cada uno. Se trata de cargar la mochila sin pasarnos del peso máximo y maximizando el valor total seleccionando un objeto de cada clase en fracciones $0 \leq x_i \leq 1$ donde $x_i \in \mathbb{R}$, es decir, podemos no seleccionar un objeto, seleccionarlo o tomar una fracción. Formalmente se escribe:

Con $p_i>0$, $v_i>0$, $0 \leq x_i \leq 1$ hemos de maximizar $V = \sum_{(i=1..n)} x_i v_i$ tal que $\sum_{(i=1..n)} x_i p_i \leq P$

Solución por estrategia voraz:

Exactamente para este algoritmo existen 3 formas de resolverlo vorazmente, por el peso mayor, el valor máximo y el ultimo por "densidad de utilidad"; es decir, ordenándola o ir metiendo los objetos de modo que $^{v1}/_{p1} \geq {}^{v2}/_{p2} \geq \cdots \geq {}^{vn}/_{pn}$.

De esta manera se meterá el objeto con mayor utilidad hasta llegar al máximo de carga en la mochila, si aún no se completa, entonces se ejecutará una división del siguiente objeto con mayor utilidad y se agregará para lograr una carga completa.

Otras formas de solución:

Este problema en particular también se puede resolver por *Back tracking* ya que este sigue buscando maximizar el beneficio de algo (en este caso de los objetos dentro de la mochila) sin importar su coste. También se puede resolver por *programación*

dinámica, ya que esta la resuelve de manera más fácil al ahorrarse la recursividad en los cálculos de la misma.

Así pues, se Indica la ecuación de recurrencia y la notación de orden (Big O).

$$\mathbf{T(n)= \ 1 + 4n + n\ log(n) \ \therefore \ O(nlogn)}$$

Cambio de monedas:

Planteamiento del problema:

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

Solución por estrategia voraz:

Se elige la moneda de mayor denominación tal que no sea mayor que la cantidad restante para alcanzar el valor objetivo. Sin embargo, este algoritmo no es adecuado para los sistemas de monedas arbitrarios: si las denominaciones de las monedas fueran 1, 3 y 4, entonces para obtener 6, el algoritmo codicioso elegiría tres monedas (4,1,1) mientras que la solución óptima es dos monedas (3,3).

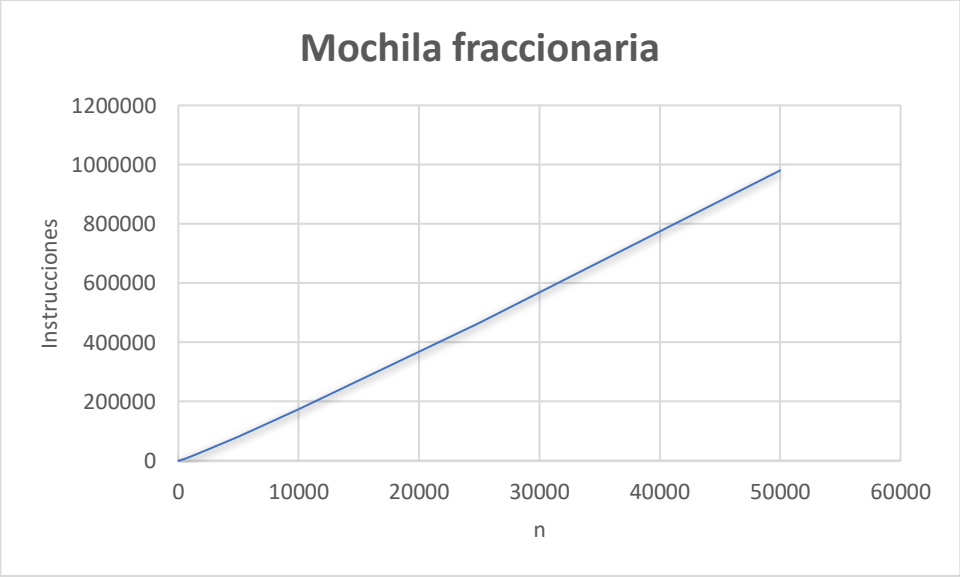
Otras formas de solución:

1. Ordeno los pares de monedas ingresados, de mayor a menor, por valor.
2. Mientras el saldo no ha sido completado, doy una moneda del mayor valor que resulta menor o igual al saldo, cada vez descontando el número de monedas disponibles de esa denominación.

Así pues, se Indica la ecuación de recurrencia y la notación de orden (Big O).

$$T(n)= 2n + 1 + n \log(n) \therefore O(n)$$

5. Graficar el comportamiento temporal de cada algoritmo (1 gráfica para cada uno de los algoritmos).



6. Ejemplifique cada algoritmo con valores de entrada y los resultados óptimos que cada algoritmo arroja para maximizar o minimizar la función objetivo.

Algoritmo de la mochila fraccionaria

índice	Peso	Valor	*Precio*	*Fracción*
0	4	10	2.5	0
1	3	40	13.333	1
2	2	20	10	0.6
3	5	30	6	1

Algoritmo del problema del cabio de monedas

- Valor: 7.5
- Posibles monedas: 2, 1, 0.5, 0.2, 0.1, 0.05, 0.02, 0.01
- Iteraciones: 45
- Cambio de monedas (resultados): [3, 1, 0, 1, 1, 0, 2, 0]
- Valor de cambio: 3×2 + 1×1 + 1×0.2 + 1×0.1 + 2×0.02 = 7.34

7. Indique si los datos del punto 6 fueron utilizados para realizar las pruebas al implementar los algoritmos, o en caso contrario, indique los valores utilizados en el problema de la mochila y el problema del cambio.

Los datos utilizados en el punto anterior fueron para probar el eficiencia del algoritmo por medio de la estrategia voraz, cabe destacar que, en algunos casos, el problema del cambio de monedas con este tipo de estrategias no es el óptimo pues a veces se va por el camino más rápido y no analiza detenidamente los datos que tiene a su disposición.

8. Realice una conclusión del comportamiento de cada algoritmo acorde a la estrategia voraz.

El problema de la mochila fraccionaria por estrategia voraz obtiene la maximización de su función al encontrar el mayor y mejor beneficio de entre los objetos y las diferentes opciones de selección, así mismo se comprobó su eficacia al programar su algoritmo y obtener su tiempo de ejecución.

Para el problema del cambio de moneda con la estrategia voraz es bueno pero no es el mejor, pues, como ya mencionamos, a veces se va por el camino más corto y con tal de solucionar rápido no toma en cuenta resultados óptimos.

9. Indique el entorno controlado para realizar las pruebas experimentales.

Se utilizó el editor de códigos Visual Studio Code junto con la terminal de comandos que esta incorpora (CMD) para la programación de los códigos en c, el equipo de cómputo donde se ejecutaron dichos códigos cuenta con las siguientes especificaciones:

- Procesador: Intel Core i7-4600 de 4 núcleos a 2.69 GHz
- Memoria RAM: DDR3 de 8 GB
- Unidad de almacenamiento: Unidad de estado sólido de 500 GB
- Sistema operativo: x64 o 64 bits
- Compilador: MinGw32-gcc v6.3.0-1
- Editor de códigos: Visual Studio Code v1.55.2

Además, se procuró que el equipo no ejecutara otros programas más que un editor de código, una consola de comandos y el mismo algoritmo.

10. Responda las siguientes cinco preguntas:

a) Para cada algoritmo ¿Existe alguna solución que no sea voraz y que sea óptima? ¿Cuál es? Si, para el algoritmo de la mochila fraccionaria existe la resolución por back tracking y por programación dinámica.

En el caso del algoritmo el cambio de monedas si existe otra resolución, por programación dinámica y dinámica por arboles de convulsión probabilística.

b) ¿Cuál de los 2 algoritmos es más fácil de implementar?

El problema del cambio de monedas

c) ¿Cuál de los 2 algoritmos es el más difícil de implementar?

El problema de la mochila fraccionaria

d) ¿El comportamiento experimental de los algoritmos era el esperado? ¿Por qué?
Si, ya que se hicieron pruebas a mano sobre los posibles resultados que se podían obtener al ejecutarlos, y en el caso de los 2 algoritmos, los resultados coincidieron.

e) ¿Qué otros algoritmos voraces recomendarían para realizar esta práctica?

Sería interesante ver el comportamiento de los arboles de expansión con los algoritmos de Prim, Kruskal y Código de Hoffman. Ya que, a parte de ser algoritmos un poco mas desafiantes, ayudaría a entender mejor los árboles binarios.