

Problema del buque mercante

Joaquín Sergio García Ibáñez
Juan Navarro Maldonado

Índice

- Enunciado del Problema
- Diseño de Componentes
- Diseño del Algoritmo
- Estudio de la optimalidad
- Traza de la ejecución
- Código y Ejecución
- Critica a Chat GPT



Enunciado del Problema

Enunciado



Se tiene un buque mercante cuya capacidad de carga es de k toneladas, y un conjunto de contenedores c_1, \dots, c_n cuyos pesos respectivos son p_1, \dots, p_n (expresados también en toneladas). Teniendo en cuenta que la capacidad del buque podría ser menor que la suma total de los pesos de los contenedores, se pide: Diseñar un algoritmo que permita decidir qué contenedores hay que cargar para maximizar la suma de los pesos de los contenedores a transportar en el barco.

Diseño de Componentes

Diseño de Componentes

- **Lista de candidatos:** Los contenedores del conjunto inicial C.



Diseño de Componentes

- **Lista de candidatos:** Los contenedores del conjunto inicial C.
- **Lista de candidatos utilizados:** Los contenedores que ya han sido insertados en la solución.



Diseño de Componentes

- **Lista de candidatos:** Los contenedores del conjunto inicial C.
- **Lista de candidatos utilizados:** Los contenedores que ya han sido insertados en la solución.
- **Criterio de selección:** Ordenaremos de mayor a menor según el peso de los contenedores y seleccionaremos como candidato aquel contenedor que hace que su peso sea mayor que los contenedores existentes.



Diseño de Componentes

- **Lista de candidatos:** Los contenedores del conjunto inicial C.
- **Lista de candidatos utilizados:** Los contenedores que ya han sido insertados en la solución.
- **Criterio de selección:** Ordenaremos de mayor a menor según el peso de los contenedores y seleccionaremos como candidato aquel contenedor que hace que su peso sea mayor que los contenedores existentes.
- **Criterio de factibilidad:** El candidato seleccionado cc se insertará en la solución siempre que sea un contenedor cuyo peso sea mayor que el del resto que quede aun.



Diseño de Componentes

- **Lista de candidatos:** Los contenedores del conjunto inicial C.
- **Lista de candidatos utilizados:** Los contenedores que ya han sido insertados en la solución.
- **Criterio de selección:** Ordenaremos de mayor a menor según el peso de los contenedores y seleccionaremos como candidato aquel contenedor que hace que su peso sea mayor que los contenedores existentes.
- **Criterio de factibilidad:** El candidato seleccionado cc se insertará en la solución siempre que sea un contenedor cuyo peso sea mayor que el del resto que quede aun.
- **Función solución:** El algoritmo parará cuando la suma de los pesos de los contenedores sea igual o superior sin superar a la capacidad de carga del buque k.



Diseño de Componentes

- **Lista de candidatos:** Los contenedores del conjunto inicial C.
- **Lista de candidatos utilizados:** Los contenedores que ya han sido insertados en la solución.
- **Criterio de selección:** Ordenaremos de mayor a menor según el peso de los contenedores y seleccionaremos como candidato aquel contenedor que hace que su peso sea mayor que los contenedores existentes.
- **Criterio de factibilidad:** El candidato seleccionado cc se insertará en la solución siempre que sea un contenedor cuyo peso sea mayor que el del resto que quede aun.
- **Función solución:** El algoritmo parará cuando la suma de los pesos de los contenedores sea igual o superior sin superar a la capacidad de carga del buque k.
- **Función objetivo:** Encontrar los contenedores con mayor peso.



Diseño del Algoritmo

Diseño del Algoritmo

Algoritmo S = Greedy(p: pesos de cada contenedor, n: num de contenedores, k: carga maxima, tam_sol: tamaño de la solucion)

Ordenar de mayor a menor el vector de pesos p[]

i = 0

sum = 0

Mientras i <= n

Si sum + p[i] <= k

 Seleccionamos el peso del contenedor i y añadimos a S

 sum = sum + p[i]

 i++

 tam_sol++

Fin Si

En otro caso

 i++

Fin En Otro Caso

Devolver S

Fin Algoritmo



Estudio de la Optimalidad

Optimalidad

Sea $T=\{c_1, \dots, c_n\}$ y suponemos que los pesos de los contenedores están ordenadas de mayor a menor, ya que si no están ordenados, no se obtiene el óptimo

$$p_1 \leq p_2 \leq \dots \leq p_n$$

Si todos los $p_i = 1$ entonces la solución óptima es trivial.

En el caso contrario, la solución que proporciona el algoritmo greedy se formará por el conjunto $S = \{c_1, c_2, \dots, c_n\}$ de modo que:

$$\sum_{i=1}^m c_i = \sum_{i=1}^k p_i \leq k \text{ y } \sum_{i=m+1}^n c_i \leq k$$

Por tanto, el beneficio de la solución de P es máximo global



Traza de la Ejecución

Traza

1. Para un ejemplo de 5 contenedores cuyos pesos son respectivamente 1, 5, 3, 20, 15 toneladas respectivamente y cuya carga máxima es 25.



Traza

1. Para un ejemplo de 5 contenedores cuyos pesos son respectivamente 1, 5, 3, 20, 15 toneladas respectivamente y cuya carga máxima es 25.
2. Ordenar dicho vector de orden mayor a menor



Traza

1. Para un ejemplo de 5 contenedores cuyos pesos son respectivamente 1, 5, 3, 20, 15 toneladas respectivamente y cuya carga máxima es 25.
2. Ordenar dicho vector de orden mayor a menor
3. Comprobar si se ha alcanzado ya a comprobar todos los contenedores y si no en el bucle comprobaría si la suma acumulada de los pesos más el peso del i ésimo contenedor es menor o igual a la carga máxima que puede llevar el buque



Traza

1. Para un ejemplo de 5 contenedores cuyos pesos son respectivamente 1, 5, 3, 20, 15 toneladas respectivamente y cuya carga máxima es 25.
2. Ordenar dicho vector de orden mayor a menor
3. Comprobar si se ha alcanzado ya a comprobar todos los contenedores y si no en el bucle comprobaría si la suma acumulada de los pesos más el peso del i ésimo contenedor es menor o igual a la carga máxima que puede llevar el buque
4. Si no se ha alcanzado el peso máximo, seleccionaremos el el contenedor i y sumaremos a nuestra suma acumulada de pesos el peso de ese contenedor



Traza

1. Para un ejemplo de 5 contenedores cuyos pesos son respectivamente 1, 5, 3, 20, 15 toneladas respectivamente y cuya carga máxima es 25.
2. Ordenar dicho vector de orden mayor a menor
3. Comprobar si se ha alcanzado ya a comprobar todos los contenedores y si no en el bucle comprobaría si la suma acumulada de los pesos más el peso del i ésimo contenedor es menor o igual a la carga máxima que puede llevar el buque
4. Si no se ha alcanzado el peso máximo, seleccionaremos el el contenedor i y sumaremos a nuestra suma acumulada de pesos el peso de ese contenedor
5. Después sumaria uno al contenedor para que la siguiente iteración del bucle comprobaria el siguiente contenedor



Traza

1. Para un ejemplo de 5 contenedores cuyos pesos son respectivamente 1, 5, 3, 20, 15 toneladas respectivamente y cuya carga máxima es 25.
2. Ordenar dicho vector de orden mayor a menor
3. Comprobar si se ha alcanzado ya a comprobar todos los contenedores y si no en el bucle comprobaría si la suma acumulada de los pesos más el peso del i ésimo contenedor es menor o igual a la carga máxima que puede llevar el buque
4. Si no se ha alcanzado el peso máximo, seleccionaremos el el contenedor i y sumaremos a nuestra suma acumulada de pesos el peso de ese contenedor
5. Después sumaria uno al contenedor para que la siguiente iteración del bucle comprobaria el siguiente contenedor
6. En otro caso sumaria uno al contenedor para que la siguiente iteración compruebe el siguiente contenedor



Traza

1. Para un ejemplo de 5 contenedores cuyos pesos son respectivamente 1, 5, 3, 20, 15 toneladas respectivamente y cuya carga máxima es 25.
2. Ordenar dicho vector de orden mayor a menor
3. Comprobar si se ha alcanzado ya a comprobar todos los contenedores y si no en el bucle comprobaría si la suma acumulada de los pesos más el peso del i ésimo contenedor es menor o igual a la carga máxima que puede llevar el buque
4. Si no se ha alcanzado el peso máximo, seleccionaremos el el contenedor i y sumaremos a nuestra suma acumulada de pesos el peso de ese contenedor
5. Después sumaria uno al contenedor para que la siguiente iteración del bucle comprobaria el siguiente contenedor
6. En otro caso sumaria uno al contenedor para que la siguiente iteración compruebe el siguiente contenedor
7. Devolver el vector solución con el número de contenedores y sus pesos



Código y Ejecución

Código

Algoritmo Greedy

```
int* AlgoritmoGreedy(int *p, int n, int k, int &tam_sol){
    OrdenaBurbuja(p,n);
    cout <<"Ordenamos los contenedores por pesos: " << endl;
    for (int i = 0; i<n; i++){
        cout << p[i]<<"t ";
    }
    cout << endl<<endl;
    int *solucion = new int[n];
    int i = 0;
    int sum = 0;

    while(i<=n){
        if(sum + p[i] <= k){
            solucion[i] = p[i];
            cout << "Aniadimos contenedor de: " << solucion[i] << "t "<< endl;
            sum = sum+p[i];
            i++;
            tam_sol++;
        }
        else
            i++;
    }
    cout << "No se pueden aniadir mas contenedores...";

    return solucion;
}
```

Algoritmo Ordenacion Burbuja

```
void OrdenaBurbuja(int *v, int n) {

    int aux;

    for(int i = 0; i < n-1; i++){
        for(int j = 0; j < n-1; j++){
            if(v[j+1] > v[j]){
                aux = v[j+1];
                v[j+1] = v[j];
                v[j] = aux;
            }
        }
    }
}
```



Ejecución

```
PS C:\Users\Wavin\Documents\GitHub\Algoritmica> .\ejer.exe 5 25 6 5 3 20 15
Carga maxima del buque: 25
Numero de contenedores: 5
Pesos de los contenedores:
6t 5t 3t 20t 15t
Ordenamos los contenedores por pesos:
20t 15t 6t 5t 3t

Aniadimos contenedor de: 20t
Aniadimos contenedor de: 5t
No se pueden aniadir mas contenedores...

Numero de contenedores aniadidos: 2
    Tiempo de ejec. (us): 1002
PS C:\Users\Wavin\Documents\GitHub\Algoritmica> |
```



Crítica a Chat GPT

Diseño de Componentes

ChatGPT español

- **Función de selección:** Esta función se encarga de seleccionar el contenedor con el peso máximo disponible en cada paso.
- **Función de factibilidad:** Verifica si agregar el contenedor seleccionado superaría la capacidad máxima del buque.
- **Función de evaluación:** Calcula la suma total de los pesos de los contenedores seleccionados.



Diseño de Componentes

ChatGPT español

- **Función de selección:** Esta función se encarga de seleccionar el contenedor con el peso máximo disponible en cada paso.
- **Función de factibilidad:** Verifica si agregar el contenedor seleccionado superaría la capacidad máxima del buque.
- **Función de evaluación:** Calcula la suma total de los pesos de los contenedores seleccionados.

ChatGPT inglés

- Input:
 - `k`: Cargo capacity of the ship (in tons)
 - `containers`: A list of containers, each with its weight (in tons)
- Output:
 - `elected_containers`: A subset of containers to be loaded on the ship



Diseño del Algoritmo

ChatGPT español

- Ordena los contenedores en orden descendente según su peso.
- Inicializa una lista vacía para almacenar los contenedores seleccionados.
- Inicializa la capacidad total del buque a k toneladas.
- Para cada contenedor en la lista ordenada: a. Si la capacidad total del buque es mayor o igual al peso del contenedor actual, agrega el contenedor a la lista de seleccionados y reduce la capacidad total del buque por el peso del contenedor. b. Si no, pasa al siguiente contenedor.
- Devuelve la lista de contenedores seleccionados y la suma total de sus pesos



Diseño del Algoritmo

ChatGPT español

- Ordena los contenedores en orden descendente según su peso.
- Inicializa una lista vacía para almacenar los contenedores seleccionados.
- Inicializa la capacidad total del buque a k toneladas.
- Para cada contenedor en la lista ordenada: a. Si la capacidad total del buque es mayor o igual al peso del contenedor actual, agrega el contenedor a la lista de seleccionados y reduce la capacidad total del buque por el peso del contenedor. b. Si no, pasa al siguiente contenedor.
- Devuelve la lista de contenedores seleccionados y la suma total de sus pesos

ChatGPT inglés

- Sort the containers in non-increasing order of their weights.
- Initialize an empty list called `selected_containers` to store the containers to be loaded.
- Initialize a variable called `remaining_capacity` with the value of k.
- Iterate through the sorted list of containers: a. If the weight of the current container is less than or equal to the `remaining_capacity`, add it to `selected_containers` and subtract its weight from `remaining_capacity`. b. If the weight of the current container is greater than the `remaining_capacity`, skip it and move to the next container.
- Return `selected_containers`.



Estudio de la optimalidad

ChatGPT español

El algoritmo greedy propuesto no garantiza una solución óptima en todos los casos. Sin embargo, en este problema específico, el enfoque greedy ofrece una solución óptima debido a la propiedad de subestructura óptima. La propiedad de subestructura óptima significa que una solución óptima a un subproblema se puede combinar con soluciones óptimas de subproblemas más pequeños para obtener una solución global óptima.

ChatGPT inglés

The algorithm uses a greedy strategy by selecting containers in non-increasing order of their weights. By selecting the heaviest containers first, we maximize the weight loaded onto the ship at each step. The algorithm does not consider the value or any other factors related to the containers, only the weight. Therefore, it may not always result in the optimal solution for the maximum total weight. However, it provides a feasible solution with a good approximation, especially when the weights of the containers are relatively similar.



Traza

ChatGPT español

Supongamos que tenemos un buque mercante con una capacidad de carga de 15 toneladas y los siguientes contenedores con sus respectivos pesos:
Contenedores: c1 = 7 toneladas, c2 = 10 toneladas, c3 = 4 toneladas, c4 = 3 toneladas

Paso 1: Ordenamos los contenedores en orden descendente según su peso: c2, c1, c3, c4.

Paso 2: Inicializamos una lista vacía para almacenar los contenedores seleccionados.

Paso 3: Inicializamos la capacidad total del buque a 15 toneladas.

Paso 4: Comenzamos a iterar por cada contenedor en la lista ordenada.

En el primer paso, seleccionamos c2 (10 toneladas) porque su peso es menor o igual a la capacidad del buque (15 toneladas). Agregamos c2 a la lista de seleccionados y actualizamos la capacidad del buque a $15 - 10 = 5$ toneladas.

Paso 5: Devolvemos la lista de contenedores seleccionados [c2] y la suma total de sus pesos, que es 10 toneladas.

En este caso, el algoritmo selecciona correctamente el contenedor más pesado que puede ser cargado en el buque, maximizando la suma total de los pesos de los contenedores transportados



Traza

ChatGPT español

Supongamos que tenemos un buque mercante con una capacidad de carga de 15 toneladas y los siguientes contenedores con sus respectivos pesos:
Contenedores: c1 = 7 toneladas, c2 = 10 toneladas, c3 = 4 toneladas, c4 = 3 toneladas

Paso 1: Ordenamos los contenedores en orden descendente según su peso: c2, c1, c3, c4.

Paso 2: Inicializamos una lista vacía para almacenar los contenedores seleccionados.

Paso 3: Inicializamos la capacidad total del buque a 15 toneladas.

Paso 4: Comenzamos a iterar por cada contenedor en la lista ordenada.

En el primer paso, seleccionamos c2 (10 toneladas) porque su peso es menor o igual a la capacidad del buque (15 toneladas). Agregamos c2 a la lista de seleccionados y actualizamos la capacidad del buque a $15 - 10 = 5$ toneladas.

Paso 5: Devolvemos la lista de contenedores seleccionados [c2] y la suma total de sus pesos, que es 10 toneladas.

En este caso, el algoritmo selecciona correctamente el contenedor más pesado que puede ser cargado en el buque, maximizando la suma total de los pesos de los contenedores transportados

ChatGPT inglés

Let's consider a simple example to understand how the algorithm works step by step: Ship's cargo capacity (k): 10 tons Containers and their weights (p1, p2, ..., pn): [5, 4, 7, 3, 1, 2]

Step 1: Sorting the containers in non-increasing order of weights: [7, 5, 4, 3, 2, 1]

Step 2: Initialization:

- selected_containers = []
- remaining_capacity = 10

Step 3: Iterating through the sorted list of containers:

Current container weight = 7

- $7 \leq 10$ (remaining_capacity)
- Add container to selected_containers: [7]
- remaining_capacity = $10 - 7 = 3$
-

Current container weight = 5

- $5 \leq 3$ (remaining_capacity)
- Add container to selected_containers: [7, 5]
- remaining_capacity = $3 - 5 = -2$ (not enough capacity)

Step 4: The remaining containers are not considered since their weights are greater than the remaining capacity.

Step 5: Return selected_containers = [7, 5]

Therefore, the algorithm selects containers with weights 7 and 5, with a total weight of 12 tons, which is the maximum weight that can be transported within the given cargo capacity.



Código Propuesto

ChatGPT español

ChatGPT inglés

```
// Función de comparación para ordenar los contenedores por peso en orden descendente
bool compararPorPesoDesc(const Contenedor& c1, const Contenedor& c2) {
    return c1.peso > c2.peso;
}

// Función para cargar los contenedores en el buque
std::vector<Contenedor> cargarContenedores(int capacidadBuque, const std::vector<Contenedor>& contenedores) {
    std::vector<Contenedor> seleccionados;
    int capacidadActual = capacidadBuque;

    // Ordenar los contenedores en orden descendente por peso
    std::vector<Contenedor> contenedoresOrdenados = contenedores;
    std::sort(contenedoresOrdenados.begin(), contenedoresOrdenados.end(), compararPorPesoDesc);

    // Iterar por cada contenedor y seleccionar los que se pueden cargar
    for (const Contenedor& contenedor : contenedoresOrdenados) {
        if (contenedor.peso <= capacidadActual) {
            seleccionados.push_back(contenedor);
            capacidadActual -= contenedor.peso;
        }
    }

    return seleccionados;
}
```

```
std::vector<int> maximizeContainerWeight(int k, const std::vector<int>& containers) {
    std::vector<int> sortedContainers(containers);
    std::sort(sortedContainers.rbegin(), sortedContainers.rend()); // Sort in non-increasing order

    std::vector<int> selectedContainers;
    int remainingCapacity = k;

    for (int container : sortedContainers) {
        if (container <= remainingCapacity) {
            selectedContainers.push_back(container);
            remainingCapacity -= container;
        }
    }

    return selectedContainers;
}
```



Comparación de las ejecuciones

ChatGPT español

```
PS C:\Users\Wavin\Documents\GitHub\Algoritmica\EjercicioPropuesto> .\chatES.exe
Contenedores seleccionados:
Id: 4 Peso: 20
Id: 2 Peso: 5

Suma total de pesos: 25 toneladas
```

ChatGPT inglés

```
PS C:\Users\Wavin\Documents\GitHub\Algoritmica\EjercicioPropuesto> .\chatEN.exe
Selected containers: 20 5
```



Crítica

Enlaces de interés

Códigos: <https://github.com/waving969/CriticaAlgoritmoChatGPT>

Conversación Español ChatGPT: <https://chat.openai.com/share/08288528-9552-47f8-809b-6e606105162a>

Conversación Inglés ChatGPT: <https://chat.openai.com/share/74e44e54-1df8-43cd-9368-0f6017c88485>