UNIVERSIDAD DE GRANADA

Algoritmos Greedy (o voraces)

Algorítmica. Práctica 3

Jose Alberto Hoces Castro
Javier Gómez López
Manuel Moya Martín Castaño

Mayo 2022

Contenidos

Objetivo de la práctica

Aprender a analizar un problema y resolverlo mediante la técnica Greedy, además de justificar su utilidad para resolver problemas de forma muy eficiente, obteniendo la solución óptima o muy cercana a la óptima.

Ejercicio 1. Contenedores

Enunciado

Se tiene un buque mercante cuya capacidad de carga es de K *toneladas y un conjunto de contenedores* $c_1, \ldots, c_n$ *cuyos pesos respectivos son* $p_1, \ldots, p_n$ *(expresados también en toneladas). Teniendo en cuenta que la capacidad del buque es menor que la suma total de los pesos de los contenedores:*

Primer ejercicio

Diseñe un algoritmo que maximice el número de contenedores cargados, y demuestre su optimalidad.

Primer ejercicio. Planteamiento del algoritmo

- Como queremos cargar el máximo número de contenedores, empezaremos cargando los más **pequeños**.
- Ordenamos de **menor a mayor** peso los contenedores.
- Empezamos a cargar los de menor peso hasta que superemos las K toneladas del buque mercante.
- Todo esto lo simulamos con un vector de enteros en nuestro código, el cual tenemos a continuación.

Primer ejercicio. Código

```
1  int contenedoresGreedy1(int *T, int n){
2
3          int used = 0;
4          int result = 0;
5      vector<int> myvector(T,T+n);
6      sort(myvector.begin(),myvector.end());
7
8      for(int i = 0; (i < n) && (used <= n); i++){
9          used += T[i];
10         result++;
11     }
12
13     return result;
14 }
```

Primer ejercicio. Enfoque Greedy

Las 6 características de nuestro problema que hacen que lo identifiquemos como problema Greedy son:

- **Un conjunto de candidatos**: En este caso, los contenedores a cargar.
- **Una lista de candidatos ya usados**: Los contenedores que ya han sido cargados.
- **Un criterio que dice cuándo un conjunto de candidatos forma una solución**: El criterio es que la suma de los pesos de un conjunto de contenedores no sea superior a las K toneladas del buque.

Primer ejercicio. Enfoque Greedy

- **Un criterio que dice cuándo un conjunto de candidatos es factible (podrá llegar a ser una solución)**: el conjunto de contenedores que se evalúe no debe superar en peso las K toneladas del buque.
- **Una función de selección que indica en cualquier instante cuál es el candidato más prometedor de los no usados todavía**: El contenedor de menor peso de los que aún no están cargados, de ahí que los ordenemos de menor a mayor peso.
- **La función objetivo que intentamos optimizar**: El número de contenedores a cargar, es lo que queremos maximizar.

Primer ejercicio. Estudio de la optimalidad

Segundo ejercicio

Diseñe un algoritmo que intente maximizar el número de toneladas cargadas.

Segundo ejercicio. Planteamiento del algoritmo

- Como queremos cargar el máximo número de toneladas, empezaremos cargando los más **pesados**.
- Ordenamos de **mayor a menor** peso los contenedores.
- Empezamos a cargar los de mayor peso hasta que superemos las K toneladas del buque mercante.
- Todo esto lo simulamos con un vector de enteros en nuestro código, el cual tenemos a continuación.

Segundo ejercicio. Código

```
1  int contenedoresGreedy2(int *T, int n){
2
3          int used = 0;
4      vector<int> myvector(T,T+n);
5      sort(myvector.begin(),myvector.end(), greater<int>());
6
7      for(int i = 0; (i < n) && (used <= n); i++){
8          used += T[i];
9      }
10
11     return used;
12 }
```

Segundo ejercicio. Estudio de la optimalidad

[5, 4, 6, 1, 1, 2, 7, 9, 8, 3] K = 10

↓

[9, 8, 7, 6, 5, 4, 3, 2, 1, 1]

Solución aportada por nuestro algoritmo: [9]

Solución óptima: [1,2,3,4]