Tópicos Avançados em Estrutura de Dados

Atividade 7

Bruna Galastri Guedes	18.00189-0
Daniel Ughini Xavier	18.00022-3
Rodolfo Cochi Bezerra	18.00202-0
Vítor Martin Simoni	18.00050-9
Leonardo Cury Haddad	18.00442-3
Leonardo de Barros Rodrigues	18.02401-7

17/05/2020

Questão 1

a)

$$\begin{aligned} func(n) &= 2 & \text{, se } n = 1; \\ func(n) &= 2 * func(n-1) & \text{, se } n > 1; \end{aligned}$$

b)

Expandindo a função, temos:

$$\begin{aligned} func(n) &= 2 * func(n-1) \\ &= 2 * (2 * func(n-2)) \\ &= 2 * (2 * (2 * func(n-3)) \cdots \\ &\cdots \\ func(n) &= 2^k * \cdots 2 * func(k-1) \end{aligned}$$

Sabendo que $func(1) = 2$ podemos deduzir que, o caso base $(k-1) = 2$, portanto $k = 3$.

c)

Substituindo k = 3, temos:

$$\begin{aligned} func(n) &= 2^3 * \cdots 2 * func(3-1) \\ &= 2^3 * \cdots 2 * func(2) \\ &= 2^3 * \cdots 2 * (2 * func(1)) \\ &= 2^3 * \cdots 2 * 4 \end{aligned}$$

Sendo assim, a complexidade é de $O(2^6)$.

d)

```
#include <stdio.h>

int main(int argc, char const *argv[]){
    // Inicializando variaveis
    int n, r;

    printf("Digite um valor de n: ");
    scanf("%i", &n);
    printf("\n");

    r = 1;

    // Calculo
    while (n > 0){
        r *= 2;
        n--;
    }
```

```
    // Mostrando o resultado
    printf("Resposta: %i\n", r);
    return 0;
}
```

Questão 2

a)

$$\begin{aligned} func(n) &= 1 \quad &, \text{se } n = 1; \\ func(n) &= 3 + func(n-1) \quad &, \text{se } n > 1; \end{aligned}$$

b)

Expandindo a função, temos:

$$\begin{aligned} func(n) &= 3 + func(n-1) \\ &= 3 + (3 + func(n-2)) \\ &= 3 + (3 + (3 + func(n-3)) \cdots \\ &\cdots \\ func(n) &= 3k + func(n-k) \end{aligned}$$

Sabendo que $func(n-k) = 1$ podemos deduzir que, o caso base $(k-1) = 1$, portanto $k = n - 1$.

c)

Substituindo k = n-1, ou seja, n-k = 1, temos:

$$\begin{aligned} func(n) &= 3 * (n-1) + func(1) \\ &= (3n - 3) + func(1) \\ &= 3n - 3 + 1 \\ &= 3n - 2 \end{aligned}$$

Sendo assim, a complexidade é de $O(n)$.

d)

```
#include <stdio.h>

int main(int argc, char const *argv[]){
    // Inicializando variaveis
    int n, r;

    printf("Digite um valor de n: ");
    scanf("%i", &n);
    printf("\n");

    r = 1;
```

```
    // Calculo
    while (n > 1){
        r += 3;
        n--;
    }

    // Mostrando o resultado
    printf("Resposta: %i\n", r);
    return 0;
}
```

Questão 3

a)

$$\begin{aligned} func(n) = 1 &\quad \text{, se } n = 0; \\ func(n) = n * func(n-1) &\quad \text{, se } n > 0; \end{aligned}$$

b)

Expandindo a função, temos:

$$\begin{gathered} func(n) = n * func(n-1) \\ = n * ((n-1) * func(n-2)) \\ = n * ((n-1) * ((n-2) * func(n-3)) \cdots \\ \cdots \\ func(n) = n * (n-1) * (n-2) * \cdots k * func(k-1) \end{gathered}$$

Sabendo que $func(0) = 1$ podemos deduzir que, o caso base $(k-1) = 1$, portanto $k = 2$.

c)

Substituindo k = 2, temos:

$$\begin{gathered} func(n) = n * (n-1) * (n-2) * \cdots 2 * func(2-1) \\ = n * (n-1) * (n-2) * \cdots 2 * func(1) \\ = n * (n-1) * (n-2) * \cdots 2 * (1 * func(0)) \\ = n * (n-1) * (n-2) * \cdots 2 * 1 \end{gathered}$$

Sendo assim, a complexidade é de $O(n!)$.

d)

```
#include <stdio.h>

int main(int argc, char const *argv[]){
    // Inicializando variaveis
```

```
    int n, r;

    printf("Digite um valor de n: ");
    scanf("%i", &n);
    printf("\n");

    r = 1;

    // Calculo
    while (n > 1){
        r *= n;
        n--;
    }

    // Mostrando o resultado
    printf("Resposta: %i", r);
    return 0;
}
```

Questão 4

a)

$$
\begin{aligned}
func(n) &= 2 \quad &, \text{se } n = 2; \\
func(n) &= 2 * func(n-1) + 3 \quad &, \text{se } n > 2;
\end{aligned}
$$

b)

Expandindo a função, temos:

$$
\begin{aligned}
func(n) &= (2 * func(n-1)) + 3 \\
&= 2 * ((2 * func(n-2)) + 3) + 3 \\
&= 2 * ((2 * ((2 * func(n-3)) + 3) + 3) + 3 \cdots \\
&\cdots \\
func(n) &= (2^k * func(n-k)) + 3k
\end{aligned}
$$

Sabendo que $func(2) = 2$ podemos deduzir que, o caso base $(n-k) = 2$, portanto $k = n-2$.

c)

Substituindo k = n-2, temos:

$$
\begin{aligned}
func(n) &= (2^{n-2} * func(2)) + 3 * (n-2) \\
&= (2^{n-2} * 2) + 3n - 6 \\
&= 2^{n-1} + 3n - 6
\end{aligned}
$$

Sendo assim, a complexidade é de $O(2^n)$.

d)

```
#include <stdio.h>

int main(int argc, char const *argv[]){
    // Inicializando variaveis
    int n, r;

    printf("Digite um valor de n: ");
    scanf("%i", &n);
    printf("\n");

    r = 2;

    // Calculo
    while(n > 2){
        n--;
        r = r * 2 + 3;
    }

    // Mostrando o resultado
    printf("Resposta: %i", r);
    return 0;
}
```

Questão 5

a)

$$\begin{aligned} func(n) &= 1 \quad &&, \text{se } n = 0 \text{ ou } n = 1; \\ func(n) &= 2 * func(n-2) + 10 \quad &&, \text{se } n > 1; \end{aligned}$$

b)

Expandindo a função, temos:

$$\begin{aligned} func(n) &= (2 * func(n-2)) + 10 \\ &= 2 * ((2 * func(n-4)) + 10) + 10 \\ &= 2 * ((2 * ((2 * func(n-6)) + 10) + 10) + 10 \\ &= (2 * 2 * 2 * func(n-6)) + (2 * 2 * 10 + 2 * 10 + 10) \\ &= 2^3 * func(n-6) + 10 * (2^2 + 2^1 + 2^0) \cdots \end{aligned}$$

$$func(n) = 2^k * func(n-2k) + 10 * 2^k - 10$$

Sabendo que $func(1) = 1$ podemos deduzir que, o caso base é $(n - 2k) = 1$, portanto $k = \frac{n-1}{2}$.

c)

Substituindo $k = \frac{n-1}{2}$, temos:

$$\begin{aligned} func(n) &= (2^{\frac{n-1}{2}} * func(n - (n-1))) + 10 * (2^{\frac{n-1}{2}}) - 10 \\ &= (2^{\frac{n-1}{2}} * 1) + 10 * (2^{\frac{n-1}{2}}) - 10 \\ &= 11 * 2^{\frac{n-1}{2}} - 10 \end{aligned}$$

Sendo assim, a complexidade é de $O(2^n)$.

d)

```c
#include <stdio.h>

int main(int argc, char const *argv[]){
    // Inicializando variaveis
    int n, r;

    printf("Digite um valor de n: ");
    scanf("%i", &n);
    printf("\n");

    r = 1;

    // Calculo
    while(n > 1){
        n -= 2;
        r = r * 2 + 10;
    }

    // Mostrando o resultado
    printf("Resposta: %i", r);
    return 0;
}
```