Aula 4 · Soluções Gananciosas

Desafios de Programação

Fernando Kiotheka Vinicius Tikara Date

UFPR

12/05/2023

Soluções gananciosas/gulosas

Um algoritmo ganancioso/guloso constrói a solução de um problema escolhendo sempre o caminho que pareça ser o melhor, isto é, através de ótimos locais.

- Geralmente rápidos, geralmente $\mathcal{O}(n \lg n)$ ou $\mathcal{O}(n)$.
- Pode ser difícil determinar a estratégia gananciosa.
- Os ótimos locais realmente levam ao ótimo global?

Aplicações

- No mundo real, esses algoritmos servem para nos dar boas aproximações para problemas computacionalmente inviáveis.
- Na maratona, eles geralmente fazem parte de um subproblema ou compõe uma questão inteira.
- Podemos criar uma solução lenta, de força bruta e comparar com a nossa solução gananciosa pra ver se ela está correta.

Problema do Troco (conjunto fixo)

Qual o menor número de moedas do conjunto S necessárias para obter o valor V?

$$S = \{1, 2, 5, 10\}$$

Para $V = 25$, por exemplo, podemos usar 3 moedas: $\{10, 10, 5\}$.

Solução gananciosa do Problema do Troco (conjunto fixo)

Acontece que existe uma solução gananciosa para este problema, e é de escolher sempre a maior moeda que não passa do valor desejado (e decrementá-lo com o valor da moeda).
Intuição: estamos tirando o máximo possível do valor atual.

Solução gananciosa do Problema do Troco (conjunto fixo)

Isso não funciona sempre. Tomemos por exemplo $S = \{1, 3, 4\}$ e $V = 6$:

1. Para $V = 6$, escolhemos a moeda 4 (`ans++`)
2. Para $V = 2$, escolhemos a moeda 1 (`ans++`)
3. Para $V = 1$, escolhemos a moeda 1 (`ans++`)

Poderíamos, no entanto, escolher a moeda 3 duas vezes!

Problema de Agendamento de Tarefas

- Conjunto de tarefas da forma (s_i, f_i).
- Tarefas são compatíveis se $f_i \leq s_j$ ou $f_j \leq s_i$.

- Isto é, enquanto se faz uma tarefa não se pode fazer outra.
- Queremos maximizar a quantidade de tarefas realizadas.

Em matematiquês, as tarefas são tratadas como *intervalos abertos*. O objetivo é criar um conjunto de intervalos de tamanho máximo de forma que todos os intervalos não tem intersecção par a par.

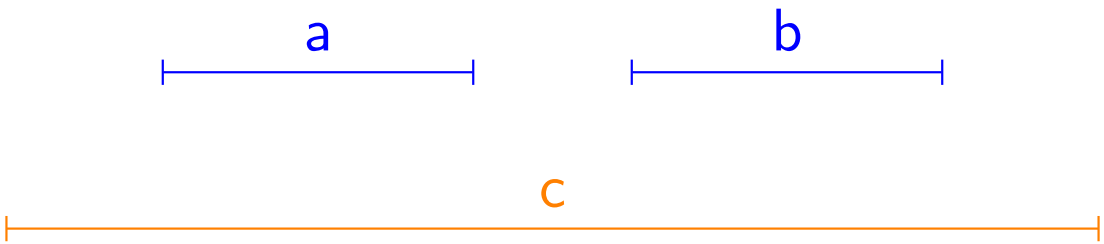
Qual critério utilizar?

Podemos pensar em alguns critérios para selecionar nossos intervalos:

- Escolher o próximo que começa primeiro
- Escolher o próximo que termina primeiro
- Escolher o próximo menor intervalo
- Escolher o próximo intervalo com menos intersecções

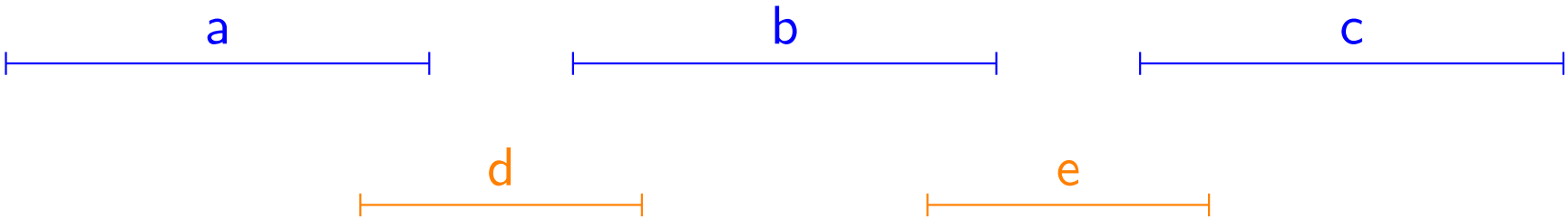
É útil pensar em contra-exemplos para cada um desses casos.

Contra-exemplo para intervalos que começam primeiro



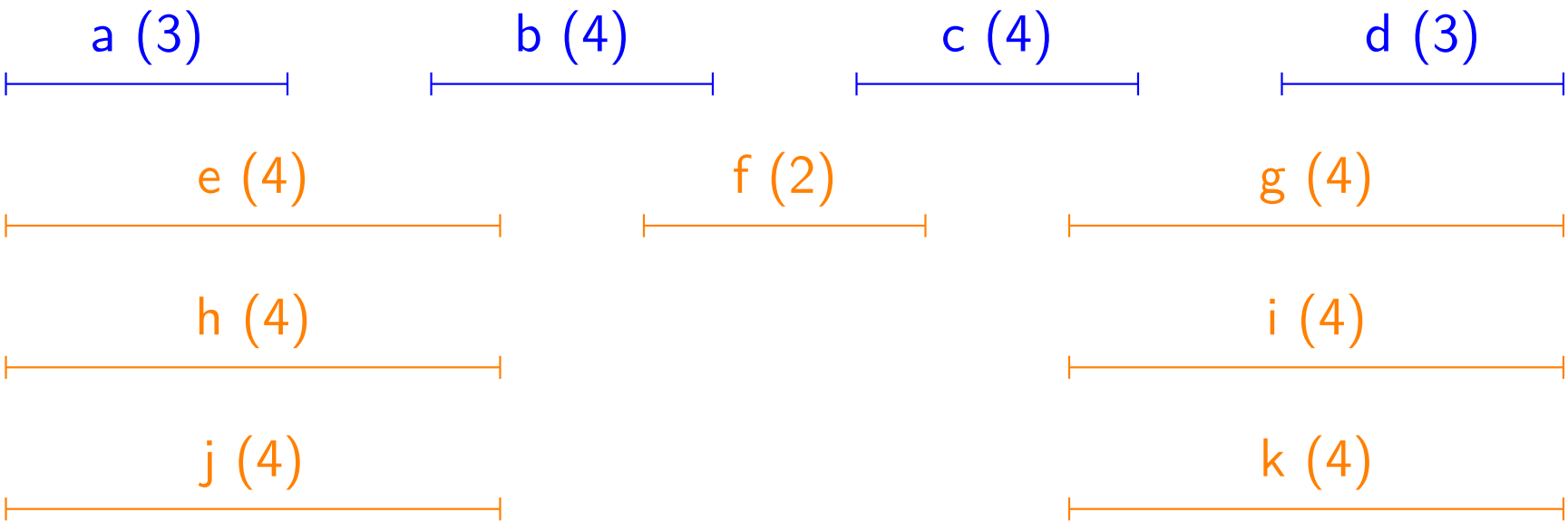
- Solução ótima: $\{a, b\}$
- Solução obtida: $\{c\}$

Contra-exemplo para menores intervalos



- Solução ótima: $\{a, b, c\}$
- Solução obtida: $\{d, e\}$

Contra-exemplo para intervalos com menos intersecções



- Solução ótima: $\{a, b, c, d\}$
- Solução obtida: $\{f, a, d\}$

Provando que intervalos que terminam primeiro funciona

- Vamos comparar o conjunto S selecionado pelo algoritmo ganancioso com o conjunto $S^\star$, a solução ótima.
- Queremos mostrar que $S = S^\star$.
- Provaremos que o algoritmo ganancioso sempre "fica na frente" por indução.