Construção e Análise de Algoritmos

aula 00: Perguntas e respostas

1. **A disciplina de CAnA é fácil ou difícil?**

   Bom, isso depende.

   A disciplina de CAnA pode ser fácil, ou ela pode ser difícil.

2. **O que torna a disciplina fácil, e o que torna ela difícil?**

   A disciplina se torna fácil se a gente tem uma certa habilidade bem desenvolvida.

   E ela se torna difícil se essa habilidade ainda não está bem desenvolvida.

3. **Que habilidade é essa?**

   A habilidade de escrever pseudo-códigos
   — (*depois que você já tem uma ideia na cabeça ...*)

4. **Porque?**

   Bom, porque a disciplina deveria se chamar Reconstrução e Análise de Algoritmos
   — (*RcAnA*)

   Porque a gente nunca encontra um algoritmo bom de primeira.

   E a gente precisa reconstruir um algoritmo porcaria para chegar no algoritmo bom.

5. **Faz sentido!**

   Mas, aonde que entra a habilidade de escrever pseudo-código?

   Bom, o ponto é que um algoritmo não é uma coisa — (*algo que já está pronto*)

   Um algoritmo é uma coisa que a gente faz — (*escrevendo pseudo-código ...*)

6. **De fato, algoritmo não dá em árvore.**

   E nem brota do chão ...

   Pois é.

   Se você encontra um algoritmo pronto por aí, ele só está lá porque foi alguém que fez — (*alguém que escreveu ...*)

   Só que, as ideias que ficam por trás do algoritmo não estão lá
   — (*aquilo que serviu de base para a pessoa escrever*)

7. **Sim!**

   E é por isso que é difícil entender aquele troço ...

   Pois é.

   É difícil de entender, é difícil de analisar, é difícil de reescrever.

8. **E daí, a disciplina de CAnA vai ficando difícil ...**

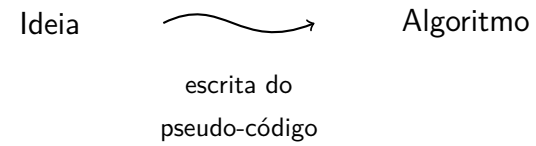
   Pois é.

   Mas não precisa ser assim.

9. **Mas, como é que fica fácil então?**

   Bom, fica fácil se você tem a habilidade de escrever pseudo-código.

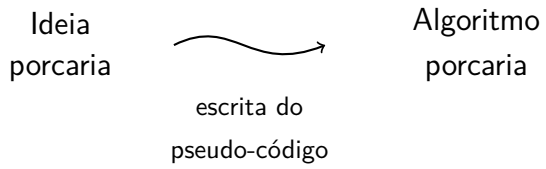
   Porque daí, você consegue sair da ideia para o algoritmo



10. **Mas, como é que eu vou ter uma boa ideia?**

    Bom, o ponto é que você não vai ter — (*ninguém tem ideia boa de primeira*)

    Então você começa com uma ideia porcaria, que te dá um algoritmo porcaria



11. **Mas isso é uma porcaria ...**

    Pois é.

    Mas o ponto é que você tem a ideia.

    E se você tem a ideia, você entende o que está acontecendo.

    E se você entende o que está acontecendo, você descobre onde está o problema.

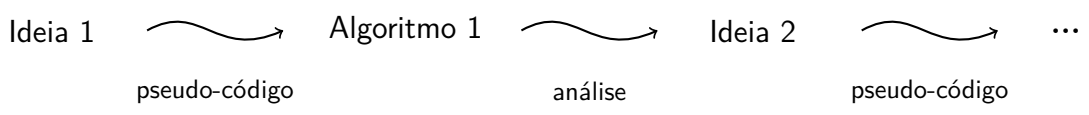
    E descobrindo isso, você vai acabar tendo outra ideia.

12. **Hmm ...**

    Mas aqui já tem outra habilidade, não?

    Sim, a habilidade da análise.

    E com a habilidade da análise, a figura fica completa



13. **Sim**!

    Reconstrução de algoritmos ...

Pois é.

É assim que a gente chega em um bom algoritmo

— (*as ideias vão aparecendo no meio do caminho ...*)

14. **Entendi.**

    Mas então, eu preciso de duas habilidades para fazer a coisa funcionar ...

    Então, a boa notícia é que não.

    Você só precisa da habilidade de escrever pseudo-código.

    Porque a habilidade de análise a gente vai desenvolver junto, ao longo da disciplina.

15. **Perfeito!**

    E como é que eu desenvolvo a habilidade de escrever pseudo-código?

    Bom, isso é uma habilidade prática.

    Então isso é uma coisa que a gente aprende a fazer fazendo.

    E para começar a praticar, você pode utilizar a lista de exercícios que está aí embaixo.

16. **Entendi.**

    Mas, e se eu não conseguir fazer algum?

    Ou tiver dificuldade logo no primeiro?

    Bom, isso não tem problema nenhum.

    Porque ninguém nasce sabendo as coisas ...

    A gente vai aprendendo as coisas ao longo da vida, com a ajuda das pessoas que já sabem.

    Então se você tiver dificuldades, basta procurar alguém que já sabe.

    E daí, a pessoa vai fazer o algoritmo pra você.

17. **Mas desse jeito, não vai ser eu que vou fazer.**

    E se eu não fazer, eu não vou aprender.

    Pois é.

    Essa é a segunda boa notícia.

    A gente também aprende habilidades práticas vendo as outras pessoas fazerem

    — (*a maior parte das coisas que a gente aprende na vida é assim*)

    Porque vendo alguém fazer, você percebe como se faz.

    E você também pode perguntar porque é que ela está fazendo daquele jeito

    — (*para entender a ideia que está por trás ...*)

    — (*porque atrás da ideia, sempre tem uma pessoa ...*)

18. **Perfeito!**

    Agora eu já entendi tudo.

    Muito obrigado 🙏

  Muito obrigado 🙏

lista de exercícios 00

1. O segundo maior elemento

   Imagine que você tem uma lista L contendo n números inteiros.

   Por exemplo,

   L:

9	42	21	14	28	3	19	32	46	6	15	7

   A tarefa consiste em

   → *Encontrar o segundo maior elemento da lista*

   No exemplo acima, isso nos daria

   => 42

   Apresente um algoritmo que realiza essa tarefa.

2. O k-ésimo maior elemento

   Imagine que você tem uma lista L contendo n números inteiros.

   Por exemplo,

   L:

9	42	21	14	28	3	19	32	46	6	15	7

   A tarefa consiste em

   → *Encontrar o k-ésimo maior elemento da lista*

   No exemplo acima, para k = 5 isso nos daria

   => 21

   Apresente um algoritmo que realiza essa tarefa.

   Você consegue fazer isso sem ordenar a lista?

3. Busca aproximada

   Imagine que você tem uma lista L contendo n números inteiros.

   Por exemplo,

   L:

9	42	21	14	25	3	19	33	45	6	15	7

   A tarefa consiste em, dado um número k

→ *Procurar o número* k *na lista* L*, e se ele não estiver lá*
retornar o elemento da lista com valor mais próximo de k

No exemplo acima, para k = 31, isso nos daria

```
=>  Não está lá, mas eu encontrei o 33
```

Apresente um algoritmo que realiza essa tarefa.

4. O dobro

Imagine que você tem uma lista L contendo n números inteiros.

Por exemplo,

L:

9	42	21	14	25	3	19	33	45	6	15	7

A tarefa consiste em

→ *Verificar se existem dois elementos na lista* L *tal que um deles é o dobro do outro*

No exemplo acima, isso nos daria

```
=>  Sim, os elementos 3 e 6
```

a) Apresente um algoritmo que resolve esse problema na lista não ordenada.

b) Apresente um algoritmo mais esperto que resolve esse problema na lista ordenada.

5. Duplicando os ímpares

Imagine que você tem uma lista ordenada L.

Por exemplo,

L:

1	6	7	8	15	16	19	21	25	33	43	48

A tarefa consiste em

→ *Duplicar todos os números da lista*

No exemplo acima, isso nos daria

L:

2	6	8	14	16	30	38	42	43	48	50	66

Apresente o algoritmo mais esperto que você conseguir para resolver esse problema.

6. Procedimento de intercalação

Imagine que você tem uma lista L contendo n números inteiros.

Imagine que a primeira metade de L está ordenada, e a segunda metade também.

Por exemplo,

L:

1	7	15	19	25	43	6	8	16	21	33	48

A tarefa consiste em

→ *Colocar a lista* L *completamente em ordem*

No exemplo acima, isso nos daria

L:	1	6	7	8	15	16	19	21	25	33	43	48

Apresente o algoritmo mais esperto que você conseguir para resolver esse problema.

7. Procedimento de partição

Imagine que você tem uma lista L contendo n números inteiros.

Por exemplo,

L:	9	42	21	14	25	3	19	33	45	6	15	7

A tarefa consiste em, dado um número k

→ *Colocar os números menores do que* k *no início da lista,*
colocar os números maiores do que k *no fim da lista,*
deixando o número k *entre eles*
— *(onde a ordem dos elementos não é importante)*

No exemplo acima, para k = 15, isso nos poderia nos dar

L:	9	14	3	6	7	15	42	21	25	19	33	45
						↑						

Apresente o algoritmo mais esperto que você conseguir para resolver esse problema.

8. Elementos repetidos

Imagine que você tem duas listas L1, L_2 contendo n números inteiros.

Por exemplo,

L_1:	9	42	21	14	25	3	19	33	45	6	15	7

L_2:	1	24	33	41	9	13	18	21	54	8	11	25

A tarefa consiste em

→ *Imprimir todos os elementos que aparecem nas duas listas*

No exemplo acima, isso nos daria

```
=>  33, 9, 21, 25
```

a) Apresente um algoritmo que resolve esse problema.

b) Apresente um algoritmo mais esperto que resolve esse problema quando L_1 e L_2 estão ordenadas.

10. Par-par, Ímpar-ímpar

Imagine que você tem uma lista L contendo $2n$ números inteiros.

E imagine que n elementos são pares, e n elementos são ímpares.

Por exemplo,

L:	9	42	21	14	25	3	28	33	44	6	15	8

A tarefa consiste em

→ *Colocar os elementos pares nas posições pares,*
e colocar os elementos ímpares nas posições ímpares.
— *(onde a ordem dos elementos não importa)*

No exemplo acima, isso nos daria

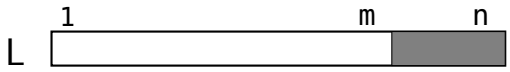
L:	42	9	14	21	28	25	44	3	6	33	8	15

— (*aqui nós estamos assumindo que a lista começa na posição 0*)

Apresente um algoritmo mais esperto que você conseguir para resolver esse problema
— (*sem utilizar uma lista auxiliar*)

11. Remoção de duplicatas

Imagine que você tem uma lista L com posições vazias no final



E imagine que essa lista pode conter elementos repetidos.

A tarefa consiste em

→ *Remover as cópias de cada elemento, de modo que no final*
cada elemento só aparece uma vez na lista

Apresente um algoritmo que realiza essa tarefa.

12. Agrupando as cópias

Imagine que você tem uma lista L que pode conter elementos repetidos.

A tarefa consiste em

→ *Reorganizar a lista de modo que todas as cópias de cada elemento*
apareçam lado a lado

Apresente um algoritmo que realiza essa tarefa.