PCC104 - Projeto e Análise de Algoritmos

Marco Antonio M. Carvalho

Departamento de Computação
Instituto de Ciências Exatas e Biológicas
Universidade Federal de Ouro Preto

Conteúdo

Fonte

Este material é baseado nos livros

- T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein. *Introduction to Algorithms*. The MIT Press, 3rd edition, 2009.
- S. Halim. *Competitive Programming*. 3rd Edition, 2013.
- Ian Parberry and William Gasarch. *Problems on Algorithms*. Second Edition, 2002.
- Ian Parberry *Lecture Notes on Algorithm Analysis and Complexity Theory*. Fourth Edition, 2001.

Licença

Recapitulando... Algoritmo

Definição Informal

É qualquer procedimento computacional bem definido que toma algum valor ou conjunto de valores como **entrada** e produz algum valor ou conjunto de valores como **saída**. Portanto, é uma sequência de passos computacionais que transformam uma entrada na saída.

Definição Informal 2

Uma ferramenta para resolver um problema computacional bem especificado.

O enunciado do problema especifica em termos gerais o relacionamento entre entrada e saída.

O algoritmo descreve um procedimento computacional para alcançar esse relacionamento da entrada com a saída.

Problema Computacional

Definição

Um problema computacional pode ser visto como uma coleção infinita de instâncias junto com uma solução para cada uma delas.

Definição Informal

Um problema computacional é uma questão geral a ser respondida, possuindo determinados parâmetros.

Exemplo

Problema de Ordenação

Ordenar uma sequência de números de maneira crescente.

Entrada

Uma sequência de n números $<a_1, a_2, \ldots, a_n>$.

Saída

Uma permutação $<a'_1, a'_2, \ldots, a'_n>$ da sequência de entrada, tal que $a'_1 < a'_2 < \ldots < a'_n$.

Instância de um Problema

Uma **instância** de um problema consiste na entrada necessária para se calcular uma solução para o problema, por exemplo <31, 41, 59, 25, 47>.

Problema Computacional

Problema de Decisão

Tipo de problema computacional em que a resposta para cada instância é **sim** ou **não**:

"Dadas uma lista de cidades e as distâncias entre todas, há uma rota que visite todas as cidades e retorne à cidade original com distância total menor do que 500 km?"

Problema de Otimização

Tipo de problema computacional em que é necessário determinar a **melhor** solução possível entre todas as soluções viáveis.

"Dadas uma lista de cidades e as distâncias entre todas, determine a menor rota que visite todas as cidades e retorne à cidade original."

Definição

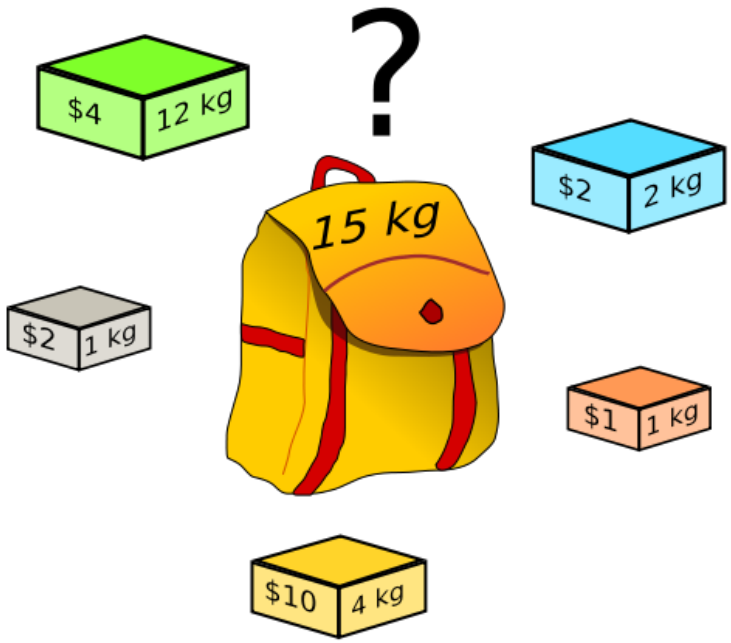
Um problema combinatório é um problema que possui um conjunto de **elementos** (ou **variáveis**) e para sua solução é exigida uma combinação de um subconjunto destes elementos.

Diferentes combinações possuem diferentes valores, porém, o **objetivo** é **otimizar** a solução (achar a de maior valor – **maximização** ou a de menor valor – **minimização**) de acordo com a **função objetivo** ou **função de avaliação**.

As combinações são limitadas por **restrições**, que são regras que definem se uma combinação é **viável** ou **inviável**.

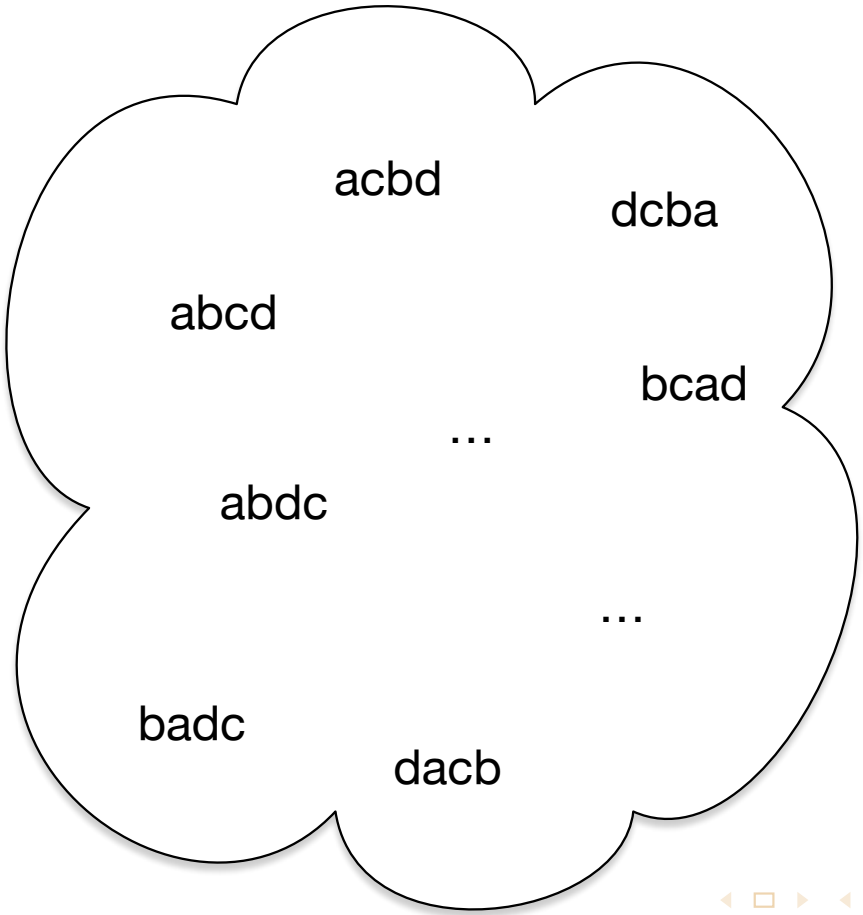
O Problema da Mochila 0-1

Dadas uma mochila de capacidade W e uma lista de n itens distintos e únicos (enumerados de 1 a n), cada um com um peso $w_1, w_2, \ldots, w_n$ e um valor $v_1, v_2, \ldots, v_n$, maximizar o valor carregado na mochila, respeitando sua capacidade.



Mais Definições

O espaço de soluções de um problema combinatório é o conjunto de todas as soluções possíveis, podendo ser restrito às soluções viáveis ou não.



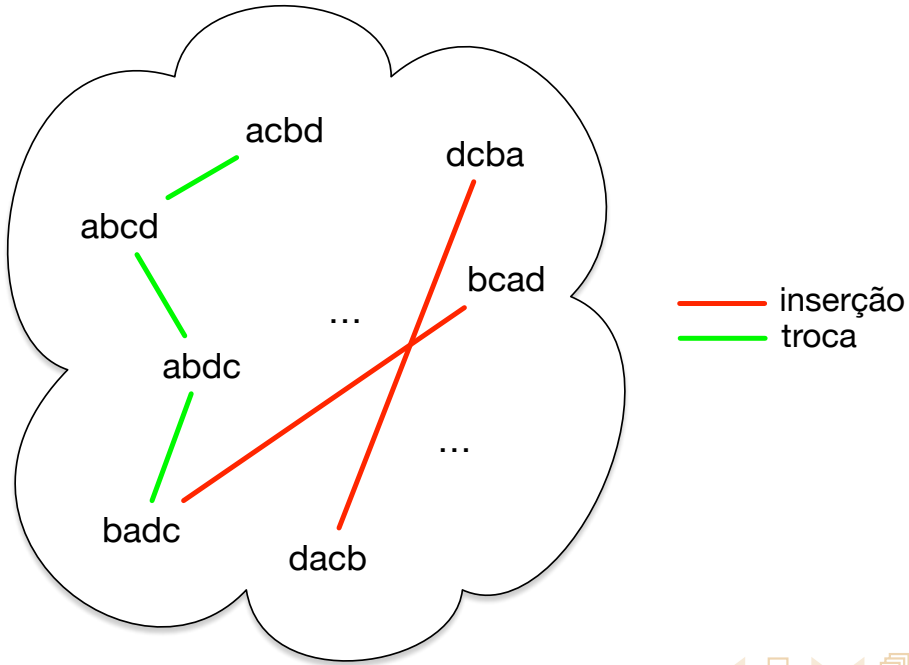
Problemas Combinatórios

Uma Mente Brilhante - "A cena do Pentágono"
https://youtu.be/aLj8WC0-2QI.

Mais Definições

Ao explorarmos o espaço de soluções utilizando alguma técnica, realizamos movimentos entre soluções, ou seja, a partir de uma solução atual, a alteramos de uma determinada maneira e chegamos a uma outra solução.

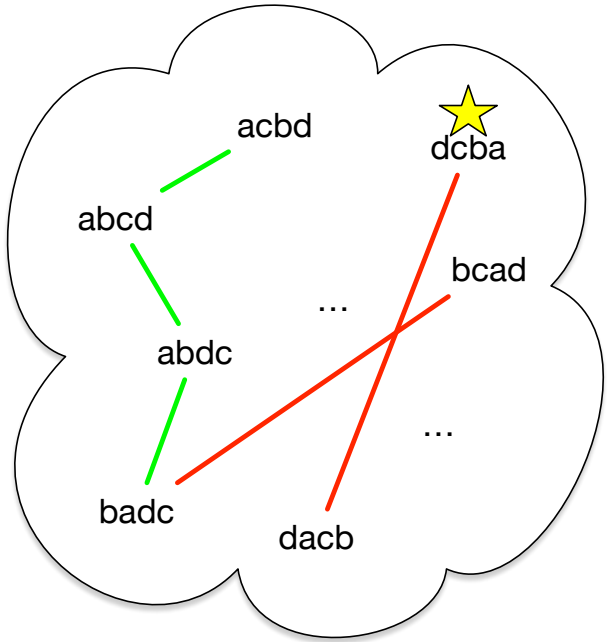
Um movimento induz uma vizinhança no espaço de soluções, tal que as soluções que se diferem entre si por um movimento são ditas vizinhas.



Mais Definições

Uma solução ótima global é uma solução viável que atinge o melhor valor possível de acordo com a função de avaliação de um problema combinatório.

Podemos ter uma ou múltiplas soluções ótimas para um problema, todas com o mesmo valor de avaliação, porém, com configurações diferentes.



Exemplo - O Problema da Mochila 0-1

Os elementos a serem combinados são os itens.

O espaço de soluções são todas as combinações de itens.

Uma função objetivo possível é a soma dos valores dos itens da mochila.

A função objetivo deve ser maximizada.

A única restrição é a de capacidade da mochila.

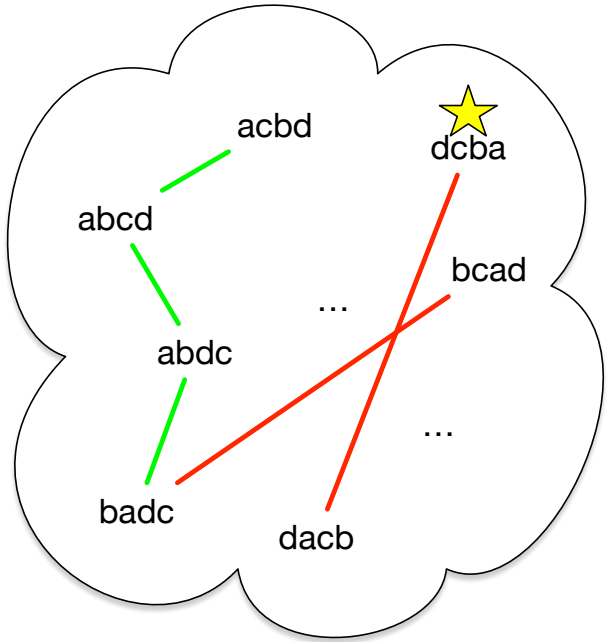
Uma solução ótima global é a solução viável cujos itens possuem a maior soma, ou seja, o melhor valor de função objetivo.

Remover um item da mochila, ou inserir um item caracteriza um movimento.

Mais Definições

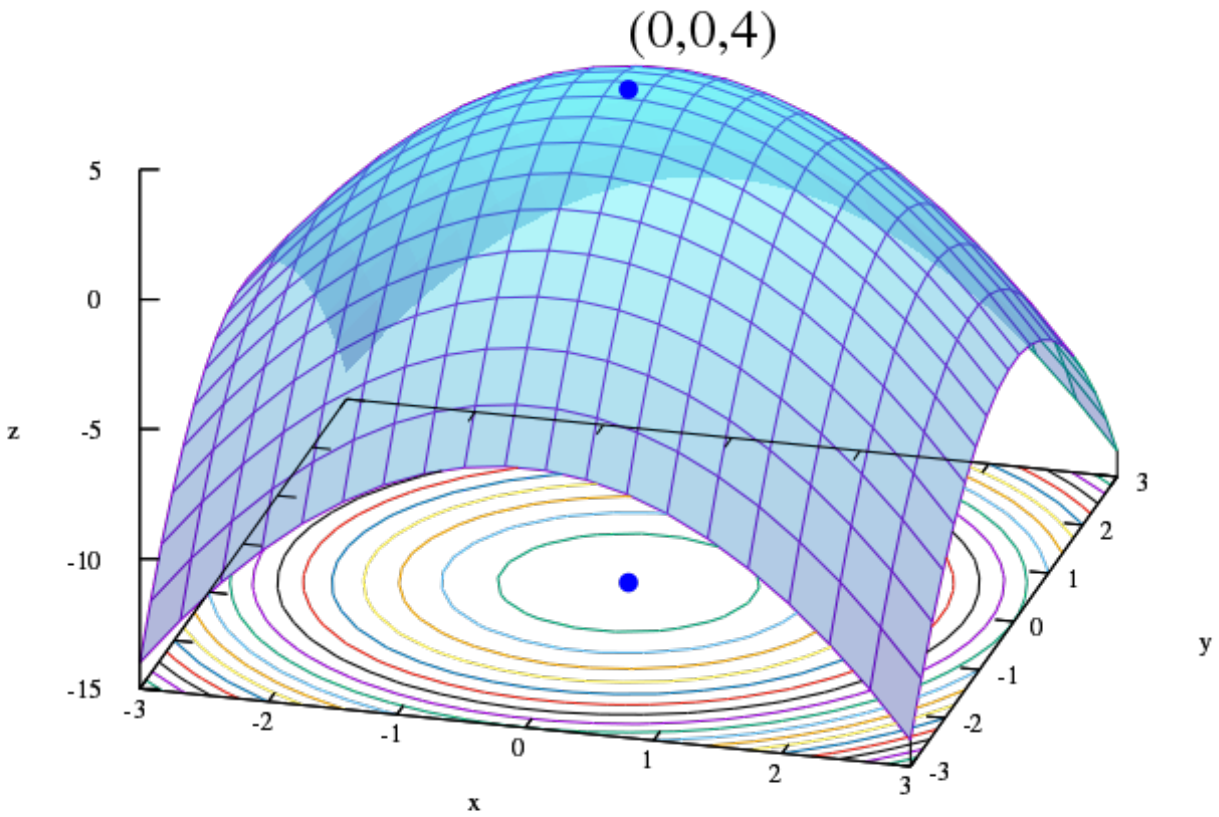
Uma solução ótima global é uma solução viável que atinge o melhor valor possível de acordo com a função de avaliação de um problema combinatório.

Podemos ter uma ou múltiplas soluções ótimas para um problema, todas com o mesmo valor de avaliação, porém, com configurações diferentes.



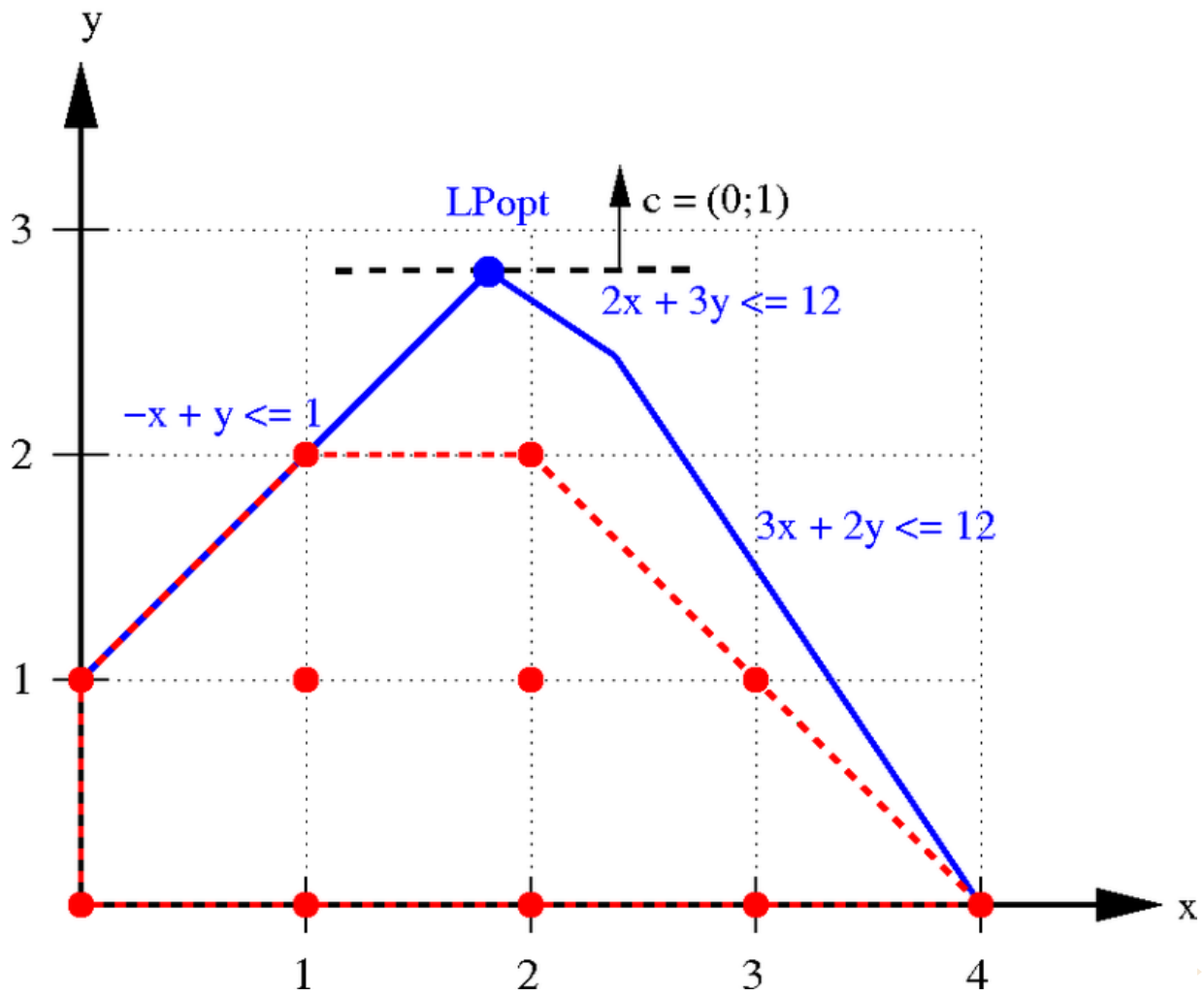
Função Objetivo e Ótimo Global

Gráfico da função $f(x, y) = -(x^2 + y^2)+4$. A solução ótima (0, 0, 4) que maximiza a função é indicada pelo ponto azul central.



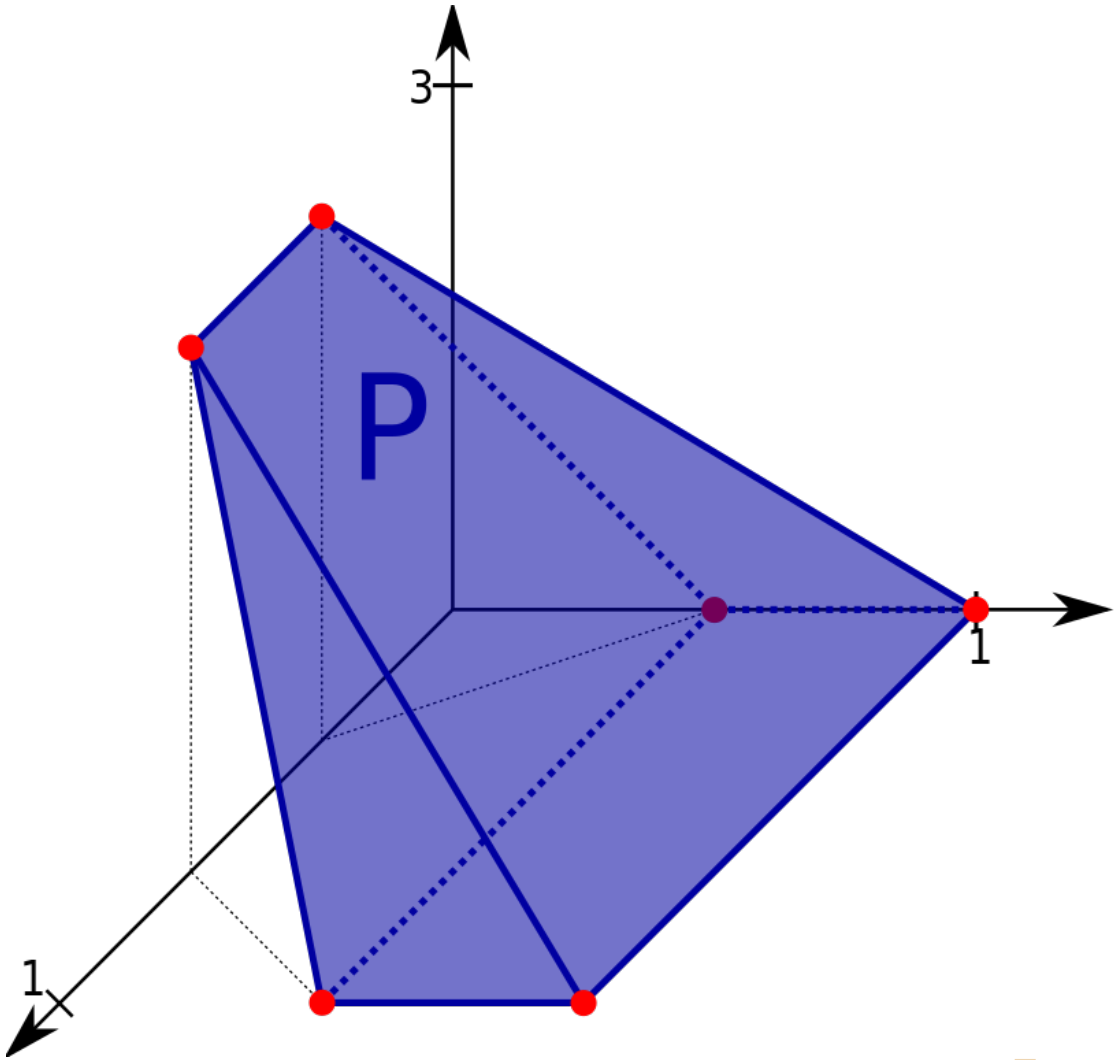
Viabilidade

Um problema com 5 restrições lineares (incluindo as 2 de não negatividade). Na ausência de restrições de integralidade, a região viável é representada em azul, caso contrário, em vermelho.



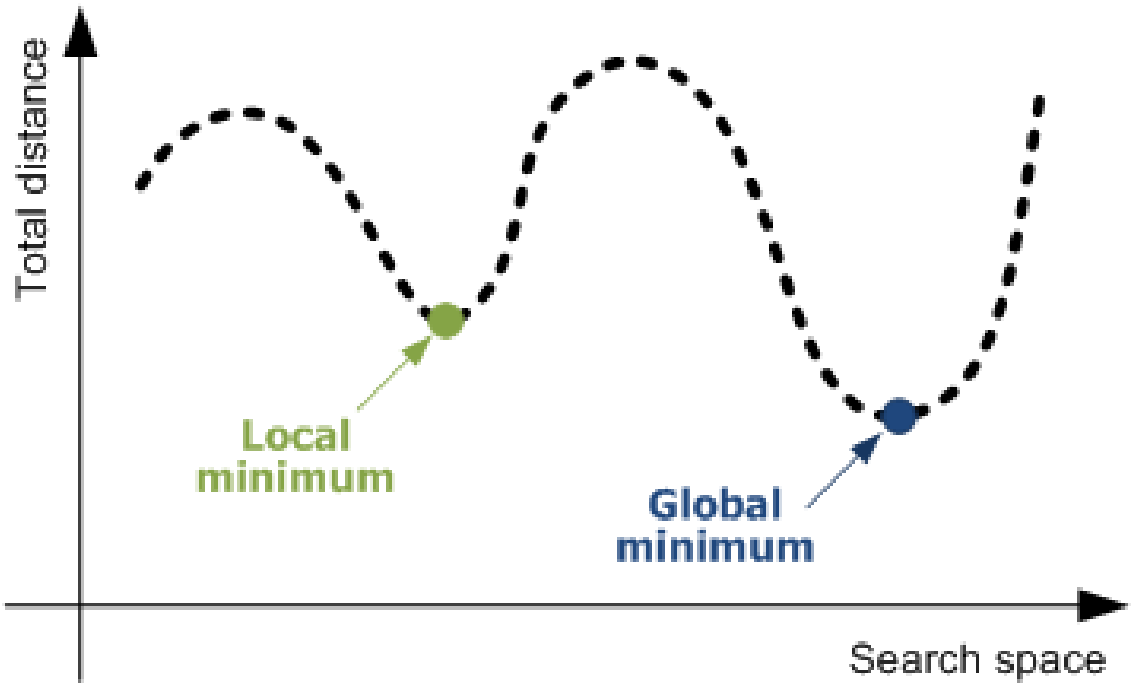
Viabilidade

Uma região viável fechada de um problema com três variáveis, representada por um poliedro convexo.



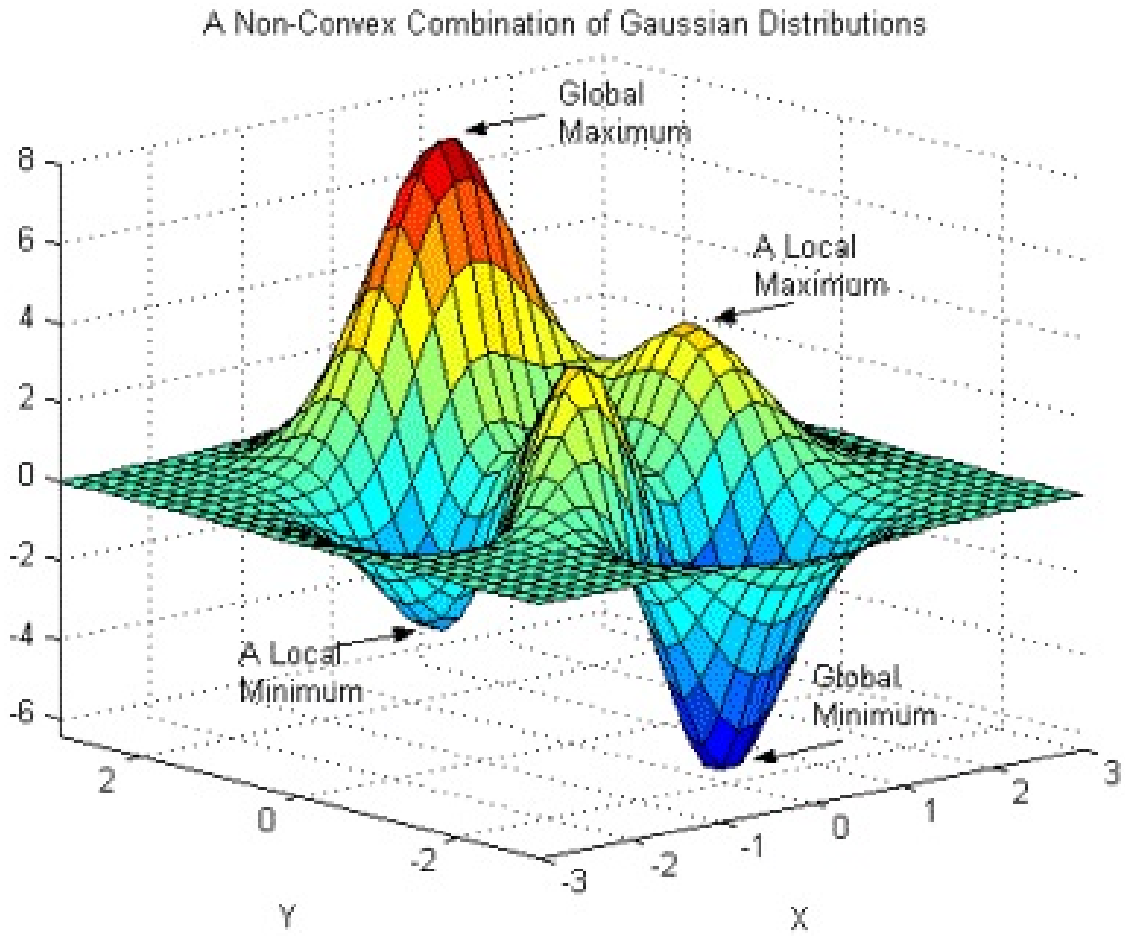
Mais Definições

Uma solução ótima local ou subótima é uma solução viável que atinge o melhor valor de função objetivo de um problema combinatório entre as soluções vizinhas.

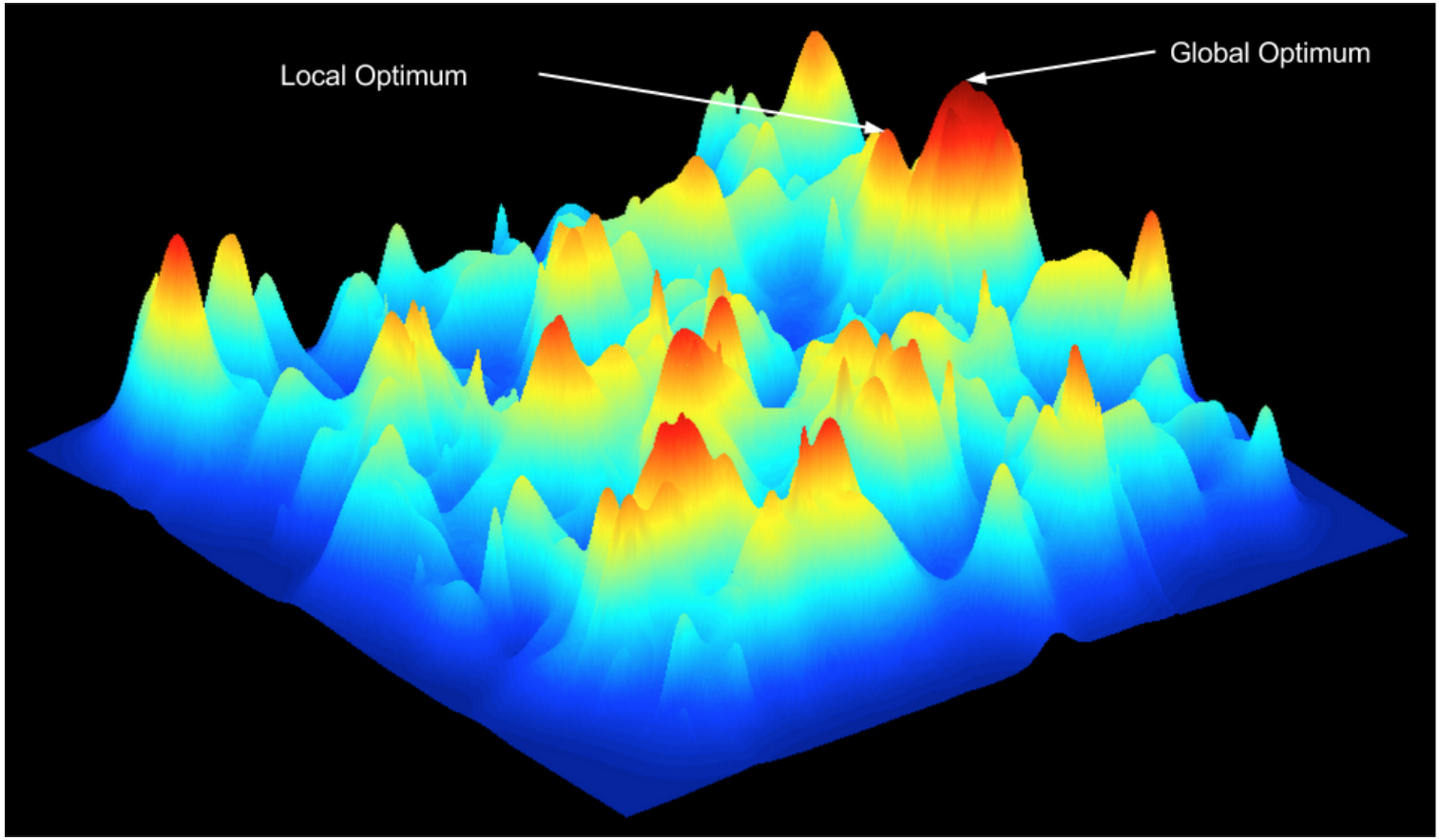


Exemplo de ótimo local e global em um problema de minimização.

Problemas Combinatórios



Problemas Combinatórios



Caracterização

Em resumo, problema combinatório é composto dos seguintes componentes:

- Função objetivo;
- Conjunto de elementos;
- Conjunto de restrições.

O Problema do Caixeiro Viajante

Dadas uma lista finita de cidades e as distâncias entre todas, determine a menor rota que visite todas as cidades e retorne à cidade original, sem repetir nenhuma cidade.

O Problema do Caixeiro Viajante

Elementos: Cidades;

Espaço de Soluções: Todas as permutações da ordem das cidades;

Função Objetivo: Comprimento da rota;

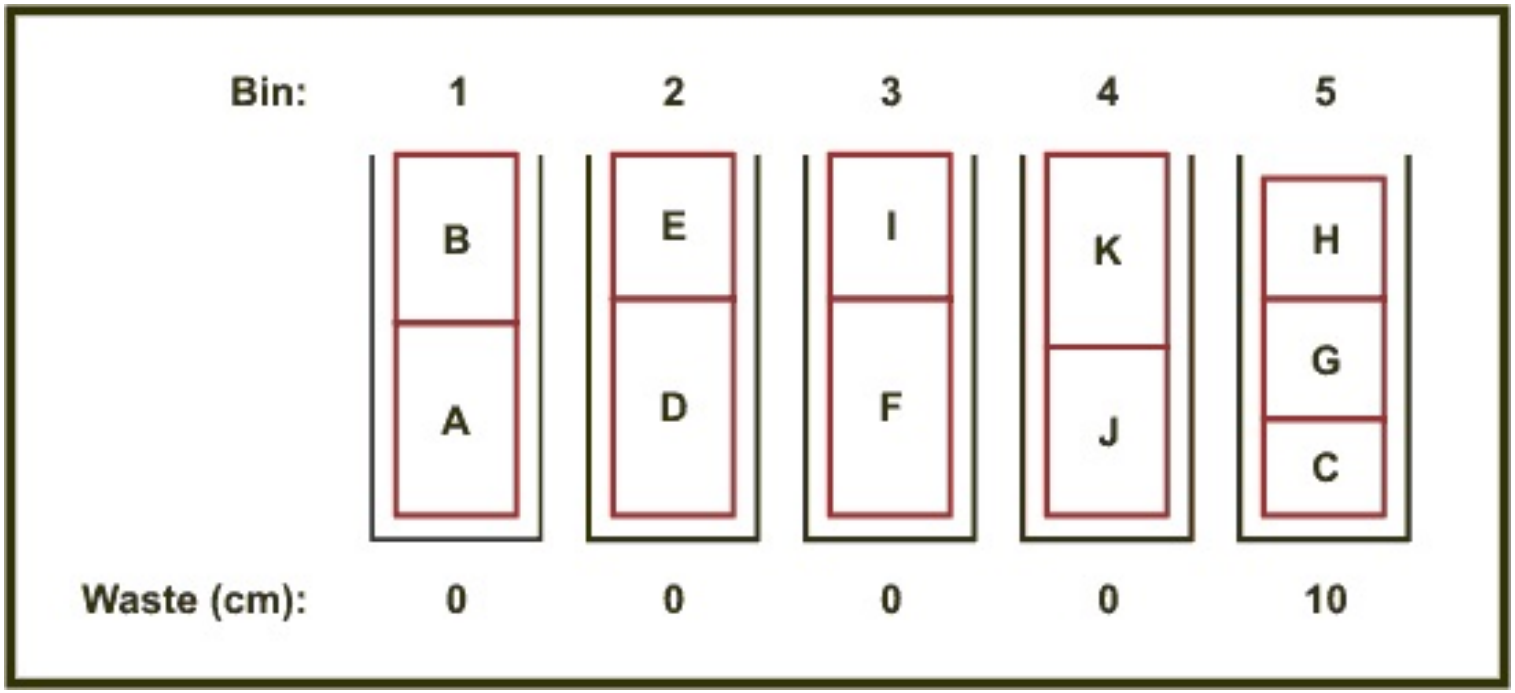
Tipo de Função Objetivo: Minimização;

Restrições: Não repetir cidades, visitar todas as cidades;

Movimento: Trocar a posição de duas cidades na rota.

Bin Packing

Dadas uma lista finita de objetos de volumes diferentes e uma lista finita de caixas de volume V, empacotar todos os objetos minimizando o número de caixas.



Bin Packing

Elementos: Objetos;

Espaço de Soluções: Todas as combinações de objetos e caixas;

Função Objetivo: Número de caixas utilizadas;

Tipo de Função Objetivo: Minimização;

Restrição: Respeitar o volume das caixas;

Movimento: Inserir ou remover um objeto de uma caixa.

Subset Sum

Dado um conjunto (ou multiconjunto) de números inteiros, determinar se há um subconjunto não vazio cuja soma seja exatamente s.

Exemplo

Consideremos o conjunto {-7, -3, -2, 5, 8 } e $s = 0$.

A resposta é *sim*, porque o subconjunto {-3, -2, 5} possui soma s.

Subset Sum

Elementos: os números inteiros;

Espaço de Soluções: todos os subconjuntos do conjunto original;

Função Objetivo: soma dos elementos;

Tipo de Função Objetivo: problema de decisão;

Restrição: a soma dos elementos deve ser s;

Movimento: adicionar ou remover um elemento do subconjunto.

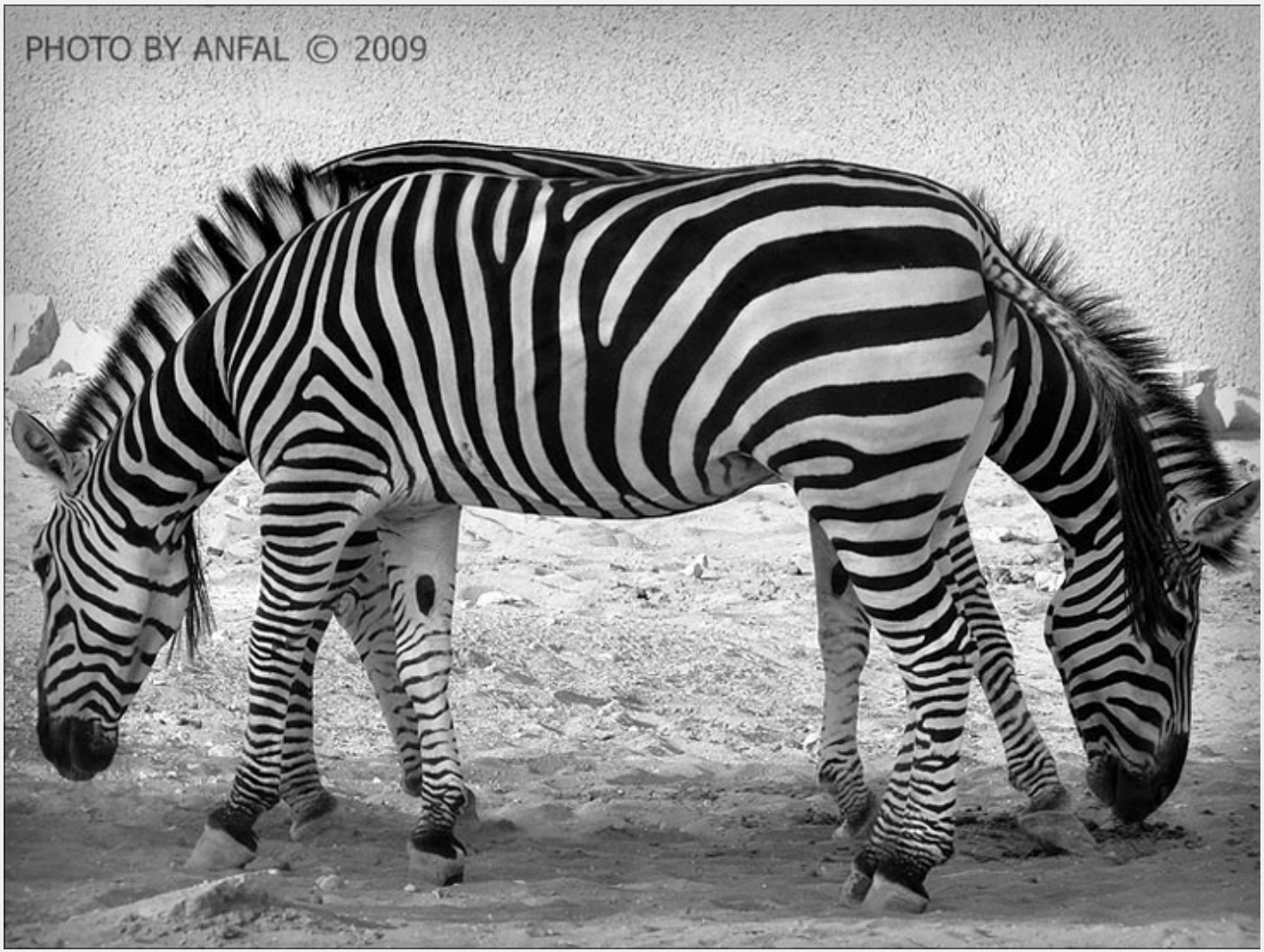
Citação I

"*If all you have is a hammer, everything looks like a nail.*"
- Abraham Maslow, 1966.

Citação II

"*If you change the way you look at things, the things you look at change.*"
- Wayne Dyer.

Solução de Problemas vs. Perspectiva Correta



Introdução

Existem paradigmas clássicos para abordagem de problemas computacionais, cada um adequado a um tipo de problema:

- Busca Completa/*Backtracking*.
- Algoritmos Gulosos.
- Divisão e Conquista.
- Programação Dinâmica.

Dominar estes paradigmas nos auxilia a utilizarmos a "ferramenta" adequada, ao invés de martelarmos todos os problemas.

Observação

Também serão vistos os tópicos abaixo, embora não sejam considerados clássicos, ou mesmo paradigmas propriamente ditos:

- Busca Aleatória.
- Recursividade.
- Algoritmos Aproximados.

Busca Completa

Ian Parberry e William Gasarch, *Problems on Algorithms*

"*Often it appears that there is no better way to solve a problem than to try all possibilities.*

This approach, called exhaustive search, is almost always slow, but sometimes it is better than nothing. It may actually be practical in real situations if the problem size is small enough."

Busca Completa

Definição

Em sua forma mais simples, a **Busca Completa** (enumeração completa, busca exaustiva ou força bruta) consiste em varrer o espaço de soluções inteiro, se necessário, em busca da solução desejada.

Busca Completa

Observações

Este método de solução não é eficiente em termos de complexidade, uma vez que tende a enumerar todas as possíveis soluções.

Por exemplo, em um problema combinatório em que o espaço de soluções é definido pelas permutações de n elementos, a complexidade de um algoritmo de busca completa é $O(n!)$, o que pode ser impraticável mesmo para valores relativamente pequenos de n.

Note que, algoritmos de busca completa possuem a propriedade de **corretude**: encontram a solução desejada para quaisquer instâncias, sempre.

Busca Completa

Exemplo

Determine todos os pares de números de 5 dígitos que possuam os dígitos de 0 a 9 sem repetição, tal que o primeiro número dividido pelo segundo seja igual a um inteiro N dado, em que $2 \leq N \leq 79$.

Em outras palavras $abcde/fghij = N$, em que cada letra representa um dígito diferente. O primeiro dígito pode ser zero, e.g., 79546/01283 = 62; 94736/01528 = 62.

Busca Completa

Análise

Analisando o problema, temos que $fghij$ pode assumir valores entre 01234 e 98765, o que gera $\approx$ 100.000 possibilidades.

Para cada $fghij$, podemos obter o valor de $abcde$ pela multiplicação $fghij \times N$ e verificar se todos os dígitos são diferentes.

Neste caso, a busca completa é viável, embora frequentemente este não seja o caso.

Busca Completa

Geradores vs. Filtradores

Algoritmos que geram todas as possíveis soluções e então selecionam aquela que é a correta (ou removem as incorretas) são chamados **filtradores**.

Os algoritmos que evitam inícios falsos e rumam à solução correta são chamados **geradores**.

Geralmente, filtradores são mais fáceis de projetar, mas são mais lentos. É necessário fazer as contas para determinar se é necessário um filtrador ou um gerador.

Aprimoramentos

A busca completa pode ser aprimorada por técnicas de poda de inviabilidade e de simetrias além do uso de pré-computação, entre outros.

Em breve veremos o paradigma *backtracking*, que pode incorporar muitas destas melhorias.

Busca Aleatória

Definição

Um **algoritmo aleatorizado** emprega um grau de aleatoriedade como parte de sua lógica.

Tipicamente, algum elemento gerado aleatoriamente pela distribuição uniforme é utilizado para guiar o comportamento do algoritmo, na espectativa de obter uma boa performance no caso médio.

Desta forma, a complexidade do algoritmo é uma variável aleatória determinada por seus componentes aleatórios.

Busca Aleatória

Observação

Buscas aleatórias são geralmente úteis quando o espaço de soluções contém a maior parte de soluções consideradas "boas".

Surpreendentemente, mesmo quando boa parte do espaço de soluções possui soluções "boas", não necessariamente é fácil obter uma boa solução deterministicamente.

Busca Aleatória

Las Vegas

Um algoritmo **Las Vegas** é um algoritmo aleatorizado cujo tempo de execução é não determinístico, mas que sempre fornece a resposta correta (ou reporta sua inexistência).

O nome é uma clara referência à cidade americana, famosa pelas apostas.

Monte Carlo

Um algoritmo **Monte Carlo** é um algoritmo aleatorizado cujo tempo de execução é determinístico, mas cuja resposta pode ser incorreta, com uma determinada probabilidade.

O nome também é uma referência clara, desta vez, ao *grand casino* de Mônaco.

Busca Aleatória

Exemplo

Considere o problema de buscar um determinado elemento a em um vetor desordenado A, contendo n elementos, metade deles sendo elementos a e a outra metade sendo elementos b.

Algoritmo Las Vegas

- Escolha um índice uniformemente aleatório i de A;
- Se $A[i] = a$ então terminamos;
- Caso contrário, continuamos selecionando um novo índice aleatório i até que $A[i] = a$ ou até que tenhamos checado todos os elementos de A.

Note que podemos acabar repetindo a escolha do índice aleatório, examinando o mesmo elemento mais do que uma vez.

A probabilidade de sucesso deste algoritmo é 1. A complexidade varia, mas o esperado, sobre várias execuções, é $O(n)$.

Algoritmo Monte Carlo

- Escolha um índice uniformemente aleatório i de A;
- Se $A[i] = a$ então terminamos;
- Caso contrário, continuamos selecionando um novo índice aleatório i até que $A[i] = a$ ou até que tenhamos executado k iterações.

Se a for encontrado, o algoritmo obtém sucesso, caso contrário, o algoritmo falha.

A probabilidade de sucesso deste algoritmo em k iterações é $P(encontrar) = 1 - (1/2)^k$.

A complexidade desde algoritmo é fixa, pois será executado exatamente k vezes, portanto, o algoritmo possui complexidade $\Theta(1)$.

Exemplo

O algoritmo de ordenação *Quicksort*, no pior caso possui complexidade $O(n^2)$, devido à escolha do pivô – normalmente, o elemento mediano.

Porém, se o algoritmo escolher os pivôs aleatoriamente em uma distribuição uniforme, há uma grande probabilidade de ser executado em tempo $O(n\ log\ n)$, independente da estrutura da entrada.

Esta complexidade é exatamente a exigida pelo caso médio da versão determinística do *Quicksort* e também o limitante inferior para a solução do problema de ordenação.

Dúvidas?