Projeto e Análise de Algoritmos

Flávio L. C. de Moura
Departamento de Ciência da Computação
Universidade de Brasília[1]

11 de junho de 2025

[1]flaviomoura@unb.br

Capítulo 1

Algoritmos Gulosos

Nesta seção, exploraremos uma poderosa técnica para resolver problemas de otimização: os *algoritmos gulosos* (*greedy algorithms*).

Esses algoritmos seguem uma estratégia intuitiva: a cada passo, eles fazem a escolha que parece mais vantajosa no momento (escolha localmente ótima), com o objetivo de alcançar uma solução globalmente ótima. No entanto, nem sempre essa abordagem garante o melhor resultado final, e entender quando ela é aplicável é fundamental.

Para ilustrar melhor o conceito, começaremos com um exemplo prático antes de detalharmos a técnica em si. Dessa forma, você poderá observar como a estratégia gulosa funciona na prática e quais são suas característica essenciais.

1.0.1 Exemplo

Consideraremos o **Problema da Alocação de Atividades**: Dado um conjunto finito de atividades $S = \{a_1, a_2, \ldots, a_n\}$ tais que cada atividade a_i possui:

- horário de início s_i;
- horário de término t_i, com $0 \leq s_i < t_i < \infty$ para $1 \leq i \leq n$.

Assim, uma atividade, digamos a_i, utiliza o recurso no intervalo $[s_i, t_i)$. O objetivo é selecionar o número máximo de atividades mutuamente compatíveis de S. Duas atividades distintas a_i e a_j são ditas *compatíveis* se seus intervalos não se sobrepõem, ou seja, $s_i \geq t_j$ ou $s_j \geq t_i$.

Assumiremos que as atividades estão ordenadas em ordem crescente de horário de término: $t_1 \leq t_2 \leq \ldots \leq t_n$. Por exemplo,

i	1	2	3	4	5	6	7	8	9	10	11
s_i	1	3	0	5	3	5	6	8	8	2	12
t_i	4	5	6	7	9	9	10	11	12	14	16

Exemplos de subconjuntos de atividades mutuamente compatíveis são $\{a_3, a_9, a_{11}\}$, $\{a_1, a_4, a_8, a_{11}\}$ e $\{a_2, a_4, a_9, a_{11}\}$.

Como no caso de **programação dinâmica**, os algoritmos gulosos devem satisfazer **propriedade da subestrutura ótima**, permitindo que soluções ótimas de subproblemas sejam combinadas para resolver

o problema original.

Para mostrarmos que o problema da alocação de tarefas satisfaz a propriedade da subsestrutura ótima, denotaremos por S_{ij} o subconjunto de S contendo as atividades que iniciam após a atividade a_i terminar, e terminam antes da atividade a_j começar. Queremos encontrar o conjunto máximo de atividades compatíveis de S_{ij}, que denotaremos por A_{ij}. Se A_{ij} possui a atividade a_k então:

- Devemos resolver o subproblemaS_{ik} (atividades entre a_i e a_k), ou seja, devemos encontrar o conjunto A_{ik};

- Devemos resolver o subproblemaS_{kj} (atividades entre a_k e a_j), ou seja, devemos encontrar o conjunto A_{kj}.

Afirmação: Uma solução ótima A_{ij} contém necessariamente soluções ótimas para os subproblemas S_{ik} e S_{kj}.

Prova. Suponha que exista um subconjunto A'_{kj} com mais atividades que A_{kj}, ou seja, tal que $|A'_{kj}| > |A_{kj}|$. Então poderíamos usar A'_{kj} no lugar de A_{kj} e teríamos $|A_{ik}| + |A'_{kj}| + 1 > |A_{ik}| + |A_{kj}| + 1 = |A_{ij}|$ o que contradiz a suposição de que A_{ij} é uma solução ótima. Um argumento análogo pode ser utilizado para A_{ik}. □

A ideia central da estratégia gulosa consiste em fazer a escolha ótima local ao invés de explorar todas as possibilidades como acontece em programação dinâmica. Isso faz com que a estratégia gulosa reduza drasticamente a complexidade do problema.

Para o problema da alocação de atividades, a melhor escolha local consiste em selecionar a atividade que termina primeiro, pois ela libera recurso o mais cedo possível (usa o recurso de forma mínima), maximizando o tempo disponível para as demais atividades. Considerando que as atividades estão ordenadas crescentemente de acordo com o horário de término, a escolha gulosa seria a_1. Note que ao fazermos uma escolha gulosa passamos a ter apenas um novo subproblema para resolver. Denotaremos por $S_k = \{a_i \in S \mid s_i \geq t_k\}$ o conjunto das atividades que iniciam depois do término da atividade a_k. Assim, fazendo a escolha gulosa a_1 teremos apenas o subproblema S_1 para ser resolvido. O subproblema S_1 é obtido eliminando todas as atividades que conflitam com a_1. A ideia é então construir uma solução ótima para S_1 e utilizá-la na construção da solução ótima do problema original. A questão que surge é: Esta ideia funciona? Ela está correta?

Afirmação: Se S_k é um subproblema não vazio e a_m é a atividade em S_k com o menor tempo de término, então a_m faz parte de alguma solução ótima para S_k.

Prova. Seja A_k uma solução ótima para S_k. Se $a_m \in A_k$ então a afirmação está correta. Caso contrário, seja a_j a atividade em A_k que termina primeiro, ou seja, a_j é tal que t_j é o menor valor de término em A_k. Como $t_m \leq t_j$, a substituição de a_j por a_m não causa conflitos com as outras atividades em A_k. Assim, $A'_k = A_k - \{a_j\} \cup \{a_m\}$ também é uma solução ótima e contém a_m. □

Algumas considerações:

1. A estratégia gulosa nem sempre produz uma solução ótima;

2. Para aplicar a estratégia gulosa devemos observar 2 pontos:

   (a) O subproblema deve possuir a **propriedade da subsestrutura ótima**, e;

(b) O subproblema deve possuir a **propriedade da escolha gulosa**: escolhas locais ótimas levam a uma solução global ótima.

1.0.2 Exercícios

1. Considere um sistema monetário com moedas de valores $\{1, 5, 10, 25\}$.

   (a) Utilize a estratégia gulosa para dar um troco de 36 com o menor número possível de moedas.

   (b) Mostre que a estratégia gulosa é ótima para este sistema monetário.

   (c) Agora suponha que as moedas disponíveis são $\{1, 4, 5\}$. A estratégia gulosa ainda funciona para dar um troco de 8? Justifique.

2. Considere o problema da mochila booleana (ou problema da mochila 0-1): Dado um conjunto de n objetos com peso w_i e valor v_i $1 \leq i \leq n$, e uma mochila com capacidade de carregar o peso W, onde W, w_i e v_i são inteiros para $1 \leq i \leq n$, quais objetos devem ser colocados na mochila para que o valor total seja máximo? Mostre que a estratégia gulosa não resolve este problema, e construa uma solução utilizando programação dinâmica para este problema.

3. Considere o problema da mochila fracionária: Considere n objetos com peso w_i e valor v_i $1 \leq i \leq n$, e uma mochila com capacidade de peso W, de forma que frações de cada objeto podem ser selecionadas. Que fração de cada objeto deve ser colocada na mochila de modo a maximizar o valor total? Em outras palavras, selecione frações $f_i \in [0, 1]$ dos itens tais que $\sum_{i=1}^{n} f_i.w_i \leq W$ e $\sum_{i=1}^{n} f_i.v_i$ é máximo. Mostre que este problema possui a propriedade da escolha gulosa.

1.0.3 Leituras complementares

1. [1] (Capítulo 16)

2. [2] (Capítulo 9)

Referências Bibliográficas

[1] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein. *Introduction to Algorithms, Third Edition.* The MIT Press, 3rd edition, 2009.

[2] A. V. Levitin. *Introduction to the Design and Analysis of Algorithms, Third Edition.* Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2012.