PROJETO E ANÁLISE DE ALGORITMOS

# ALGORITMO GULOSO

O algoritmo guloso é uma estratégia de solução de problemas que toma decisões localmente ótimas em cada estágio na esperança de alcançar uma solução globalmente ótima. Esse algoritmo simples e intuitivo pode ser aplicado para resolver qualquer problema de otimização que exija o resultado ótimo máximo ou mínimo.

Para implementar uma solução gulosa é necessário que siga as propriedades a seguir:

* Propriedade da escolha gananciosa: escolher a melhor opção em cada fase pode levar a uma solução ótima global (geral).
* Subestrutura ótima: se uma solução ótima para o problema completo contém as soluções ótimas para os subproblemas, o problema tem uma subestrutura ótima.

Para gerar uma solução gulosa é necessário seguir as etapas abaixo:

* Etapa 1: em um determinado problema, encontre a melhor subestrutura ou subproblema.
* Etapa 2: determine o que a solução incluirá (por exemplo, maior soma, caminho mais curto).
* Etapa 3: crie um processo iterativo para analisar todos os subproblemas e criar uma solução ideal.

Interface gráfica do usuário, Texto, Aplicativo, Email

Descrição gerada automaticamente

# ALGORITMO DA MOCHILA

Esse é um dos problemas clássicos da computação: imagine que você tem que planejar uma longa viagem e tem uma mochila para caber os vários itens que precisa carregar, por exemplo, uma seleção de frutas que vai lhe nutrir durante toda a viagem. Como você tem uma capacidade limitada nesta mochila, decidiu levar diferentes quantidades de diferentes frutas, mas o objetivo final é maximizar a “ingestão de calorias” que você obtém de sua mochila cheia.

Interface gráfica do usuário, Aplicativo

Descrição gerada automaticamente

# ALGORITMOS DE PROGRAMAÇÃO DINÂMICA

Programação dinâmica é quebrar os problemas em subproblemas e salvar o resultado para o futuro, para que não tenhamos que calcular o mesmo problema novamente. A otimização adicional de subproblemas que otimiza a solução geral é conhecida como propriedade de subestrutura ótima.

Duas maneiras pelas quais a programação dinâmica pode ser aplicada:

* Top-Down: Nesse método, o problema é decomposto e se o problema já estiver resolvido, o valor salvo é retornado, caso contrário, o valor da função é memorizado, ou seja, será calculado pela primeira vez; para todas as outras vezes, o valor armazenado será chamado de volta;
* Bottom-Up: Aqui, as soluções para pequenos problemas são calculadas, o que cria a solução para o problema geral.

A programação dinâmica é uma estratégia para linearizar problemas de programação exponencialmente difíceis, ou seja, a ideia é armazenar os resultados dos subproblemas para que não tenhamos que recalculá-los posteriormente.

# ALGORITMOS DE DIVISÃO E CONQUISTA

O algoritmo de divisão e conquista envolve três etapas em cada nível de recursão:

* Dividir: Como sugerido pelo nome, nesta etapa dividimos o problema em subproblemas menores até que o problema seja pequeno o suficiente para ser resolvido.
* Conquistar: Agora vamos resolvê-las realizando operações básicas. Uma vez que os subproblemas se tornam pequenos o suficiente para que não seja mais recursiva, dizemos que a recursão finalizou e que chegamos ao caso base. Quando chegamos ao caso base, resolvemos o subproblema.
* Combine: Mesclamos a saída da etapa de conquista para resolver subproblemas maiores.

O algoritmo de divisão e conquista tem muitos benefícios que o tornam um algoritmo de resolução de problemas muito útil, afinal, ajuda a resolver problemas difíceis em 3 etapas simples usando o algoritmo top-down recursivamente.

# BACKTRACKING

Trata-se de uma técnica algorítmica cujo objetivo é usar força bruta para encontrar todas as soluções para um problema, o que implica compilar gradualmente um conjunto de todas as soluções possíveis. Como um problema terá restrições, as soluções que não as atenderem serão removidas.

Ele encontra uma solução construindo-a passo a passo, aumentando os níveis ao longo do tempo através de chamadas recursivas. Uma árvore de busca conhecida como árvore de espaço de estados é usada para encontrar essas soluções, em que cada ramo em uma árvore de espaço de estados representa uma variável e cada nível representa uma solução.