Coding dojo

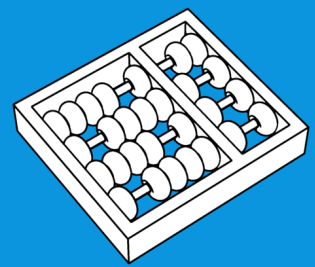
MC102 - Algoritmos e Programação de Computadores

06/25

26

Santiago Valdés Ravelo
https://ic.unicamp.br/~santiago/
ravelo@unicamp.br

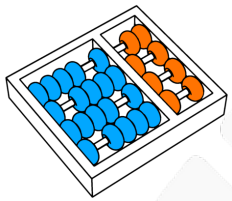
UNICAMP



“Se houver esforço, sempre haverá conquista.”

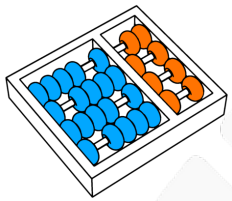
Jigoro Kano.

Dúvidas da aula anterior



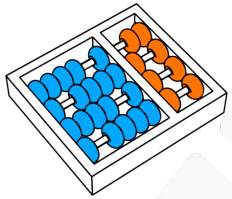
Dúvidas selecionadas

- Sei que python não tem intenção de priorizar performance, mas queria saber se tem um jeito de "reservar"memória pra alguma estrutura, se eu tenho uma noção de quantos elementos ela vai ter.



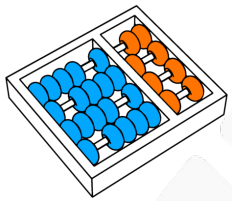
Dúvidas selecionadas

- Sei que python não tem intenção de priorizar performance, mas queria saber se tem um jeito de "reservar"memória pra alguma estrutura, se eu tenho uma noção de quantos elementos ela vai ter.
- Achei muito difícil e acho que não entendi como ocorre a parte lógica do algoritmo do cavalo do Xadrez.



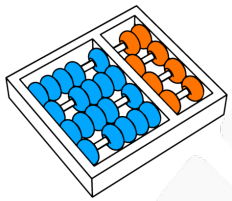
Dúvidas selecionadas

- Sei que python não tem intenção de priorizar performance, mas queria saber se tem um jeito de "reservar"memória pra alguma estrutura, se eu tenho uma noção de quantos elementos ela vai ter.
- Achei muito difícil e acho que não entendi como ocorre a parte lógica do algoritmo do cavalo do Xadrez.
- Nos exercícios da aula, foi criado uma função recursiva com mais parâmetros e outra função que chama a recursiva mas só com os parâmetros desejados. Por exemplo, a função sequencias(k, n) que tem os parâmetros que definem o problema k e n chama a função recursiva sequencias_rec(mem, k, n, usados), a qual possui parâmetros que são mais convenientes para resolver o problema recursivamente. Isso é uma prática comum ao resolver esses tipos de problemas?



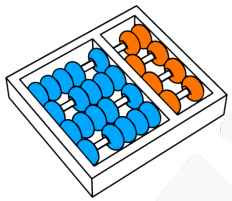
Dúvidas selecionadas

- Sei que python não tem intenção de priorizar performance, mas queria saber se tem um jeito de "reservar"memória pra alguma estrutura, se eu tenho uma noção de quantos elementos ela vai ter.
- Achei muito difícil e acho que não entendi como ocorre a parte lógica do algoritmo do cavalo do Xadrez.
- Nos exercícios da aula, foi criado uma função recursiva com mais parâmetros e outra função que chama a recursiva mas só com os parâmetros desejados. Por exemplo, a função sequencias(k, n) que tem os parâmetros que definem o problema k e n chama a função recursiva sequencias_rec(mem, k, n, usados), a qual possui parâmetros que são mais convenientes para resolver o problema recursivamente. Isso é uma prática comum ao resolver esses tipos de problemas?
- Os algoritmos de backtracking são, em geral, melhores que outros algoritmos quando se resolve um mesmo problema?



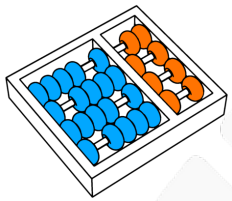
Dúvidas selecionadas

- Sei que python não tem intenção de priorizar performance, mas queria saber se tem um jeito de "reservar"memória pra alguma estrutura, se eu tenho uma noção de quantos elementos ela vai ter.
- Achei muito difícil e acho que não entendi como ocorre a parte lógica do algoritmo do cavalo do Xadrez.
- Nos exercícios da aula, foi criado uma função recursiva com mais parâmetros e outra função que chama a recursiva mas só com os parâmetros desejados. Por exemplo, a função sequencias(k, n) que tem os parâmetros que definem o problema k e n chama a função recursiva sequencias_rec(mem, k, n, usados), a qual possui parâmetros que são mais convenientes para resolver o problema recursivamente. Isso é uma prática comum ao resolver esses tipos de problemas?
- Os algoritmos de backtracking são, em geral, melhores que outros algoritmos quando se resolve um mesmo problema?
- não entendi direito a interação que acontece quando uma função recursiva é chamada dentro de um loop.



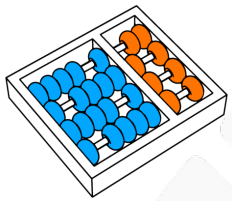
Dúvidas selecionadas

- Sei que python não tem intenção de priorizar performance, mas queria saber se tem um jeito de "reservar"memória pra alguma estrutura, se eu tenho uma noção de quantos elementos ela vai ter.
- Achei muito difícil e acho que não entendi como ocorre a parte lógica do algoritmo do cavalo do Xadrez.
- Nos exercícios da aula, foi criado uma função recursiva com mais parâmetros e outra função que chama a recursiva mas só com os parâmetros desejados. Por exemplo, a função sequencias(k, n) que tem os parâmetros que definem o problema k e n chama a função recursiva sequencias_rec(mem, k, n, usados), a qual possui parâmetros que são mais convenientes para resolver o problema recursivamente. Isso é uma prática comum ao resolver esses tipos de problemas?
- Os algoritmos de backtracking são, em geral, melhores que outros algoritmos quando se resolve um mesmo problema?
- não entendi direito a interação que acontece quando uma função recursiva é chamada dentro de um loop.
- fiquei confuso com a recursão da torre, não entendi como funciona a passagem de uma peça para outra torre.



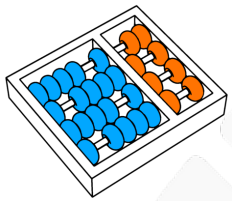
Dúvidas selecionadas

- Sei que python não tem intenção de priorizar performance, mas queria saber se tem um jeito de "reservar"memória pra alguma estrutura, se eu tenho uma noção de quantos elementos ela vai ter.
- Achei muito difícil e acho que não entendi como ocorre a parte lógica do algoritmo do cavalo do Xadrez.
- Nos exercícios da aula, foi criado uma função recursiva com mais parâmetros e outra função que chama a recursiva mas só com os parâmetros desejados. Por exemplo, a função sequencias(k, n) que tem os parâmetros que definem o problema k e n chama a função recursiva sequencias_rec(mem, k, n, usados), a qual possui parâmetros que são mais convenientes para resolver o problema recursivamente. Isso é uma prática comum ao resolver esses tipos de problemas?
- Os algoritmos de backtracking são, em geral, melhores que outros algoritmos quando se resolve um mesmo problema?
- não entendi direito a interação que acontece quando uma função recursiva é chamada dentro de um loop.
- fiquei confuso com a recursão da torre, não entendi como funciona a passagem de uma peça para outra torre.
- É possível que um backtracking entre em um loop caso ele não encontre uma solução geral?



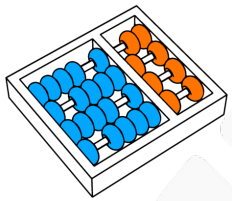
Dúvidas selecionadas

- Sei que python não tem intenção de priorizar performance, mas queria saber se tem um jeito de "reservar"memória pra alguma estrutura, se eu tenho uma noção de quantos elementos ela vai ter.
- Achei muito difícil e acho que não entendi como ocorre a parte lógica do algoritmo do cavalo do Xadrez.
- Nos exercícios da aula, foi criado uma função recursiva com mais parâmetros e outra função que chama a recursiva mas só com os parâmetros desejados. Por exemplo, a função sequencias(k, n) que tem os parâmetros que definem o problema k e n chama a função recursiva sequencias_rec(mem, k, n, usados), a qual possui parâmetros que são mais convenientes para resolver o problema recursivamente. Isso é uma prática comum ao resolver esses tipos de problemas?
- Os algoritmos de backtracking são, em geral, melhores que outros algoritmos quando se resolve um mesmo problema?
- não entendi direito a interação que acontece quando uma função recursiva é chamada dentro de um loop.
- fiquei confuso com a recursão da torre, não entendi como funciona a passagem de uma peça para outra torre.
- É possível que um backtracking entre em um loop caso ele não encontre uma solução geral?
- Esses algoritmos q estamos vendo são voltados para melhorar nossa lógica ou vamos usá-los na prática?



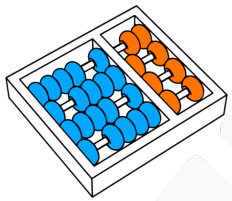
Dúvidas selecionadas

- Sei que python não tem intenção de priorizar performance, mas queria saber se tem um jeito de "reservar"memória pra alguma estrutura, se eu tenho uma noção de quantos elementos ela vai ter.
- Achei muito difícil e acho que não entendi como ocorre a parte lógica do algoritmo do cavalo do Xadrez.
- Nos exercícios da aula, foi criado uma função recursiva com mais parâmetros e outra função que chama a recursiva mas só com os parâmetros desejados. Por exemplo, a função sequencias(k, n) que tem os parâmetros que definem o problema k e n chama a função recursiva sequencias_rec(mem, k, n, usados), a qual possui parâmetros que são mais convenientes para resolver o problema recursivamente. Isso é uma prática comum ao resolver esses tipos de problemas?
- Os algoritmos de backtracking são, em geral, melhores que outros algoritmos quando se resolve um mesmo problema?
- não entendi direito a interação que acontece quando uma função recursiva é chamada dentro de um loop.
- fiquei confuso com a recursão da torre, não entendi como funciona a passagem de uma peça para outra torre.
- É possível que um backtracking entre em um loop caso ele não encontre uma solução geral?
- Esses algoritmos q estamos vendo são voltados para melhorar nossa lógica ou vamos usá-los na prática?
- Não entendi direito como gerar as sequencias possíveis de 3 números.



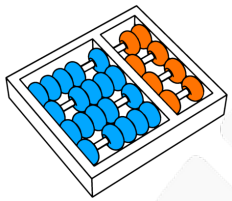
Dúvidas selecionadas

- Sei que python não tem intenção de priorizar performance, mas queria saber se tem um jeito de "reservar"memória pra alguma estrutura, se eu tenho uma noção de quantos elementos ela vai ter.
- Achei muito difícil e acho que não entendi como ocorre a parte lógica do algoritmo do cavalo do Xadrez.
- Nos exercícios da aula, foi criado uma função recursiva com mais parâmetros e outra função que chama a recursiva mas só com os parâmetros desejados. Por exemplo, a função sequencias(k, n) que tem os parâmetros que definem o problema k e n chama a função recursiva sequencias_rec(mem, k, n, usados), a qual possui parâmetros que são mais convenientes para resolver o problema recursivamente. Isso é uma prática comum ao resolver esses tipos de problemas?
- Os algoritmos de backtracking são, em geral, melhores que outros algoritmos quando se resolve um mesmo problema?
- não entendi direito a interação que acontece quando uma função recursiva é chamada dentro de um loop.
- fiquei confuso com a recursão da torre, não entendi como funciona a passagem de uma peça para outra torre.
- É possível que um backtracking entre em um loop caso ele não encontre uma solução geral?
- Esses algoritmos q estamos vendo são voltados para melhorar nossa lógica ou vamos usá-los na prática?
- Não entendi direito como gerar as sequencias possíveis de 3 números.
- Como faço para enxergar uma solução recursiva?



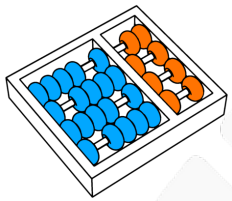
Dúvidas selecionadas

- Sei que python não tem intenção de priorizar performance, mas queria saber se tem um jeito de "reservar"memória pra alguma estrutura, se eu tenho uma noção de quantos elementos ela vai ter.
- Achei muito difícil e acho que não entendi como ocorre a parte lógica do algoritmo do cavalo do Xadrez.
- Nos exercícios da aula, foi criado uma função recursiva com mais parâmetros e outra função que chama a recursiva mas só com os parâmetros desejados. Por exemplo, a função sequencias(k, n) que tem os parâmetros que definem o problema k e n chama a função recursiva sequencias_rec(mem, k, n, usados), a qual possui parâmetros que são mais convenientes para resolver o problema recursivamente. Isso é uma prática comum ao resolver esses tipos de problemas?
- Os algoritmos de backtracking são, em geral, melhores que outros algoritmos quando se resolve um mesmo problema?
- não entendi direito a interação que acontece quando uma função recursiva é chamada dentro de um loop.
- fiquei confuso com a recursão da torre, não entendi como funciona a passagem de uma peça para outra torre.
- É possível que um backtracking entre em um loop caso ele não encontre uma solução geral?
- Esses algoritmos q estamos vendo são voltados para melhorar nossa lógica ou vamos usá-los na prática?
- Não entendi direito como gerar as sequencias possíveis de 3 números.
- Como faço para enxergar uma solução recursiva?
- Não entendi direito o algoritmo dos parênteses.



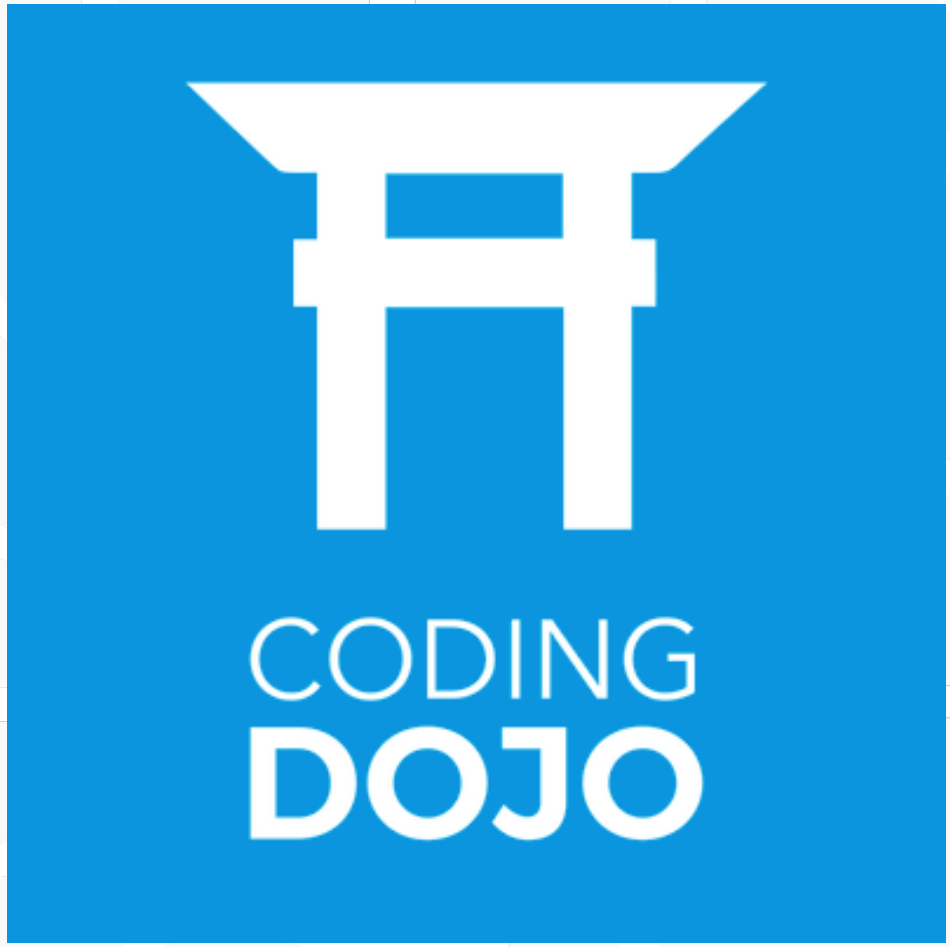
Dúvidas selecionadas

- Sei que python não tem intenção de priorizar performance, mas queria saber se tem um jeito de "reservar"memória pra alguma estrutura, se eu tenho uma noção de quantos elementos ela vai ter.
- Achei muito difícil e acho que não entendi como ocorre a parte lógica do algoritmo do cavalo do Xadrez.
- Nos exercícios da aula, foi criado uma função recursiva com mais parâmetros e outra função que chama a recursiva mas só com os parâmetros desejados. Por exemplo, a função sequencias(k, n) que tem os parâmetros que definem o problema k e n chama a função recursiva sequencias_rec(mem, k, n, usados), a qual possui parâmetros que são mais convenientes para resolver o problema recursivamente. Isso é uma prática comum ao resolver esses tipos de problemas?
- Os algoritmos de backtracking são, em geral, melhores que outros algoritmos quando se resolve um mesmo problema?
- não entendi direito a interação que acontece quando uma função recursiva é chamada dentro de um loop.
- fiquei confuso com a recursão da torre, não entendi como funciona a passagem de uma peça para outra torre.
- É possível que um backtracking entre em um loop caso ele não encontre uma solução geral?
- Esses algoritmos q estamos vendo são voltados para melhorar nossa lógica ou vamos usá-los na prática?
- Não entendi direito como gerar as sequencias possíveis de 3 números.
- Como faço para enxergar uma solução recursiva?
- Não entendi direito o algoritmo dos parênteses.
- Não seria igualmente eficiente criar um While Loop no lugar de uma recursão?

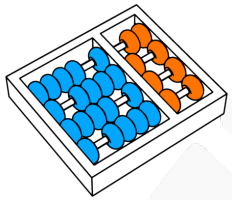


Dúvidas selecionadas

- Sei que python não tem intenção de priorizar performance, mas queria saber se tem um jeito de "reservar"memória pra alguma estrutura, se eu tenho uma noção de quantos elementos ela vai ter.
- Achei muito difícil e acho que não entendi como ocorre a parte lógica do algoritmo do cavalo do Xadrez.
- Nos exercícios da aula, foi criado uma função recursiva com mais parâmetros e outra função que chama a recursiva mas só com os parâmetros desejados. Por exemplo, a função sequencias(k, n) que tem os parâmetros que definem o problema k e n chama a função recursiva sequencias_rec(mem, k, n, usados), a qual possui parâmetros que são mais convenientes para resolver o problema recursivamente. Isso é uma prática comum ao resolver esses tipos de problemas?
- Os algoritmos de backtracking são, em geral, melhores que outros algoritmos quando se resolve um mesmo problema?
- não entendi direito a interação que acontece quando uma função recursiva é chamada dentro de um loop.
- fiquei confuso com a recursão da torre, não entendi como funciona a passagem de uma peça para outra torre.
- É possível que um backtracking entre em um loop caso ele não encontre uma solução geral?
- Esses algoritmos q estamos vendo são voltados para melhorar nossa lógica ou vamos usá-los na prática?
- Não entendi direito como gerar as sequencias possíveis de 3 números.
- Como faço para enxergar uma solução recursiva?
- Não entendi direito o algoritmo dos parênteses.
- Não seria igualmente eficiente criar um While Loop no lugar de uma recursão?
- Log n é o melhor algoritmo possível?

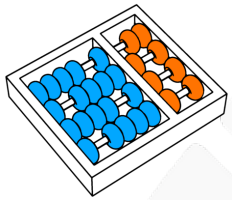

CODING
DOJO

Lembrando o funcionamento do Coding Dojo



Coding Dojo

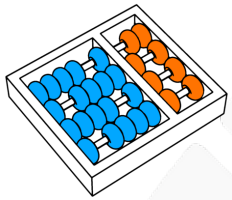
Inspirado nas práticas dos dojos de artes marciais japonesas, um Coding Dojo pode ter vários estilos.



Coding Dojo

Inspirado nas práticas dos dojos de artes marciais japonesas, um Coding Dojo pode ter vários estilos.

O formato que usaremos será o: Randori Kata.

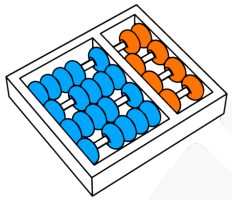


Coding Dojo

Inspirado nas práticas dos dojos de artes marciais japonesas, um Coding Dojo pode ter vários estilos.

O formato que usaremos será o: Randori Kata.

Componentes:



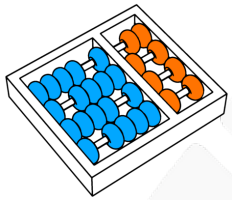
Coding Dojo

Inspirado nas práticas dos dojos de artes marciais japonesas, um Coding Dojo pode ter vários estilos.

O formato que usaremos será o: Randori Kata.

Componentes:

- Um computador.



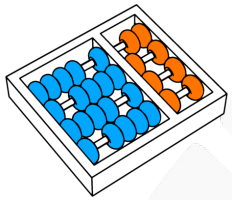
Coding Dojo

Inspirado nas práticas dos dojos de artes marciais japonesas, um Coding Dojo pode ter vários estilos.

O formato que usaremos será o: Randori Kata.

Componentes:

- Um computador.
- Um projetor.



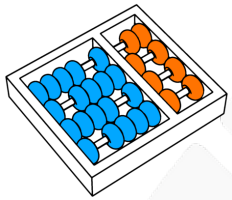
Coding Dojo

Inspirado nas práticas dos dojos de artes marciais japonesas, um Coding Dojo pode ter vários estilos.

O formato que usaremos será o: Randori Kata.

Componentes:

- Um computador.
- Um projetor.
- Um mestre



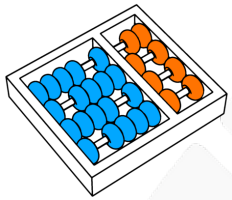
Coding Dojo

Inspirado nas práticas dos dojos de artes marciais japonesas, um Coding Dojo pode ter vários estilos.

O formato que usaremos será o: Randori Kata.

Componentes:

- Um computador.
- Um projetor.
- Um mestre (o tal de Santiago?).



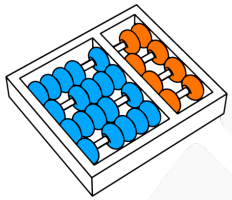
Coding Dojo

Inspirado nas práticas dos dojos de artes marciais japonesas, um Coding Dojo pode ter vários estilos.

O formato que usaremos será o: Randori Kata.

Componentes:

- Um computador.
- Um projetor.
- Um mestre (o tal de Santiago?).
- Um piloto



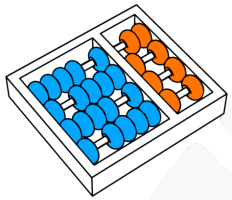
Coding Dojo

Inspirado nas práticas dos dojos de artes marciais japonesas, um Coding Dojo pode ter vários estilos.

O formato que usaremos será o: Randori Kata.

Componentes:

- Um computador.
- Um projetor.
- Um mestre (o tal de Santiago?).
- Um piloto (aluno voluntário).



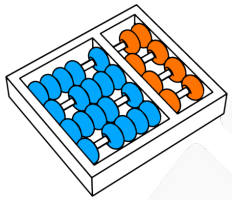
Coding Dojo

Inspirado nas práticas dos dojos de artes marciais japonesas, um Coding Dojo pode ter vários estilos.

O formato que usaremos será o: Randori Kata.

Componentes:

- Um computador.
- Um projetor.
- Um mestre (o tal de Santiago?).
- Um piloto (aluno voluntário).
- Um copiloto



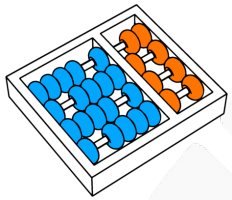
Coding Dojo

Inspirado nas práticas dos dojos de artes marciais japonesas, um Coding Dojo pode ter vários estilos.

O formato que usaremos será o: Randori Kata.

Componentes:

- Um computador.
- Um projetor.
- Um mestre (o tal de Santiago?).
- Um piloto (aluno voluntário).
- Um copiloto (aluno voluntário).



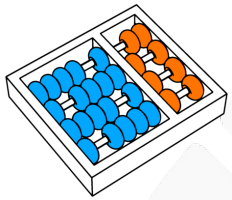
Coding Dojo

Inspirado nas práticas dos dojos de artes marciais japonesas, um Coding Dojo pode ter vários estilos.

O formato que usaremos será o: Randori Kata.

Componentes:

- Um computador.
- Um projetor.
- Um mestre (o tal de Santiago?).
- Um piloto (aluno voluntário).
- Um copiloto (aluno voluntário).
- Plateia



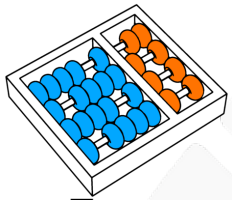
Coding Dojo

Inspirado nas práticas dos dojos de artes marciais japonesas, um Coding Dojo pode ter vários estilos.

O formato que usaremos será o: Randori Kata.

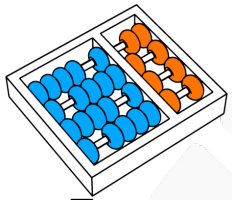
Componentes:

- Um computador.
- Um projetor.
- Um mestre (o tal de Santiago?).
- Um piloto (aluno voluntário).
- Um copiloto (aluno voluntário).
- Plateia (restante da turma).



Coding dojo - Randori Kata

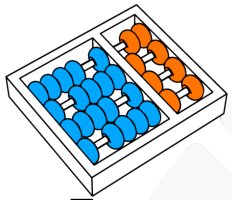
Funcionamento:



Coding dojo - Randori Kata

Funcionamento:

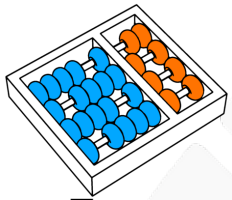
- O mestre propõe um problema.



Coding dojo - Randori Kata

Funcionamento:

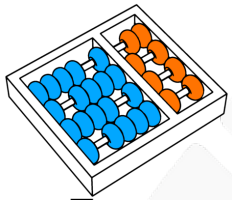
- O mestre propõe um problema.
- O piloto e o copiloto tentam solucioná-lo durante 5 minutos:



Coding dojo - Randori Kata

Funcionamento:

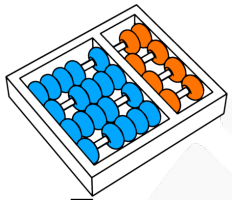
- O mestre propõe um problema.
- O piloto e o copiloto tentam solucioná-lo durante 5 minutos:
  - Piloto e copiloto devem explicar a ideia de solução para a plateia.



Coding dojo - Randori Kata

Funcionamento:

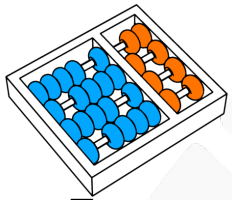
- O mestre propõe um problema.
- O piloto e o copiloto tentam solucioná-lo durante 5 minutos:
  - Piloto e copiloto devem explicar a ideia de solução para a plateia.
  - Só o piloto pode programar.



Coding dojo - Randori Kata

Funcionamento:

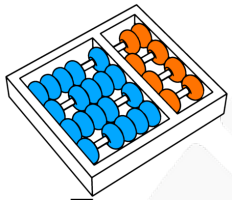
- O mestre propõe um problema.
- O piloto e o copiloto tentam solucioná-lo durante 5 minutos:
  - Piloto e copiloto devem explicar a ideia de solução para a plateia.
  - Só o piloto pode programar.
  - O copiloto pode apontar erros e dar sugestões.



Coding dojo - Randori Kata

Funcionamento:

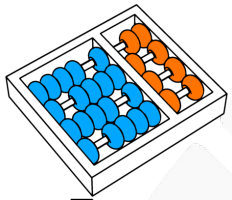
- O mestre propõe um problema.
- O piloto e o copiloto tentam solucioná-lo durante 5 minutos:
  - Piloto e copiloto devem explicar a ideia de solução para a plateia.
  - Só o piloto pode programar.
  - O copiloto pode apontar erros e dar sugestões.
  - A plateia só pode participar se o piloto ou o copiloto pedem ajuda.



Coding dojo - Randori Kata

Funcionamento:

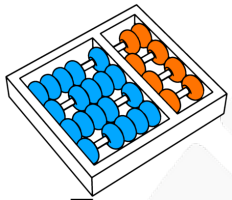
- O mestre propõe um problema.
- O piloto e o copiloto tentam solucioná-lo durante 5 minutos:
  - Piloto e copiloto devem explicar a ideia de solução para a plateia.
  - Só o piloto pode programar.
  - O copiloto pode apontar erros e dar sugestões.
  - A plateia só pode participar se o piloto ou o copiloto pedem ajuda.
  - Em caso de necessidade, podem perguntar ao mestre, mas ele responde com outra pergunta.



Coding dojo - Randori Kata

Funcionamento:

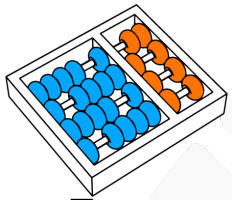
- O mestre propõe um problema.
- O piloto e o copiloto tentam solucioná-lo durante 5 minutos:
  - Piloto e copiloto devem explicar a ideia de solução para a plateia.
  - Só o piloto pode programar.
  - O copiloto pode apontar erros e dar sugestões.
  - A plateia só pode participar se o piloto ou o copiloto pedem ajuda.
  - Em caso de necessidade, podem perguntar ao mestre, mas ele responde com outra pergunta.
- Passados os 5 minutos, o desafio pausa (mesmo não sendo solucionado):



Coding dojo - Randori Kata

Funcionamento:

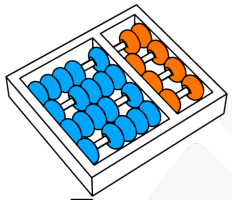
- O mestre propõe um problema.
- O piloto e o copiloto tentam solucioná-lo durante 5 minutos:
  - Piloto e copiloto devem explicar a ideia de solução para a plateia.
  - Só o piloto pode programar.
  - O copiloto pode apontar erros e dar sugestões.
  - A plateia só pode participar se o piloto ou o copiloto pedem ajuda.
  - Em caso de necessidade, podem perguntar ao mestre, mas ele responde com outra pergunta.
- Passados os 5 minutos, o desafio pausa (mesmo não sendo solucionado):
  - O piloto volta para a plateia.



Coding dojo - Randori Kata

Funcionamento:

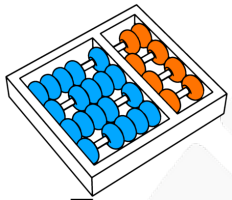
- O mestre propõe um problema.
- O piloto e o copiloto tentam solucioná-lo durante 5 minutos:
  - Piloto e copiloto devem explicar a ideia de solução para a plateia.
  - Só o piloto pode programar.
  - O copiloto pode apontar erros e dar sugestões.
  - A plateia só pode participar se o piloto ou o copiloto pedem ajuda.
  - Em caso de necessidade, podem perguntar ao mestre, mas ele responde com outra pergunta.
- Passados os 5 minutos, o desafio pausa (mesmo não sendo solucionado):
  - O piloto volta para a plateia.
  - O copiloto se torna piloto.



Coding dojo - Randori Kata

Funcionamento:

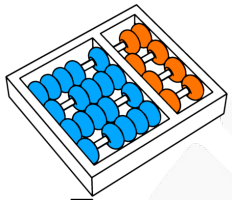
- O mestre propõe um problema.
- O piloto e o copiloto tentam solucioná-lo durante 5 minutos:
  - Piloto e copiloto devem explicar a ideia de solução para a plateia.
  - Só o piloto pode programar.
  - O copiloto pode apontar erros e dar sugestões.
  - A plateia só pode participar se o piloto ou o copiloto pedem ajuda.
  - Em caso de necessidade, podem perguntar ao mestre, mas ele responde com outra pergunta.
- Passados os 5 minutos, o desafio pausa (mesmo não sendo solucionado):
  - O piloto volta para a plateia.
  - O copiloto se torna piloto.
  - Um novo membro da plateia se torna copiloto.



Coding dojo - Randori Kata

Funcionamento:

- O mestre propõe um problema.
- O piloto e o copiloto tentam solucioná-lo durante 5 minutos:
  - Piloto e copiloto devem explicar a ideia de solução para a plateia.
  - Só o piloto pode programar.
  - O copiloto pode apontar erros e dar sugestões.
  - A plateia só pode participar se o piloto ou o copiloto pedem ajuda.
  - Em caso de necessidade, podem perguntar ao mestre, mas ele responde com outra pergunta.
- Passados os 5 minutos, o desafio pausa (mesmo não sendo solucionado):
  - O piloto volta para a plateia.
  - O copiloto se torna piloto.
  - Um novo membro da plateia se torna copiloto.
- A solução do desafio continua com os novos piloto e copiloto.



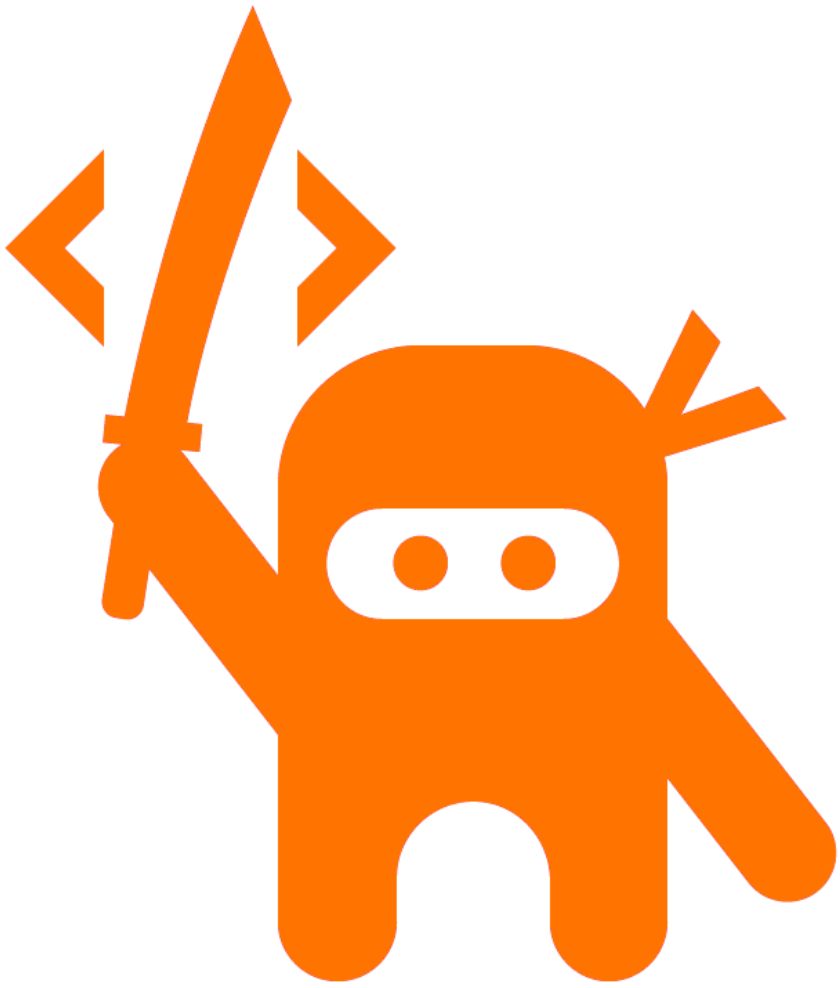
Coding dojo - Randori Kata

Funcionamento:

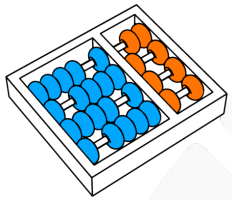
- O mestre propõe um problema.
- O piloto e o copiloto tentam solucioná-lo durante 5 minutos:
  - Piloto e copiloto devem explicar a ideia de solução para a plateia.
  - Só o piloto pode programar.
  - O copiloto pode apontar erros e dar sugestões.
  - A plateia só pode participar se o piloto ou o copiloto pedem ajuda.
  - Em caso de necessidade, podem perguntar ao mestre, mas ele responde com outra pergunta.
- Passados os 5 minutos, o desafio pausa (mesmo não sendo solucionado):
  - O piloto volta para a plateia.
  - O copiloto se torna piloto.
  - Um novo membro da plateia se torna copiloto.
- A solução do desafio continua com os novos piloto e copiloto.
- Se o desafio for concluído, o mestre lança um novo desafio.

Coding dojo

Prontos para começar?

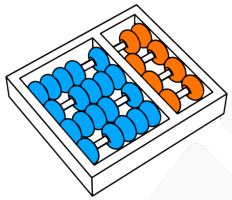


CODE
DOJO



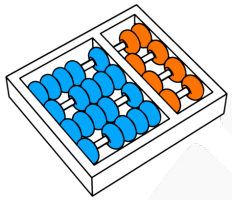
Palíndromo

Faça uma função recursiva que dada uma string, determina se é um palíndromo ou não.



Produto

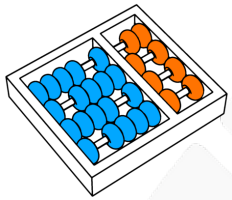
Dados dois inteiros a e b, faça uma função recursiva que calcule $a \cdot b$, sem usar multiplicação.



Potência

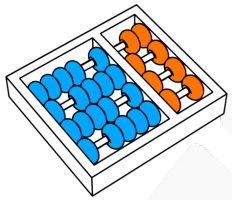
Dados a e n, faça uma função recursiva que calcule a^n.

- Consegue fazer com aproximadamente $\log_2 n$ multiplicações?



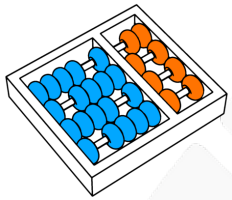
Busca binária

Faça uma função que implemente a busca binária.



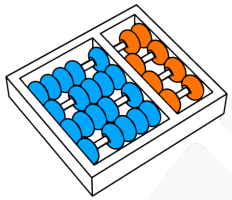
Eliminando duplicatas

Faça uma função recursiva que, dada uma string s, retorna uma string que representa s após remover todos os caracteres iguais consecutivos.



Anagramas

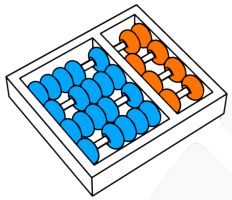
Faça uma função recursiva que imprima todos os anagramas de uma dada string.



Todos os caminhos

Dada uma matriz binária, um caminho do extremo superior esquerdo ao extremos inferior direito é uma sequência (sem repetição) de casas adjacentes (na horizontal ou vertical), todas com valor 1 (verdadeiro).

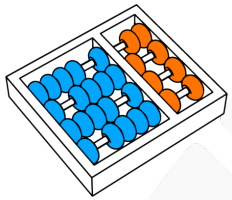
- Faça uma função recursiva que recebe uma matriz binária e imprime todos os caminhos válidos do extremos superior esquerdo ao inferior direito.



Colorindo

Dada uma matriz inteira, onde os números representam cores, casas adjacentes (na horizontal, vertical ou diagonal) com uma mesma cor formam uma região dessa cor.

- Faça uma função recursiva que recebe uma matriz inteira, uma posição (i, j) e um número, e modifica todas as casas da região que contém (i, j) para serem iguais ao número dado.



Maior palíndromo

Faça uma função recursiva que, dada uma string, imprima o tamanho do maior palíndromo que é substring dela.

Coding dojo

MC102 - Algoritmos e Programação de Computadores

06/25

26

Santiago Valdés Ravelo
https://ic.unicamp.br/~santiago/
ravelo@unicamp.br

UNICAMP