

A cartoon illustration of a man with a large nose and dollar signs for eyes, sitting on a pile of money. The man has a wide, greedy grin and is wearing a suit. He is surrounded by stacks of gold coins and banknotes. The word "Greedy" is written in large, bold, dark blue letters across the center of the image.

Greedy

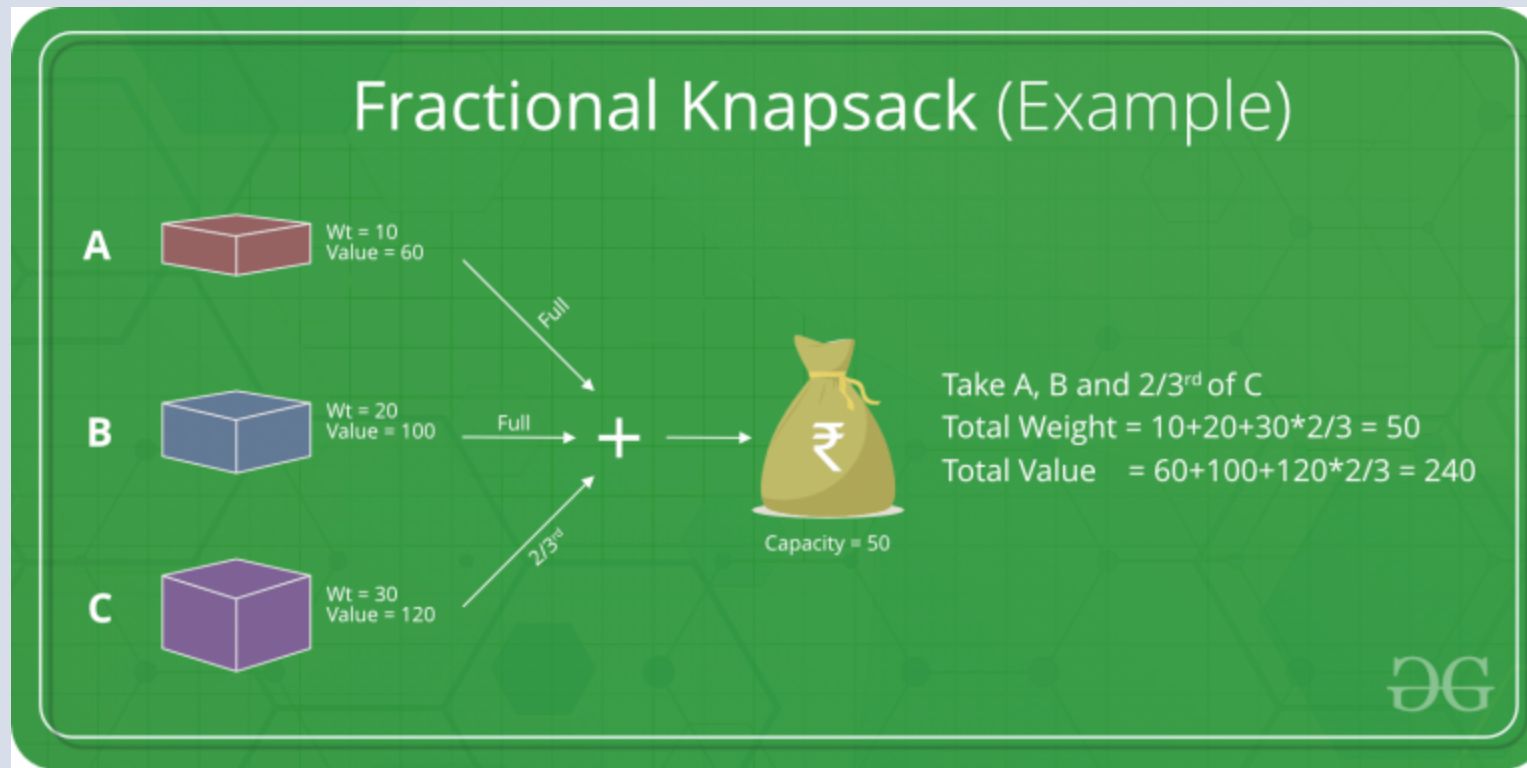
Por Ariel Parra.

¿Qué es un Algoritmo Greedy?

Greedy es un paradigma algoritmo que construye una pieza de solución por pieza, siempre eligiendo la siguiente pieza que ofrece el beneficio más obvio e inmediato. Así que los problemas donde elegir localmente óptimo también conduce a la solución global son el mejor ajuste para Greedy.

Ejemplo

Por ejemplo, considere el problema fraccional de Knapsack. La estrategia óptima local es elegir el elemento que tiene el valor máximo vs ratio de peso. Esta estrategia también conduce a una solución globalmente óptima porque se nos permite tomar fracciones de un artículo.



Estructura basica

- Declarar un resultado vacio = 0.
- Hacemos una elección Greedy para seleccionar, Si
- la elección es factible añadirlo al resultado final.
- retornar el resultado.

Elementos del algoritmo

Para poder resolver un problema usando el enfoque greedy, tendremos que considerar, 6 elementos:

1. **Conjunto de candidatos** (elementos seleccionables).
2. **Solución parcial** (candidatos seleccionados).
3. **Función de selección** (determina el mejor candidato del conjunto de candidatos seleccionables).
4. **Función de factibilidad** (determina si es posible completar la solución parcial para alcanzar una solución del problema).
5. **Criterio que define lo que es una solución**(indica si la solución parcial obtenida resuelve el problema).
6. **Función objetivo** (valor de la solución alcanzada)

Problema:

Supongamos que tienes un conjunto de monedas con diferentes valores (por ejemplo, 1, 5, 10, 25 centavos) y deseas dar cambio a un cliente usando la menor cantidad de monedas posible.

1. **Conjunto de candidatos:** En este caso, el conjunto de candidatos son las monedas disponibles para dar cambio, es decir, las monedas de 1, 5, 10 y 25 centavos.
2. **Solución parcial:** Al comenzar, no hemos seleccionado ninguna moneda, por lo que la solución parcial es vacía.
3. **Función de selección:** La función de selección determina el mejor candidato del conjunto de monedas seleccionables. En este caso, el candidato seleccionado sería la moneda de mayor valor que no exceda el cambio que debemos dar. Por ejemplo, si debemos dar 16 centavos de cambio, seleccionaríamos la moneda de 10 centavos.

4. **Función de factibilidad:** La función de factibilidad determina si es posible completar la solución parcial para alcanzar una solución del problema. En este caso, si la moneda seleccionada es menor o igual al cambio restante, la solución es factible. Si no, tendríamos que seleccionar una moneda de menor valor.
5. **Criterio que define lo que es una solución:** En este caso, una solución parcial se considera una solución completa cuando hemos dado cambio por completo al cliente.
6. **Función objetivo:** La función objetivo en este caso sería el valor total de las monedas seleccionadas, es decir, el valor del cambio entregado.

Supongamos que debemos dar 47 centavos de cambio a un cliente:

1. **Conjunto de candidatos:** {25 centavos, 10 centavos, 5 centavos, 1 centavo}
2. **Solución parcial:** {}
3. **Función de selección"** Seleccionamos la moneda de 25 centavos (la de mayor valor que no excede el cambio).
4. **Función de factibilidad:** La solución parcial es factible, ya que 25 centavos es menor o igual a 47 centavos.
5. **Criterio que define lo que es una solución:** La solución parcial no es completa todavía, ya que falta dar 22 centavos de cambio.
6. **Función objetivo:** El valor de la solución actual es 25 centavos.

Luego, repetiríamos los pasos 3-6 hasta completar el cambio. En este caso, seleccionaríamos una moneda de 10 centavos, luego una de 10 centavos nuevamente, y finalmente una de 1 centavo, lo que nos daría un total de 47 centavos de cambio con la menor cantidad de monedas posible.

Codigo del ejemplo

```
vector<int> darCambio(int cambio) {  
    vector<int> monedas = {25, 10, 5, 1};  
    vector<int> solucion;  
    for (int i = 0; i < monedas.size(); i++) {  
        while (cambio >= monedas[i]) {  
            solucion.push_back(monedas[i]);  
            cambio -= monedas[i];  
        }  
    }  
    return solucion;}  

```

```
int main() {  
    int cantidadCambio; cin >> cantidadCambio;  
    vector<int> solucion = darCambio(cantidadCambio);  
    cout << cantidadCambio << ".";  
    for (int i = 0; i < solucion.size(); i++) {  
        cout << solucion[i] << endl;  
    }  
    return 0;}  

```

Referencias

<https://www.geeksforgeeks.org/greedy-algorithms/>

<https://www.geeksforgeeks.org/introduction-to-greedy-algorithm-data-structures-and-algorithm-tutorials/>

<https://www.geeksforgeeks.org/c-program-for-greedy-algorithm-to-find-minimum-number-of-coins/>

<https://www.youtube.com/watch?v=ALtJncFD8N8>

https://elvex.ugr.es/decsai/algorithms/slides/4_greedy.pdf

<https://github.com/CPCFI-org/lectures/tree/main/3-CS-DQ-Greedy>