

8

Algoritmos gulosos

Entendendo algoritmos

Um guia *ilustrado* para programadores
e outros curiosos

Aditya Y. Bhargava



novatec

MANNING



Neste capítulo

- Você aprenderá como lidar com o impossível: problemas que não têm um algoritmo de solução rápida (problemas NP-completo).
- Você aprenderá como identificar esses problemas ao se deparar com eles, de forma que não perca tempo tentando achar um algoritmo rápido para solucioná-los.

- Você conhecerá os algoritmos de aproximação, que podem ser usados para encontrar, de maneira rápida, uma solução aproximada para um problema NP-completo.
- Você conhecerá a estratégia gulosa, uma estratégia muito simples para resolver problemas.

O problema do cronograma da sala de aula



Suponha que você tenha uma sala de aula e queira reservar o máximo de aulas possível nela. Assim, recebe-se uma lista das aulas.

AULA INÍCIO FIM

ARTES	9 AM	10 AM
INGLÊS	9:30 AM	10:30 AM
MATEMÁTICA	10 AM	11 AM
CC	10:30 AM	11:30 AM
MÚSICA	11 AM	12 PM

Você não pode reservar *todas* essas aulas na sala porque os horários de algumas delas coincidem.

9	9:30	10	10:30	11	11:30	12

ARTES

INGLÊS

MATEMÁTICA

CC

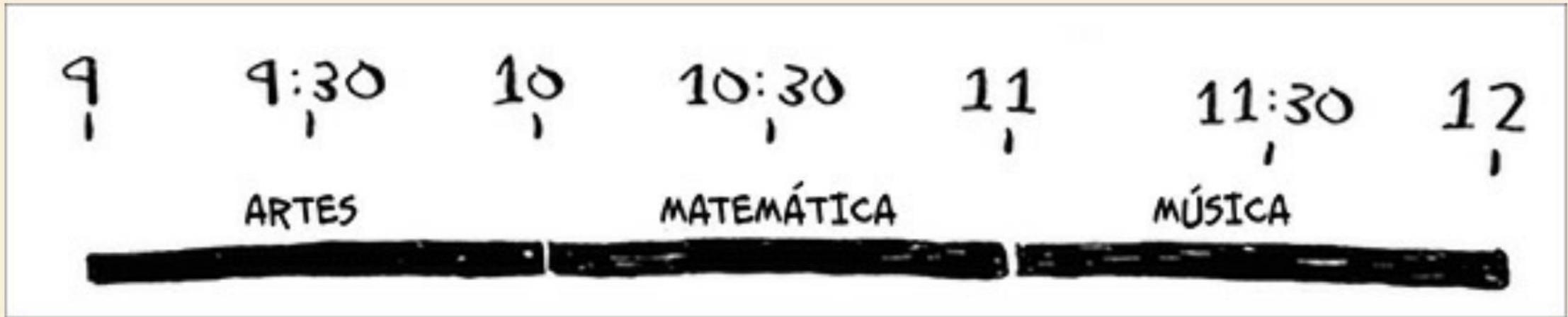
MÚSICA

Soa como um problema difícil, não? Na realidade, o algoritmo é tão simples que pode surpreender.

Aqui temos o funcionamento dele:

1. Pegue a aula que termina mais cedo. Esta é a primeira aula que você colocará nessa sala.
2. Agora você precisa pegar uma aula que comece depois da primeira aula. De novo, pegue a aula que termine mais cedo. Esta é a segunda aula que você colocará

Continue fazendo isso e no final você terá a sua resposta! Vamos testar: Artes termina mais cedo, às 10h00, então esta é a aula escolhida.



Muitas pessoas me dizem que esse algoritmo parece ser fácil. Mas ele é óbvio demais, logo, deve estar errado. No entanto essa é a beleza dos algoritmos guloso (também chamados de algoritmos gananciosos): eles são fáceis! Um algoritmo guloso é simples: a cada etapa, deve-se escolher o movimento ideal. Nesse caso, cada vez que você escolhe uma aula, deve escolher a que acaba mais cedo. Em termos técnicos: *a cada etapa, escolhe-se a solução ideal*, e no fim você tem uma solução global ideal. Acredite ou não, esse algoritmo simples acha a solução ideal para esse problema!

Obviamente os algoritmos gulosos nem sempre funcionam, mas eles são tão simples de escrever!

absolutamente necessários.



O problema da cobertura de conjuntos

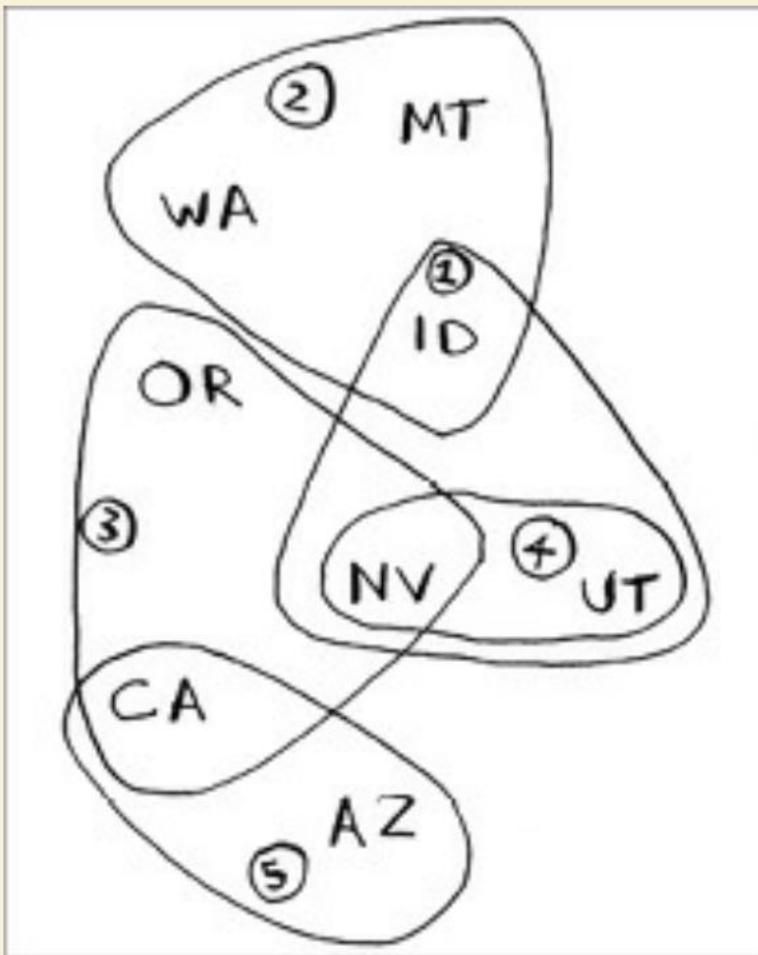
Suponha que você esteja começando um programa de rádio e queira atingir ouvintes em todos os cinquenta estados americanos. É necessário decidir em quais estações transmitir para atingir todos os ouvintes. Porém transmitir em diferentes estações tem um custo, e você está tentando minimizar

o número de estações nas quais você transmite para minimizar o custo. Temos uma lista de estações.

ESTAÇÃO DE RÁDIO	DISPONÍVEL EM
KUM	ID, NV, UT
KDOIS	WA, ID, MT
KTRÊS	OR, NV, CA
KQUATRO	NV, UT
KCINCO	CA, AZ

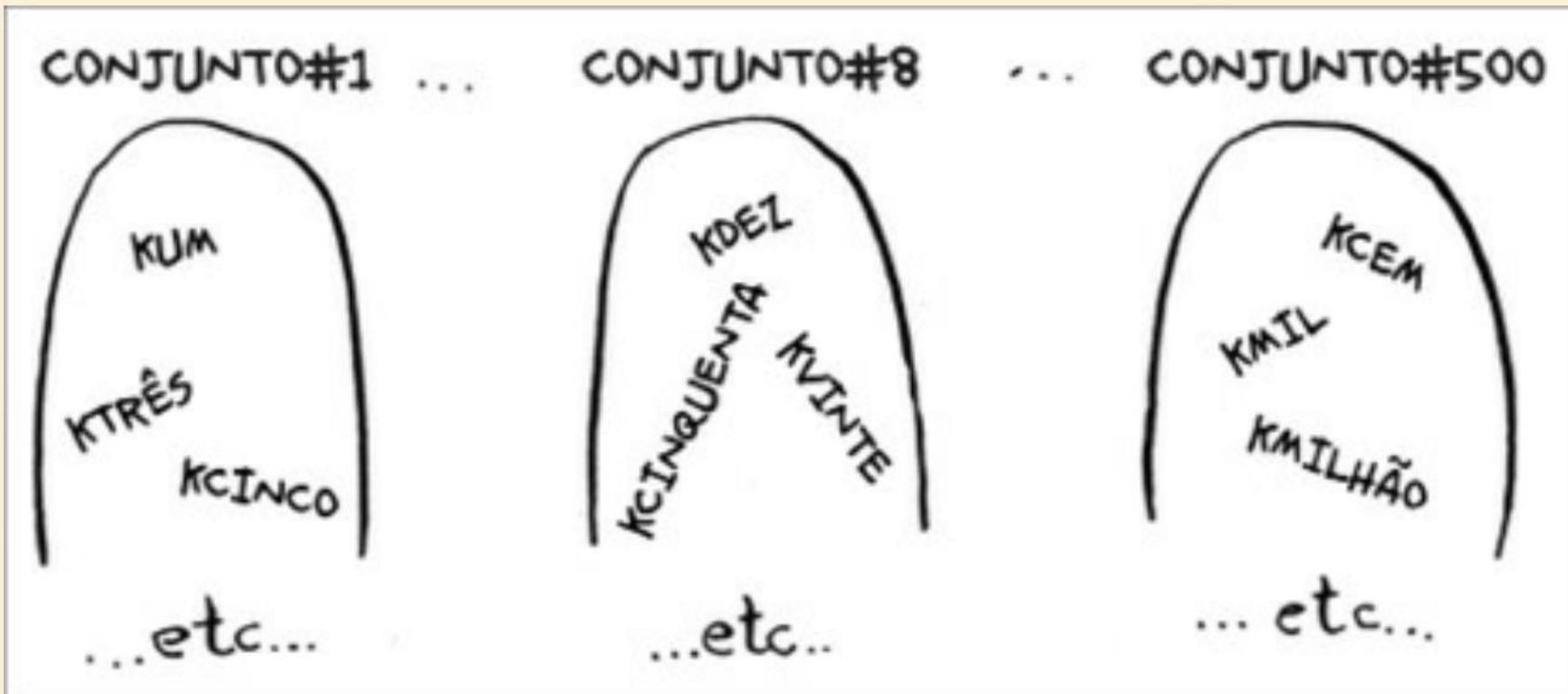
...etc...

Cada estação abrange uma região e existe uma sobreposição.



Como descobrir o menor conjunto de estações nas quais você pode transmitir e abranger os cinquenta estados? Soa fácil, não? Acontece que é extremamente difícil. Aqui está uma solução:

1. Liste cada subconjunto possível de estações. Isso é chamado de *conjunto de partes* (também conhecido como conjunto de potência). Neste caso, existem 2^n possíveis conjuntos.



2. Entre eles, escolha o conjunto com o menor número de estações que abranja todos os cinquenta estados.

O problema neste caso é que o tempo para calcular cada possível subconjunto de estações é muito longo, uma vez que o tempo de execução é $O(2^n)$, pois existem 2^n subconjuntos. Seria possível calcular se você tivesse um grupo pequeno de cinco a dez estações, mas, como em todos os exemplos aqui, pense o que aconteceria se você tivesse muitos itens. O tempo com um maior número de estações será longo demais. Para exemplificar, suponha que você consiga calcular dez subconjuntos por segundo.

Não existe um algoritmo que resolva isso rápido o suficiente! O que você pode fazer?

NÚMERO DE ESTAÇÕES	TEMPO NECESSÁRIO
5	3.2 seg
10	102.4 seg
32	13.6 anos
100	4×10^{21} anos

Algoritmos de aproximação

Algoritmos gulosos ao resgate! Aqui temos um algoritmo goso que chega bem perto da solução:

1. Pegue a estação que abrange o maior número de estados que ainda não foram cobertos. Tudo bem se

a estação abrange alguns estados que já foram cobertos.

2. Repita isso até que todos os estados tenham sido cobertos.

Isto se chama *algoritmo de aproximação*. Quando é necessário muito tempo para calcular a solução exata, um algoritmo de aproximação é uma boa ideia e funciona. Os algoritmos de aproximação são avaliados

- por sua rapidez;
- pela capacidade de chegar à solução ideal.

Os algoritmos gulosos são uma boa escolha porque eles são de fácil compreensão e sua simplicidade também indica que geralmente eles são de rápida execução. Nesse caso, o algoritmo guloso tem tempo de execução $O(n^2)$, em que n é o número de estações de rádio.

Vamos ver como é esse problema em código.