Material Didático sobre Algoritmos Gulosos

Victor de Oliveira Colombo
Instituto de Matemática e Estatística da Universidade de São Paulo

Objetivos

- Apresentar algoritmos gulosos de uma maneira pragmática, a partir de problemas de competições de programação, como a Maratona de Programação, destoando dos problemas clássicos que são frequentemente abordados nos livros-texto.
- Resolver problemas que utilizem algoritmos gulosos aliados a outros tópicos, como Programação Dinâmica, Busca Binária e estruturas de dados.
- Desenvolver o raciocínio e a intuição por trás das técnicas gulosas, a fim de criar uma ferramenta sistemática para resolvê-los.

Problema exemplo: Problema da Partição modificado

- É dado um vetor de $\mathbf{n}$ números inteiros, $\mathbf{V = \{v_1, v_2, \ldots, v_n\}}$.
- Queremos encontrar $\mathbf{r = \{r_1, r_2, \ldots, r_n\}}$, com $\mathbf{r_i \in \{-1, 1\}}$ para todo $\mathbf{i \in [1, n]}$, tal que:
$$\sum_{i=1}^{n} v_i * r_i = 0$$
- Ao contrário do Problema da Partição clássico, que é NP-completo, há a restrição de que $\mathbf{1 \leq v_i \leq i}$ para todo $\mathbf{i \in [1, n]}$.

Desenvolvimento

- É fácil ver que se $\sum_{i=1}^{n} \mathbf{v_i}$ é ímpar, não há solução.
- Caso contrário, cada partição deve somar:
$$\frac{\sum_{i=1}^{n} v_i}{2}$$
- Ideia: Encaminhar o elemento $\mathbf{v_i}$ para a partição mais vazia.
- **Não funciona** quando se itera de $\mathbf{1}$ a $\mathbf{n}$.
  Contraexemplo: $\mathbf{V = \{1, 2, 3, 4\}}$ resulta em partições de somas distintas $\mathbf{4}$ e $\mathbf{6}$.
- **Funciona** quando se itera de $\mathbf{n}$ a $\mathbf{1}$.

Ideias para demonstração

- Na iteração $\mathbf{n - i + 1}$ do algoritmo, definir a "folga" de cada partição, $\mathbf{a_i}$ e $\mathbf{b_i}$.
- Mostrar, por contradição, que alguma partição terá "folga" suficiente em todas as iterações.
- Utilizar a restrição $\mathbf{1 \leq v_i \leq i}$ e quebrar em dois casos: paridade de $\mathbf{a_i}$ e $\mathbf{b_i}$ são iguais ou diferentes.
- Montar intervalos de $\mathbf{a_i + b_i}$ e notar que suas intersecções são vazias, levando ao absurdo.

Corolários e Implementação

- $\sum_{i=1}^{n} \mathbf{v_i}$ é par $\Leftrightarrow$ Há solução.
- Se existe "folga" suficiente nas duas partições, o elemento pode ser encaminhado para **qualquer** partição (não necessáriamente a menor).

```
função SomaVetor(v, n)
    soma ← 0
    para i de 1 até n faça
        soma ← soma + v_i
    devolve soma
função Resolve(v, n)
    soma ← SomaVetor(v, n)
    se soma%2 ≠ 0 então
        devolve Impossível
    folgaA ← soma/2
    r ← {0}^n
    para i de n até 1 faça
        se folgaA > v_i então
            folgaA ← folgaA − v_i
            r_i ← 1
        senão
            r_i ← −1
    devolve r
```

Figura 1: Pseudocódigo para resolução do Problema da Partição modificado

- Consome tempo $\mathbf{O(n)}$ e memória $\mathbf{O(1)}$.

Argumento de troca

- É muito comum que problemas que aceitam soluções gulosas possuam diversas outras soluções ótimas.
- Demonstrações por contradição que assumem a existência de uma solução ótima que é diferente da solução gulosa são **insuficientes**.
- Alternativa: Tomamos uma solução ótima "mais parecida" com a solução gulosa possível.
- A solução gulosa for ótima, elas coincidirão.
- Encontrar a contradição o assumindo que as escolhas diferem em algum momento. Tentar "trocar" a decisão da solução ótima pela solução gulosa e mostramos que tal troca não altera o resultado.

Problema exemplo: Salto do sapo

Existem pedras $\mathbf{n}$ pedras numa reta numérica, em posições distintas $\mathbf{v_1, v_2, \ldots, v_{n-1}, v_n}$. Dizemos que o sapo pode saltar de uma pedra $\mathbf{v_i}$ para outra pedra $\mathbf{v_j}$ desde que a distância entre elas seja menor ou igual a $\mathbf{\Delta}$. Um sapo está inicialmente na pedra $\mathbf{v_1}$. Qual é o menor número de saltos que ele precisa dar para chegar na pedra $\mathbf{v_n}$?

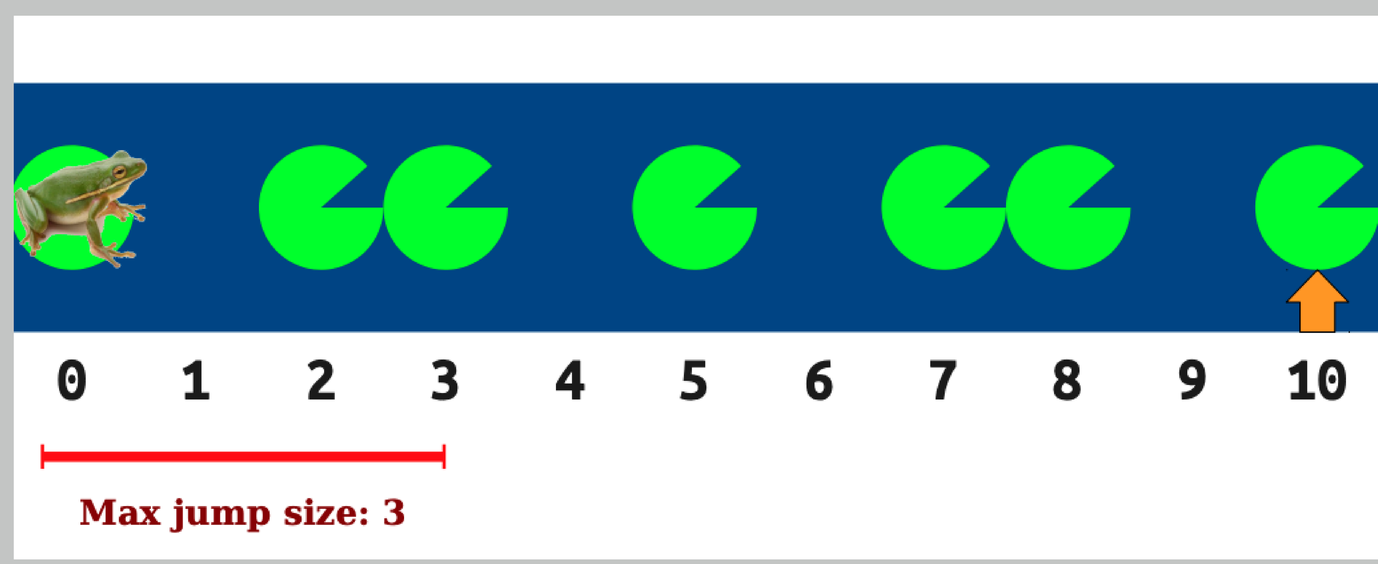

Figura 2: Exemplo para $\mathbf{v = \{0, 2, 3, 5, 7, 8, 10\}}$ e $\mathbf{\Delta = 3}$.
Fonte: https://web.stanford.edu/class/archive/cs/cs161/cs161.1138/lectures/13/Slides13.pdf

Algumas soluções para a instância acima são:

- $\mathbf{0 \to 3 \to 5 \to 8 \to 10}$
- $\mathbf{0 \to 2 \to 5 \to 7 \to 10}$

Desenvolvimento

- Intuitivamente, nunca vale a pena "voltar", isto é, escolher um número menor que o escolhido anteriormente, pois isto aumentaria desnecessariamente a sequência, já que poderíamos descartar a escolha anterior e escolher apenas o menor número diretamente.
- É intuitivo também que sempre vale a pena dar o maior salto possível, pois não há vantagem de estar numa pedra mais distante do objetivo.
- Disso segue um simples algoritmo: a cada iteração, saltar para a pedra mais distante da atual (em direção ao destino) que esteja no alcance do sapo.

Idea de demonstração

- Seja uma sequência de saltos produzida pelo algoritmo guloso $\mathbf{u = \{u_1, u_2, \ldots, u_k\}}$ e uma sequência de saltos ótima $\mathbf{u^* = \{u_1^*, u_2^*, \ldots, u_{k^*}^*\}}$ com maior prefixo de escolhas em comum com $\mathbf{u}$.
- Suponha que $\mathbf{u \neq u^*}$. Seja $\mathbf{l}$ o índice da primeira diferença entre $\mathbf{u}$ e $\mathbf{u^*}$.
- Podemos **trocar** $\mathbf{u_l^*}$ por $\mathbf{u_l}$ mantendo a solução ótima válida.
- Encontramos assim $\mathbf{\bar{u} = \{u_1, u_2, \ldots, u_l, u_{l+1}^*, u_{l+2}^*, \ldots, u_{k^*}^*\}}$ que é tão bom quanto $\mathbf{u^*}$ mas mais com maior prefixo em comum com $\mathbf{u}$, contrariando a hipótese.

Veja mais

Mais problemas e demonstrações completas diponíveis na monografia, acessível em https://linux.ime.usp.br/~colombo/mac0499/