Aviso sobre as Aulas de Sábado

Reposições

Era para ser:

Inicio: 7:20h
Término 10:00
Com aula na quinta-feira

Como será:

Inicio: 8:20h
Término 12:00
Sem aula na quinta-feira

NA/NC → 8:20h → 10:10h
NB → 10:10h → 12:00h

Estrutura de Dados – Revisão Recursão

Voncarlos Marcelo de Araújo

Recursividade

Recursividade

A ideia é que um problema pode ser resolvido da seguinte maneira:

Recursividade

A ideia é que um problema pode ser resolvido da seguinte maneira:

- **Primeiro, definimos as soluções para casos básicos**

Recursividade

A ideia é que um problema pode ser resolvido da seguinte maneira:

- **Primeiro, definimos as soluções para casos básicos**
- **Em seguida, tentamos reduzir o problema para instâncias menores do problema**

Recursividade

A ideia é que um problema pode ser resolvido da seguinte maneira:

- **Primeiro, definimos as soluções para casos básicos**
- **Em seguida, tentamos reduzir o problema para instâncias menores do problema**
- **Finalmente, combinamos o resultado das instâncias menores para obter um resultado do problema original**

Genericamente

Caso base:

- **resolve instâncias pequenas diretamente**

Caso geral:

- **reduz o problema para instâncias menores do mesmo problema**
- **chama a função recursivamente**

Genericamente

Caso base:

- **resolve instâncias pequenas diretamente**

Caso geral:

- **reduz o problema para instâncias menores do mesmo problema**
- **chama a função recursivamente**

```
1 int fat(int n) {
2   if (n == 0) /* caso base */
3     return 1;
4   else /* caso geral */
5     return n * fat(n-1); /* instância menor */
6 }
```

Definições recursivas

- **Algumas operações matemáticas ou objetos matemáticos têm uma definição recursiva**

Definições recursivas

- **Algumas operações matemáticas ou objetos matemáticos têm uma definição recursiva**
  - Ex: fatorial, sequência de Fibonacci, palíndromos, etc...
  - ou podem ser vistos do ponto de vista da recursão
    - multiplicação, divisão, exponenciação, etc..
- **Isso nos permite projetar algoritmos para lidar com essas operações/objetos**

Ex: Exponenciação

Definições recursivas

- **Isso nos permite projetar algoritmos para lidar com essas operações/objetos**

Ex: Exponenciação

Seja a é um número real e b é um número inteiro não-negativo

- Se $b = 0$, então $a^b = 1$
- Se $b > 0$, então $a^b = a \cdot a^{b-1}$

```
1 double potencia(double a, int b) {
2   if (b == 0)
3     return 1;
4   else
5     return a * potencia(a, b-1);
6 }
```

Busca Binária

Para buscar x no vetor ordenado dados entre as posições l e r

Casos base:

- **Se o intervalo for vazio (l > r), x não está no vetor**
- **Se dados[m] == x, onde m = (l + r)/2**
  - Devolvemos **m**

Caso geral:

- **Se dados[m] < x, então x só pode estar entre m + 1 e r**
  - **Devolvemos o resultado da chamada recursiva**
- **Se dados[m] > x, então x só pode estar entre l e m - 1**
  - **Devolvemos o resultado da chamada recursiva**

Busca Binária

Para buscar x no vetor ordenado dados entre as posições l e r

Casos base:

- **Se o intervalo for vazio (l > r), x não está no vetor**
- **Se dados[m] == x, onde m = (l + r)/2**
  - Devolvemos **m**

Caso geral:

- **Se dados[m] < x, então x só pode estar entre m + 1 e r**
- **Se dados[m] > x, então x só pode estar entre l e m - 1**
  - **Devolvemos o resultado da chamada recursiva**

```
1 int busca_binaria(int *dados, int l, int r, int x) {
2   int m = (l + r)/2;
3   if (l > r)
4     return -1;
5   if (dados[m] == x)
6     return m;
7   else if (dados[m] < x)
8     return busca_binaria(dados, m + 1, r, x);
9   else
```

Comparando recursão X iteração

Normalmente algoritmos recursivos são:

- **mais simples de entender**
- **menores e mais fáceis de programar**
- **mais “elegantes”**

Mas algumas vezes podem ser

- **muito ineficientes (quando comparados a algoritmos iterativos para o mesmo problema)**

Estratégia ideal:

1. encontrar algoritmo recursivo para o problema

2. reescrevê-lo como um algoritmo iterativo

Fibonacci: recursivo vs. iterativo

Sequência de Fibonacci:

Fibonacci: recursivo vs. iterativo

Sequência de Fibonacci: 1, 1,

Fibonacci: recursivo vs. iterativo

Sequência de Fibonacci: 1, 1, 2,

Fibonacci: recursivo vs. iterativo

Sequência de Fibonacci: 1, 1, 2, 3,

Fibonacci: recursivo vs. iterativo

Sequência de Fibonacci: 1, 1, 2, 3, 5,

Fibonacci: recursivo vs. iterativo

Sequência de Fibonacci: **1, 1, 2, 3, 5, 8,**

Fibonacci: recursivo vs. iterativo

Sequência de Fibonacci: 1, 1, 2, 3, 5, 8, 13, . . .

Fibonacci: recursivo vs. iterativo

Sequência de Fibonacci: 1, 1, 2, 3, 5, 8, 13, . . .

```
1 int fib_rec(int n) {
2   if (n == 1)
3     return 1;
4   else if (n == 2)
5     return 1;
6   else
7     return fib_rec(n-2)+
      fib_rec(n-1);
8 }
```

Fibonacci: recursivo vs. iterativo

Sequência de Fibonacci: 1, 1, 2, 3, 5, 8, 13, . . .

```
1 int fib_rec(int n) {
2   if (n == 1)
3     return 1;
4   else if (n == 2)
5     return 1;
6   else
7     return fib_rec(n-2)+
      fib_rec(n-1);
8 }
```

```
1  int fib_iterativo(int n) {
2    int ant, atual, prox, i;
3    ant = atual = 1;
4    for (i = 3; i <= n; i++) {
5      prox = ant + atual;
6      ant = atual;
7      atual = prox;
8    }
9    return atual;
10 }
```

Fibonacci: recursivo vs. iterativo

Sequência de Fibonacci: 1, 1, 2, 3, 5, 8, 13, . . .

```
1 int fib_rec(int n) {
2   if (n == 1)
3     return 1;
4   else if (n == 2)
5     return 1;
6   else
7     return fib_rec(n-2)+
      fib_rec(n-1);
8 }
```

```
1  int fib_iterativo(int n) {
2    int ant, atual, prox, i;
3    ant = atual = 1;
4    for (i = 3; i <= n; i++) {
5      prox = ant + atual;
6      ant = atual;
7      atual = prox;
8    }
9    return atual;
10 }
```

Número de operações:

- iterativo: $\approx n$
- recursivo: $\approx \texttt{fib}(n)$

Exercícios

Calculando o Fatorial

Escreva uma função recursiva e iterativa que calcule o fatorial de um elemento *n*

Calculando Fibonacci

Escreva uma função recursiva e iterativa que descubra o valor do Fibonacci de uma determinada posição do vetor de *n* posições.

Calculando o Máximo

Escreva uma função recursiva e iterativa que calcule o máximo de um vetor dado, com n elementos

Busca Binária

Escreva uma função recursiva e iterativa que faça uma busca binária entre 10 elementos que o usuário digitar

Palíndromos

Escreva uma função recursiva e iterativa que descubra se uma palavra digitada pelo usuário é palíndromo ou não.