Recursão

Técnicas de Programação

Igor Montagner (igorsm1@insper.edu.br)

Recursão

algoritmo que chama a si mesmo uma ou mais vezes

passando uma instância menor do problema

Igor Montagner (igorsm1@insper.edu.br)

Recursão

- facilita muito a criação de alguns tipos de algoritmos
- técnica avançada difícil de entender
  - simular um algoritmo recursivo não é fácil
  - outras maneiras de pensar sobre corretude

Uma maneira melhor de pensar

1. Para quais entradas não precisamos fazer chamada recursiva?
2. O programa termina?
3. O programa funciona, supondo que as chamadas recursivas retornam o valor correto?

QuickSort

```
QUICKSORT(A, INI, FIM)

IF INI <= FIM - 1 THEN
    RETURN
END

LP = PARTICIONA(A, INI, FIM)
QUICKSORT(A, INI, LP)
QUICKSORT(A, LP+1, FIM)
```

Igor Montagner (igorsm1@insper.edu.br)

Uma maneira melhor de pensar

1. Para quais entradas não precisamos fazer chamada recursiva?
2. O programa termina?
3. O programa funciona, supondo que as chamadas recursivas retornam o valor correto?

Uma maneira melhor de pensar

1. Para quais entradas não precisamos fazer chamada recursiva?

Arrays de tamanho 0 (vazio) e 1. Ambos já estão ordenados

2. O programa termina?
3. O programa funciona, supondo que as chamadas recursivas retornam o valor correto?

Uma maneira melhor de pensar

1. Para quais entradas não precisamos fazer chamada recursiva?

Arrays de tamanho 0 (vazio) e 1. Ambos já estão ordenados

2. O programa termina?

As chamadas recursivas passam entradas de tamanho sempre menor. Logo, eventualmente temos entradas de tamanho 1 ou 0.

3. O programa funciona, supondo que as chamadas recursivas retornam o valor correto?

Uma maneira melhor de pensar

1. Para quais entradas não precisamos fazer chamada recursiva?

Arrays de tamanho 0 (vazio) e 1. Ambos já estão ordenados

2. O programa termina?

As chamadas recursivas passam entradas de tamanho sempre menor. Logo, eventualmente temos entradas de tamanho 1 ou 0.

3. O programa funciona, supondo que as chamadas recursivas retornam o valor correto?

Sim! O particiona coloca um elemento no lugar correto. Se o restante funciona então estará tudo ordenado!

Simulação vs Análise de Algoritmos

Para mostrar que um algoritmo **funciona**:

1. passos para uma **prova por indução**
2. simulação mostra que funciona para **um caso específico**
3. análise de algoritmos prova que funciona **para todos os casos**

Simulação vs Análise de Algoritmos

E para mostrar que **NÃO** funciona?

1. prova por **contra-exemplo**
2. simulação mostra que **NÃO** funciona para **um caso específico**
    - logo, não funciona :)

Ordena os dois lados e depois junta em um ordenado só

Merge sort

Igor Montagner (igorsm1@insper.edu.br)

Merge sort

1. divida o vetor em duas metades
2. ordene a metade da direita
3. ordene a metade da esquerda
4. combine as duas metades ordenadas tal que agora o vetor inteiro esteja ordenado

Merge sort

Combinar dois arrays ordenados em um só é o desafio

Igor Montagner (igorsm1@insper.edu.br)

Vamos fazer juntos

- A1 = [1 3 5 7]
- A2 = [2 4 6 8]

AUX =

Vamos fazer juntos (de novo)

- A1 = [1 3 5 7]
- A2 = [2 4 6 8]

AUX =

Vamos fazer juntos (última vez)

- A1 = [1 3 5 7]
- A2 = [2 4 6 8]

AUX =

Atividade 1 - o algoritmo `MERGE`

Igor Montagner (igorsm1@insper.edu.br)

MERGE (A, AUX, ini, fim, meio)

Igor Montagner (igorsm1@insper.edu.br)

O Merge Sort (recursivo)

1. Quais tamanhos de array não preciso chamar recursivo?

O Merge Sort (recursivo)

2. O algoritmo termina? Como é feita a divisão do array pela metade?

O Merge Sort (recursivo)

3. Se os dois lados forem ordenados corretamente, o inteiro vai ficar ordenado?

MERGE-SORT

```
MERGE-SORT(A, AUX, INI, FIM)

IF INI <= FIM - 1 THEN
    RETURN
END

MEIO <- (INI + FIM - 1) / 2
MERGE-SORT(A, INI, MEIO)
MERGE-SORT(A, MEIO, FIM)

MERGE(A, AUX, INI, FIM, MEIO)
```

Igor Montagner (igorsm1@insper.edu.br)

Atividade final - simulação do MERGE no PL

1. Implementações não caem na prova!
2. Mas, recursão cai.
3. Algoritmos `MERGE` e `PARTICIONA` também caem.

Igor Montagner (igorsm1@insper.edu.br)