Algoritmos Voraces o Greedy

Análisis y Diseño de Algoritmos

Dr. Jaime Osorio Ubaldo

Algoritmo Greedy

Características

1. Generalmente son utilizados para resolver problemas de optimización (maximización o minimización de la función objetivo).
2. Estos algoritmos toman decisiones basándose en la información que se tiene disponible de forma inmediata, sin tomar en cuenta los efectos que pudieran causar en el futuro.
3. Son fáciles de implementar.

Algoritmo Greedy

Procedimiento.

1. Se define el conjunto candidatos, por ejemplo: el conjunto de monedas disponibles, las aristas de un grafo que se pueden utilizar para construir una ruta, el conjunto de tareas que hay que planificar, etc.
2. En cada paso se considera al candidato indicado por una función de selección voraz, que indica cuál es el candidato más prometedor de entre los que aún no se han considerado, es decir, los que aún no han sido seleccionados ni rechazados.
3. Conforme avanza el algoritmo, los elementos del conjunto inicial de candidatos pasan a formar parte de uno de dos conjuntos: el de los candidatos que ya se consideraron y que forman parte de la solución o el conjunto de los candidatos que ya se consideraron y se rechazaron.
4. Mediante una función de factibilidad se determina si es posible o no completar un conjunto de candidatos que constituya una solución al problema. Es decir, se analiza si existe un candidato compatible con la solución parcial construida hasta el momento.

El problema del cambio

1. Dado un monto n y un conjunto de denominaciones de billetes, se desea elegir la mínima cantidad de billetes cuya suma sea n.
2. Para construir la solución con un algoritmo voraz tomaremos cada vez el billete de mayor denominación, siempre y cuando la suma acumulada no sea mayor a n. El algoritmo falla cuando se presenta el caso en el que la suma de los billetes no puede igualar a n, en este caso el algoritmo ávido no encuentra una solución.
3. Nótese que en este caso en particular no existe restricción en cuanto al número de billetes que se pueden tomar, es decir, siempre es posible elegir k billetes de una determinada denominación. Por lo tanto no es necesario tomar en cuenta si un billete ya se consideró o no.

El problema del cambio

Veamos como sería la implementación

```
#define NUM 5
int denominaciones[NUM] = {100, 20, 10, 5, 1};
int numBilletes[NUM];
for(int i = 0; i < NUM; i++)
    numBilletes[i] = 0;
int SeleccionVoraz(int resto){
    int i = 0;
    while(i < NUM ∧ denominaciones[i] > resto)
        i++;
return i;
};
```

```
void cambio(int monto, int *numBilletes){
   int suma = 0;
   int iDen = 0;
   while(suma < monto){
      iDen = SeleccionVoraz(monto - suma);
      if(iDen >= NUM){
           cout << "No se encontró solución.Cambioincompleto";
           return;
      };
   numBilletes[iDen] = numBilletes[iDen] + 1;
   suma = suma + denominaciones[iDen];
   };
   return;
}
```

Los algoritmos voraces construyen la solución en etapas sucesivas, tratando siempre de tomar la decisión óptima para cada etapa. Sin embargo, la búsqueda de óptimos locales no tiene por qué conducir siempre a un óptimo global. A continuación presentamos un caso en el que falla en algoritmo del cambio.

El problema del cambio

Supongamos que el sistema monetario está compuesto por los siguientes billetes de 11,5,1 consideremos que se desea dar de cambio cuyo monto es 15 el algoritmo voraz tomará primero el billete mayor, por lo que la solución que obtendrá es

$$15 = 11 + 1 + 1 + 1 + 1$$

sin embargo, la solución óptima es

$$15 = 5 + 5 + 5$$

"La estrategia de los algoritmos ávidos consiste en tratar de ganar todas las batallas sin pensar que, como bien saben los estrategas militares y los jugadores de ajedrez, para ganar la guerra muchas veces es necesario perder alguna batalla." [Guerqueta y Vallecillo, 2000].

Temas de investigación.

1. ¿Qué ventajas y desventajas presenta esta técnica de diseño?.
2. Explique una aplicación de esta técnica de diseño.