Universidad de Granada

Algorítmica

Algoritmos Greedy, parte 1

Laura Calle Caraballo
Cristina María Garrido López
Germán González Almagro
Javier León Palomares
Antonio Manuel Milán Jiménez

18 de abril de 2016

Índice

1. Introducción.

1.1. Descripción del problema.

Un granjero necesita disponer siempre de un determinado fertilizante. La cantidad máxima que puede almacenar la consume en r días y, antes de que eso ocurra, necesita acudir a una tienda del pueblo para abastecerse. El problema es que dicha tienda tiene un horario de apertura muy irregular (sólo abre determinados días). El granjero conoce los días en que abre la tienda, y desea minimizar el número de desplazamientos al pueblo para abastecerse.

1.2. Resolución.

Se ha diseñado un algoritmo *Greedy* para determinar los días en los que debe ir al pueblo a abastecerse durante un cierto intervalo de tiempo. Asimismo, se demuestra que las soluciones proporcionadas por este algoritmo son óptimas.

2. Algoritmo desarrollado.

2.1. Elementos.

- Conjunto de Candidatos: lista de días que abre la tienda.
- Conjunto de Seleccionados: lista de días que ha ido a la tienda.
- Función Solución: ha cubierto el período de tiempo para el que tenía información.
- Función de Factibilidad: si entre dos días de apertura pasan más de r días, el problema no tiene solución.
- Función Selección: puede o no esperar al próximo día que abra la tienda.
- Función Objetivo: cuántos días ha abierto la tienda.

2.2. Pseudocódigo.

```
function Greedy(listaApertura, limiteDias);
begin
    ultimoDiaCompra ⟵ 0;
    listaDias ⟵ ∅;
    for i = 1 to n − 1 do
        if listaApertura[i + 1] - ultimoDiaCompra > limiteDias then
            listaDias ⟵ listaDias ∪ listaApertura[i];
            ultimoDiaCompra ⟵ listaApertura[i];
        end
    end
    return listaDias, |listaDias|;
end
```

3. Demostración de optimalidad.

Suponemos una solución óptima B, lo que quiere decir que resolverá el problema en menos visitas al proveedor que nuestra solución A. Llamaremos k al máximo valor posible donde ambas soluciones son iguales, es decir, $b(k) = a(k)$.

Asumiendo que:

1. $0 = a(0) < a(1) < ... < a(p) =$ *SN-Óptima*, siendo p el último día seleccionado por el algoritmo. Asumimos que esta solución no es óptima.
2. Sean las soluciones óptimas $0 = b(0) < b(1) < ... < b(q)$, $q < p$. Llamaremos k al máximo valor posible donde $b(0) = a(0)$, $b(1) = a(1)$, ..., $b(k) = a(k)$ y sea *S-Óptima* una de estas soluciones óptimas.

Entonces, sabemos que:

1. $a(k+1) > b(k+1)$, ya que el algoritmo *Greedy* siempre escogerá el día más alejado.
2. $0 = a(0) < ... < a(k) < a(k+1) < b(k+2) < ... < b(q)$, donde q es el último día, sería otra solución al problema; igualmente, sería óptima, puesto que tiene el mismo tamaño que la solución óptima *S-Óptima* y en la que, además, la solución A y la nueva solución coinciden hasta $k+1$ y no hasta k.
3. Llegamos a una contradicción, puesto que k no es el máximo valor en el que $a(k)$ y $b(k)$ son iguales.

4. Conclusión.

Hemos podido comprobar que para este tipo de problemas un algoritmo *Greedy* resulta la mejor opción ya que resuelve el problema de manera óptima. Además presenta un orden de eficiencia $O(n)$, por lo que podemos decir que es bastante eficiente.