PAA - Aula 12
Seleção de Atividades e Mochila Fracionária

Roteiro para projetar um algoritmo guloso:

- Compreender o problema
- Propor estratégias/critérios gulosos
- Resolver exemplos usando essas estratégias
- Selecionar a estratégia mais promissora
- Escrever o pseudocódigo do algoritmo
- Analisar a eficiência do algoritmo
- Provar a corretude do critério guloso

Problema da Seleção de Atividades

No problema da seleção de atividades temos:

- um conjunto S com n atividades, i.e., $S = \{a_1, a_2, a_3, \ldots, a_n\}$
- e cada atividade a_i tem
  - tempo de início s_i e tempo de término t_i, com $0 \leq s_i < t_i$

$s_i, t_i \in \mathbb{Z}^+$

A motivação para esse problema é que

- as atividades precisam de um mesmo recurso
  - e temos que decidir quais atividades serão escolhidas.
- Imagine selecionar apresentações para assistir em um evento.

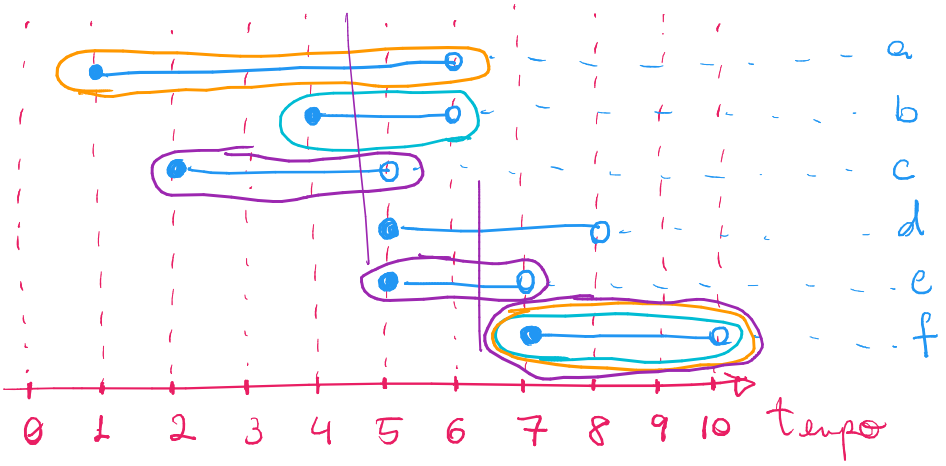
O objetivo é encontrar um subconjunto A de atividades com cardinalidade máxima

- tal que nenhum par de atividades interfira com outro.

Duas atividades a_i, a_j interferem se uma começa enquanto a outra acontece,

- i.e., se $s_i \leq s_j < t_i$ ou $s_j \leq s_i < t_j$

Exemplo:



Ideias de estratégias gulosas visando evitar interferências/conflitos:

i. Pegar o menor, $A = \{b, f\}$ $|A| = 2$
   - a ideia é que atividades pequenas interferem menos com as outras.

ii. Pegar o que começa mais cedo, $A = \{a, f\}$ $|A| = 2$
   - a ideia é pegar a próxima atividade pra qual estou disponível.

iii. Pegar o que termina mais cedo, $A = \{c, e, f\}$ $|A| = 3$
   - a ideia é deixar o máximo de tempo para alocar atividades vindouras.

↳ nosso critério candidato

Pseudocódigo do algoritmo:

```
SelecAtividades(S):
    Ordenar atividades em S em ordem crescente de tempo de término   O(n lg n)
    (renomear as atividades de 1 a n segundo essa ordem)
    t = 0        A = ∅
    para i = 1 até n:                                                 O(n)
        se s_i ≥ t:        → se a_i não interfere c/ as atividades selecionadas
            A = A ∪ {a_i}
            t = t_i
    devolva A
```

Eficiência: a ordenação inicial leva tempo $O(n \lg n)$

- e o laço principal leva tempo $O(n)$

- Assim, o tempo total é $O(n \lg n)$
  - sendo n o número de atividades em S, i.e., $n = |S|$

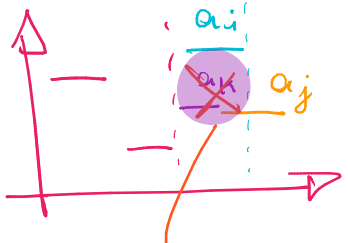
Prova de corretude para o critério guloso:

- vamos usar um argumento de troca e a prova será por contradição.

A = sol. obtida pelo alg. guloso

Por absurdo, A^* = sol. ótima melhor que A, i.e., $|A^*| > |A|$, sendo que A^* tem o maior # de atividades iguais a A. ⇐

Seja a_i a atividade que está em $A \setminus A^*$ c/ menor tempo de término. Pela análise ao lado, sabemos que a_i interfere c/ no máximo um item a_j em A^*.



$a_i \in A$

~~$a_i \in A^*$~~

$a_j \in A^*$

a_k não pode existir, pois a_i é o próximo item c/ menor tempo de término

Assim, $A' = (A^* \setminus \{a_j\}) \cup \{a_i\}$ é uma solução viável c/ $|A'| = |A^*|$ sendo que A' é mais parecida c/ A que A^* (absurdo).

Concluímos a prova de que A é ótima.

Problema da Mochila Fracionária

Na entrada do problema da mochila fracionária recebemos:

- Um conjunto de itens S sendo que
  - cada item i tem peso $p_i > 0$ e valor $v_i > 0$
- Também recebemos uma capacidade $C \in \mathbb{Z}^+$

Uma solução para o problema é:

- Uma coleção de frações de itens
  - cujo peso total não ultrapassa C
  - e cujo valor total seja máximo.

De modo compacto (usando programação linear) temos a definição:

$$\text{f.o.} \quad \max \sum_{i \in S} v_i x_i$$

$$\text{s.a.} \quad \sum_{i \in S} p_i x_i \leq C$$

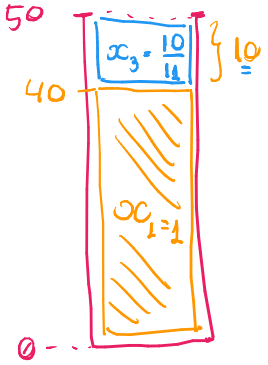
$$0 \leq x_i \leq 1 \quad \forall i \in S$$

Exemplo: $S = \{1,2,3\}$ e $C = 50$

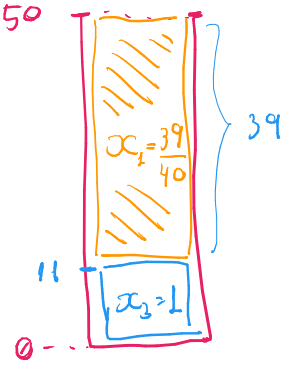
1: $p_1 = 40$ $v_1 = 200$

2: $p_2 = 20$ $v_2 = 20$

3: $p_3 = 11$ $v_3 = 110$



$200 \cdot 1 + 110 \cdot \frac{10}{11} = 300$



$200 \cdot \frac{39}{40} + 110 \cdot 1 = 305$

Definindo a estratégia gulosa:

- Se todos os itens tiverem o mesmo peso, os mais valiosos seriam preferidos,
  - pois acrescentam maior valor quando escolhidos.
- Se todos os itens tiverem o mesmo valor, os mais leves teriam prioridade,
  - pois ocupam menos capacidade quando escolhidos.
- Combinando essas ideias, a função de densidade $f(i) = v_i / p_i$
  - é uma boa candidata para descrever a prioridade de cada item.
- Nosso algoritmo guloso pode, a cada passo, pegar "pedaços"
  - do item ainda não escolhido que tem maior densidade de valor.

Pseudocódigo:

```
mochilaFrac(S, p, v, C):
    ordenar os itens em S de modo decrescente de densidade f(i) = v_i / p_i      O(n lg n)
    (renomear os itens de 1 a n segundo essa ordem)
    C' = C                                  # C' é a capacidade residual
    para i = 1 até n:                       O(n)
        se p_i ≤ C':
            x_i = 1        C' = C' - p_i
        senão
            x_i = C' / p_i   ⟺   x_i . p_i = C'
            C' = 0
    devolva x
```

Eficiência: a ordenação inicial leva tempo $O(n \lg n)$

- e o laço principal leva tempo $O(n)$

- Assim, o tempo total é $O(n \lg n)$
  - sendo n o número de itens em S, i.e., $|S|$

Prova de corretude para o critério guloso:

- vamos usar um argumento de troca e a prova será por contradição.

$\Rightarrow x^*$ = sol. ótima melhor que a gulosa (ela não segue o critério guloso!)

$\exists$ um par i, j com $i < j$ t.q. i não está inteiro em x^* ($x^*_i < 1$)
e j tem ao menos um pedaço em x^* ($x^*_j > 0$)

Como $i < j \Rightarrow \frac{v_i}{p_i} > \frac{v_j}{p_j} \Rightarrow v_i p_j > v_j p_i \Rightarrow v_i p_j - v_j p_i > 0$

Vamos remover ϵ de j de x^* e adicionar ϵ' de i em x^*.

Note que, $\epsilon \cdot p_j = \epsilon' \cdot p_i$ p/ respeitar a capacidade C

Qual a variação no valor da f.o. $\left(\sum_{k \in S} v_k x^*_k\right)$?

$$\Delta = \epsilon' v_i - \epsilon v_j = \epsilon \frac{p_j}{p_i} \cdot v_i + \epsilon v_j = \epsilon \frac{p_j v_i}{p_i} - \epsilon \frac{p_i v_j}{p_i}$$

$$= \frac{\epsilon}{p_i}\left(p_j v_i - p_i v_j\right) > 0$$

Logo a sol. x^* aumenta de valor, Temos uma contradição c/ela ser ótima.