Análise de Algoritmos

Introdução à
Projeto e análise de algoritmos

Projeto e análise de algoritmos

Introdução aos Algoritmos Gulosos

Técnica para projeto de algoritmos eficientes

Permite atender requisitos de performance em casos de problemas complexos

Adequada em situações específicas

Algoritmos Gulosos

Técnica voltada para:

- Problemas de otimização
- Trabalha com a melhor escolha em cada ponto do processo, para assim compor uma solução ótima

Algoritmos Gulosos

Técnica voltada para:

- Problemas de otimização
- Trabalha com a melhor escolha em cada ponto do processo, para assim compor uma solução ótima

Visão geral de funcionamento:

- Quando subproblemas podem ocorrer de forma repetida em diversas etapas

Algoritmos Gulosos

Técnica voltada para:

- Problemas de otimização
- Trabalha com a melhor escolha em cada ponto do processo, para assim compor uma solução ótima

Visão geral de funcionamento:

- Quando subproblemas podem ocorrer de forma repetida em diversas etapas
- Trata cada situação de forma a realizar a melhor escolha naquele momento

Algoritmos Gulosos

Técnica voltada para:

- Problemas de otimização
- Trabalha com a melhor escolha em cada ponto do processo, para assim compor uma solução ótima

Visão geral de funcionamento:

- Quando subproblemas podem ocorrer de forma repetida em diversas etapas
- Trata cada situação de forma a realizar a melhor escolha naquele momento
- Considera que a soma das melhores escolhas parciais leva à melhor escolha global

Algoritmos Gulosos

Comparação com programação dinâmica:

- Ambas tratam de um problema geral a partir de uma subdivisão em subproblemas menores
- O resultado final depende da integração dos resultados parciais

Algoritmos Gulosos

Comparação com programação dinâmica:

- Ambas tratam de um problema geral a partir de uma subdivisão em subproblemas menores
- O resultado final depende da integração dos resultados parciais

Diferença básica:

- Algoritmos gulosos usam abordagens mais simples, para escolhas locais consideradas ótimas
- Possibilidade de redução dos aspectos tratados em cada interação

Algoritmos Gulosos

Voltados para problemas de otimização

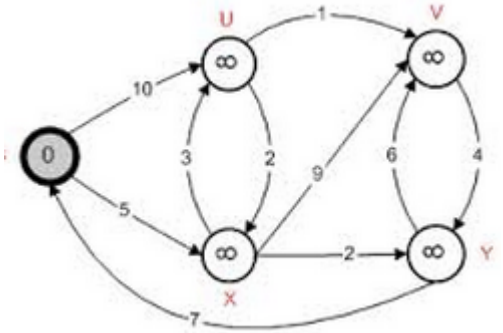
Problemas de otimização

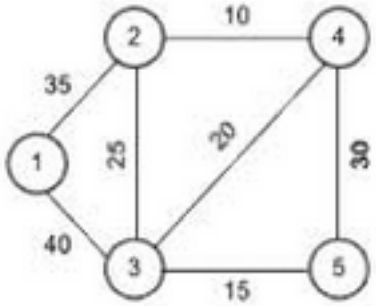
- Podem ter diversas soluções possíveis
- Valores ótimos dependem de contextualização
- Existem parâmetros de aceitação
- Existem várias soluções ótimas

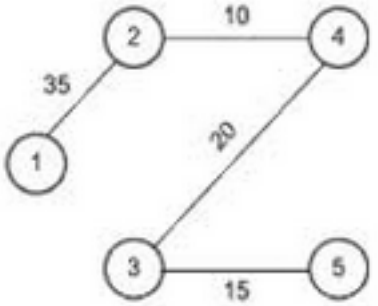
Algoritmos Gulosos

Exemplos de problemas tratados:

- Seleção de tarefas
- Árvore geradora mínima
- Caminhos mais curtos em grafos







Algoritmos Gulosos – estrutura geral

```
função ALGORITMOGULOSO(C: conjunto)                  ▷ C é o conjunto de candidatos
    S ← ∅                                            ▷ S é o conjunto que irá conter a solução
    enquanto C ≠ ∅ e não solução(S) faça
        x ← seleciona C
        C ← C {x}
        se é viável S ∪ {x} então
            S ← S ∪ {x}
        fim se
    fim enquanto
    se solução(S) então
        retorne S
    senão
        retorne "Não existe solução!"
    fim se
fim função
```

Fonte: Rocha

Algoritmos Gulosos

Etapas gerais:

- Determinar a subestrutura ótima do problema

Algoritmos Gulosos

Etapas gerais:

- Determinar a subestrutura ótima do problema
- Descrever o processo recursivo de solução

Algoritmos Gulosos

Etapas gerais:

- Determinar a subestrutura ótima do problema
- Descrever o processo recursivo de solução
- Analisar o resultado da escolha gulosa (deve ser somente um subproblema)

Algoritmos Gulosos

Etapas gerais:

- Determinar a subestrutura ótima do problema
- Descrever o processo recursivo de solução
- Analisar o resultado da escolha gulosa (deve ser somente um subproblema)
- Avaliar se a escolha gulosa é sempre correta e segura

Algoritmos Gulosos

Etapas gerais:

- Determinar a subestrutura ótima do problema
- Descrever o processo recursivo de solução
- Analisar o resultado da escolha gulosa (deve ser somente um subproblema)
- Avaliar se a escolha gulosa é sempre correta e segura
- Desenvolver um algoritmo recursivo

Algoritmos Gulosos

Etapas gerais:

- Determinar a subestrutura ótima do problema
- Descrever o processo recursivo de solução
- Analisar o resultado da escolha gulosa (deve ser somente um subproblema)
- Avaliar se a escolha gulosa é sempre correta e segura
- Desenvolver um algoritmo recursivo
- Transformar o algoritmo de recursivo em iterativo

Algoritmos Gulosos

Abordagem geral

- Expressar o problema de otimização de forma que:
  - Seja definida uma única escolha
  - Seja identificado um único subproblema para resolver

Algoritmos Gulosos

Abordagem geral

- Expressar o problema de otimização de forma que:
  - Seja definida uma única escolha
  - Seja identificado um único subproblema para resolver
- Mostrar que existe uma escolha ótima e que a escolha gulosa é sempre segura

Algoritmos Gulosos

Abordagem geral

- Expressar o problema de otimização de forma que:
  - Seja definida uma única escolha
  - Seja identificado um único subproblema para resolver
- Mostrar que existe uma escolha ótima e que a escolha gulosa é sempre segura
- Demonstrar a subestutura ótima:
  - A solução ótima para o problema, combinada com a escolha gulosa leva à solução global ótima

Algoritmos Gulosos

Propriedades da escolha gulosa (1)

- Montar uma solução global ótima fazendo escolhas locais ótimas

Algoritmos Gulosos

Propriedades da escolha gulosa (1)

- Montar uma solução global ótima fazendo escolhas locais ótimas
- A cada momento, considera a escolha a fazer, sem analisar os demais subproblemas

Algoritmos Gulosos

Propriedades da escolha gulosa (1)

- Montar uma solução global ótima fazendo escolhas locais ótimas
- A cada momento, considera a escolha a fazer, sem analisar os demais subproblemas
- Ocorre de cima para baixo, reduzindo cada instância do problema a um problema menor

Algoritmos Gulosos

Propriedades da escolha gulosa (1)

- Montar uma solução global ótima fazendo escolhas locais ótimas
- A cada momento, considera a escolha a fazer, sem analisar os demais subproblemas
- Ocorre de cima para baixo, reduzindo cada instância do problema a um problema menor
- Diferente de programação dinâmica: a seleção da melhor opção não depende das demais seleções; não age de baixo para cima.

Algoritmos Gulosos

Propriedades da escolha gulosa (2)

- Aplicável em problemas com subestrutura ótima:
  - Uma solução ótima para o problema é também a solução ótima para os subproblemas

Algoritmos Gulosos

Propriedades da escolha gulosa (2)

- Aplicável em problemas com subestrutura ótima:
  - Uma solução ótima para o problema é também a solução ótima para os subproblemas
- Avaliar se a escolha gulosa em um subproblema, em conjunto com a escolha anterior, leva à solução ótima

Algoritmos Gulosos

Exemplo simples – troco de moedas

Visão geral: como identificar o menor número de moedas para dar como troco para um determinado valor?

Algoritmos Gulosos

Exemplo simples – troco de moedas

Visão geral: como identificar o menor número de moedas para dar como troco para um determinado valor?

Versão intuitiva: repetição de um ciclo com a escolha da maior moeda possível, a cada etapa.

Algoritmos Gulosos

Exemplo simples – troco de moedas

Visão geral: como identificar o menor número de moedas para dar como troco para um determinado valor?

Versão intuitiva: repetição de um ciclo com a escolha da maior moeda possível, a cada etapa.

Ex.: R$ 2,89 [100, 50, 25, 10, 1]

Algoritmos Gulosos

Exemplo simples – troco de moedas

Visão geral: como identificar o menor número de moedas para dar como troco para um determinado valor?

Versão intuitiva: repetição de um ciclo com a escolha da maior moeda possível, a cada etapa.

Ex.: R$ 2,89 [100, 50, 25, 10, 1]
➔ 2x1,00 + 3x0,25 + 1x0,10 + 4*0,01 [10]

Algoritmos Gulosos

Exemplo simples – troco de moedas

Visão geral: como identificar o menor número de moedas para dar como troco para um determinado valor?

Versão intuitiva: repetição de um ciclo com a escolha da maior moeda possível, a cada etapa.

Ex.: R$ 2,89 [100, 50, 25, 10, 1]
➔ 2x1,00 + 3x0,25 + 1x0,10 + 4*0,01
➔ 2x1,00 + 1x0,50 + 1x0,25 + 1*0,10+ 4*0,01 [9]

Algoritmos Gulosos

Exemplo simples – troco de moedas

função TROCO(n)
    const $C \leftarrow \{100, 25, 10, 5, 1\}$ ▷ C é o conjunto de moedas
    $S \leftarrow \emptyset$ ▷ S é o conjunto que irá conter a solução
    $s \leftarrow 0$ ▷ s é a soma dos itens em S
    enquanto $s \neq n$ **faça**
        $x \leftarrow$ o maior item em C tal que $s + x \leq n$
        se este item não existe **então**
            retorne "Não foi encontrada uma solução!"
        fim se
        $S \leftarrow A \cup \{$uma moeda de valor x$\}$
        $s \leftarrow s + x$
    fim enquanto
    retorne S
fim função

Fonte: Rocha