Mochila Fracionária - Análise de Complexidade

O algoritmo da Mochila fracionária consiste em, dados um conjunto de elementos com valor e peso e um reservatório com uma determinada capacidade (metaforicamente, a mochila), calcular como obter o melhor valor possível composto por elementos do primeiro conjunto, sem superar a capacidade do reservatório. Para criar o algoritmo, foi utilizada a linguagem C com as seguintes estruturas:

```c
typedef struct item {
 int id;
 int weight;
 int value;
} Item;

typedef struct term {
 int id;
 double value;
} Term;
```

Das quais "Item" possui identificadores e representa os elementos de valor e peso do primeiro conjunto e Term representa o id do item e um termo multiplicativo, que vai de 0 a 1 (por isso Mochila Fracionária).
O algoritmo a seguir recebe um array de itens "items", o tamanho desse array "n" e a capacidade máxima do reservatório "w"; devolvendo um array de termos.

```c
Term* fractionalBackpack(Item* items, int n, int w) {
 Term *terms = malloc(n * sizeof(Term));
 int c = w;

 mergeSort(items, n, sizeof(Item), compareItemsByProportion);
 for (int j = n - 1; j >= 0; j--) {
   terms[j].id = items[j].id;
   int currentWeight = items[j].weight;
   if (c >= currentWeight) {
```

```
      c -= currentWeight;
      terms[j].value = 1;
    } else {
      terms[j].value = (double)c / (double)currentWeight;
      c = 0;
    }
  }

  return terms;
}
```

Observações do algoritmo: O método mergeSort é um algoritmo de ordenação genérico, isto é, aceita qualquer tipo de dado através do uso de ponteiros void* na linguagem C. Por ser genérico, precisa de um método de comparação entre elementos, que no caso é a comparação por proporção. Assumiremos que a complexidade do mergeSort é O(n log(n)). Como logo abaixo teremos um laço “para” que executará para todos os elementos invariavelmente, atribuindo diferentes valores conforme a lógica do algoritmo para cada elemento, podemos concluir que esse algoritmo possui complexidade O(n.log(n)) + O(n), portanto, sua complexidade dominante é O(n.log(n)) (determinada pelo algoritmo de ordenação).