Mochila Binária - primeiras soluções

Técnicas de Programação - 2025/02

Última atualização: 06/10/2025 11:03

Vimos na parte expositiva que a mochila binária é um problema complexo e vamos tentar algumas soluções simples para sua resolução. Duas soluções razoáveis e úteis são

1. selecionar os objetos mais caros primeiro. Se houver empate, selecione o de menor índice primeiro.
2. selecionar os objetos mais leves primeiro. Se houver empate, selecione o de menor índice primeiro.

Em todos os algoritmos (e implementações deste módulo receberemos como argumentos:

1. `N` - número de objetos
2. `C` - capacidade máxima da nossa mochila
3. `V` - array com o valor de cada objeto
4. `P` - array com o peso de cada objeto

A saída de cada algoritmo será

1. `O` - array de `VERDADEIRO/FALSO` indicando se o objeto `I` foi selecionado
2. `VALOR_TOTAL`
3. `PESO_TOTAL`, que deverá ser menor que `C`

Mais caro primeiro

A ideia desta heurística é não deixar nenhum objeto valioso para trás! Por isso vamos ser ganaciosos e pegar **primeiro os objetos mais caros**! Se um objeto valioso não couber passamos para os mais baratos e prosseguimos até examinar todos objetos.

> Relembrando...
>
> Repare que aqui usaremos a mesma ideia do *SLEXSORT*: ordenamos um array de índices com base no valor de outro array (peso ou valor).

Diferente do *SLEXSORT*, não podemos mexer nos arrays `V` e `P`. Por isso, vamos criar um novo array `IDS` contendo os índices tal que $V[IDS[i]] \geq V[IDS[j]] \ \ \forall i < j$. Ou seja, o $IDS[0]$ contém o índice do maior elemento de V, $IDS[1]$ contém o índice do segundo maior e assim por diante. No algoritmo, vamos nos referir a esse passo como `ORDENAR IDS USANDO COMO CHAVE O ARRAY V`.

Essa ideia se chama **ordenação indireta**. Ao invés de ordenar `IDS` por seu valor, o faremos usando o valor de um segundo array. Quando formos comparar os índices `I` e `J` de `IDS` faremos a comparação entre `V[IDS[I]` e `V[IDS[J]]`. Ou seja, no fim `IDS` contém os índices de elementos de `V` tal que a ordem desses elementos é (de)cresente.

> Usos de ordenação indireta
>
> Esse truque é muito útil quando precisamos ordenar uma lista de elementos mas não podemos mexer no array/lista que guarda esses elementos.

Exercício: Complete o pseudo código abaixo para implementar essa ideia.

```
MAIS_CARO(N, C, V, P)

IDS <- ARRAY ordenado como descrito acima
```

Exercício: Qual a complexidade computacional do algoritmo `MAIS-CARO`? Qual é a operação mais cara feita?

Mais leve primeiro

Vamos testar uma abordagem oposta: **quantidade agora é o foco**. Por isso vamos ser práticos e pegar **o maior número de objetos possível**! Começaremos agora pelos objetos mais leves e vamos torcer para que a quantidade grande de objetos selecionados resulte em uma mochila com alto valor.

Exercício: Escreva abaixo um algoritmo em pseudo-código para implementar a heurística descrita acima.

```
MAIS_LEVE(N, C, V, P)

IDS <- ARRAY criado com ordenação indireta usando P
```

Você deve ter percebido que as mudanças foram mínimas! Isso ocorre pois só mudamos o critério de preenchimento da mochila. O algoritmo em sí é o mesmo.

Analisando nossas soluções

Exercício: Crie uma entrada em que a heurística do mais valioso seja muito melhor que a do mais leve. Escreva abaixo as saídas de cada programa.

Exercício: Crie uma entrada em que a heurística do mais leve seja muito melhor que a do mais valioso. Escreva abaixo as saídas de cada programa.

Exercício: Com base nas suas respostas acima, em quais situações a heurística do mais valioso é melhor?

Exercício: Com base nas suas respostas acima, em quais situações a heurística do mais leve é melhor?