Trabalho 1 – Estruturas Discretas

Lucas Hardman – 1113567

Stephanie Fay – 1121598

**Exercício 1**

**(a)**

Algoritmo em pseudocódigo:

função quociente (x, y, k){

se (k igual a 1)

então retorna 1

retorna x^(k-1)+ y\*q[k-1]

}

**(b)**

Algoritmo em Python:

def quociente (x, y, k):

if k == 1:

return 1 # para quando q[k - 1] = 1

return x\*\*(k - 1) + (y \* quociente (x, y, k - 1)) # q[k] = x^(k - 1) + y\*q[k - 1]

Testes:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| X | Y | K | Q[K] | Tempo | Execuções | Execuções por segundo |
| 1 | 2 | 5 | 31 | 5,00028610 | 481317 | 96257 |
| 1 | 2 | 10 | 1023 | 5,00028610 | 255345 | 51066 |
| 1 | 2 | 15 | 32767 | 5,00028586 | 178564 | 35710 |
| 1 | 2 | 20 | 1048575 | 5,00028586 | 114964 | 22991 |
| 1 | 2 | 25 | 33554431 | 5,00028586 | 105155 | 21029 |
| 1 | 5 | 2 | 6 | 5,00028610 | 978300 | 195648 |
| 1 | 10 | 2 | 11 | 5,00028610 | 787903 | 157571 |
| 1 | 15 | 2 | 16 | 5,00028586 | 663620 | 132716 |
| 1 | 20 | 2 | 21 | 5,00028586 | 716388 | 143269 |
| 1 | 25 | 2 | 26 | 5,00028610 | 873504 | 174690 |
| 5 | 1 | 2 | 6 | 5,00028586 | 711613 | 142314 |
| 10 | 1 | 2 | 11 | 5,00028586 | 771478 | 154286 |
| 15 | 1 | 2 | 16 | 5,00028586 | 963514 | 192691 |
| 20 | 1 | 2 | 21 | 5,00028610 | 823931 | 164776 |
| 25 | 1 | 2 | 26 | 5,00028586 | 825827 | 165155 |

Conclusão:

Abaixo estão três gráficos para ilustrar a diferença da mudança dos valores de X, Y e K em relação a quantidade de execuções por segundo. Eixo Y = execuções por segundo e eixo X = X, Y ou K.

Podemos observar no gráfico “Exec/s alterando K” que a quantidade de execuções por segundo diminui bastante quando aumentamos o valor de K, saindo da faixa de 100 mil para 20 mil. O valor de K influencia diretamente (de 1 para 1) na quantidade de vezes que a função é chamada, uma vez que a recursão é chamada sempre com k-1 até que k == 1. Por exemplo, se K = 50 a função vai ser chamada 50 vezes.

Já ao aumentar os valores de X ou Y, observamos pouca alteração na quantidade de execuções por segundo. Alterar os valores de X e Y não influencia na quantidade de recursões da função, a pouca alteração é causada por fatores externos ao programa, como por exemplo outros processos sendo executados em paralelo. Nos gráficos, as quantidades de execuções por segundo estão sempre na faixa de 125 mil até 200 mil.

**Exercício 2:**

**(a)**

**(b)**

**(c)**

**Exercício 3:**

**(a)**

Temos que:

o numero de rodadas é r = 2^k – 1,

o numero de jogos é j = 2^(k – 1),

o numero de equipes é n = 2^k.

Teorema do caso base:

-> k = 1

r = 2^1 – 1 = 1

j = 2^(1 – 1) = 1

n = 2^1 = 2

Ou seja, o caso base é duas equipes jogando uma rodada de um jogo. Como são duas equipes e apenas um jogo, logo uma equipe está competindo contra a outra. Ou seja, nesta rodada (que é apenas um jogo) as equipes estão enfrentando uma equipe diferente.

Teorema do passo indutivo:

Hipótese indutiva: assumimos que o teorema 3 é válido para um k qualquer.

Queremos mostrar que o teorema também é válido para k + 1. Ou seja,

n = 2ˆ(k+1),

r = 2ˆ(k+1)-1.

j = 2ˆ(k+1-1) = 2ˆk.

Dividimos as equipes em dois grupos, A e B, de 2ˆk equipes cada.

Na primeira etapa, tratamos cada grupo como torneios diferentes e **ao mesmo tempo.** Pela hipótese indutiva temos dois torneios, de 2ˆk - 1 rodadas cada e 2ˆ(k - 1) jogos. Como as rodadas de cada torneio são simultâneas, temos ao todo 2ˆk - 1 rodadas com 2\*(2ˆ(k - 1)) = 2ˆk jogos cada.

Ao fim da primeira etapa, todas equipes de A e B jogaram contra todas as outras equipes de seus respectivos grupos. Resta que cada equipe de A jogue contra cada equipe de B.

Para a segunda etapa, vamos considerar o grupo A como uma fila e o grupo B como uma fila circular.

A cada rodada da segunda etapa, a i-ésima equipe de A jogará contra a i-ésima equipe de B, para um total de 2ˆk jogos por rodada (afinal, A e B, cada um, tem 2ˆk equipes).

Ao final de cada rodada, **andamos uma posição na fila circular de B**, e repetimos o processo até que voltemos à posição inicial de B. Nesse ponto, todas as equipes de A terão jogado contra todas as equipes de B. Observe que serão 2ˆk rodadas nessa segunda etapa (pois B tem 2ˆk posições/equipes).

Assim, **a primeira e segunda etapas juntas terão (2ˆk - 1) + 2ˆk = 2ˆ(k + 1) – 1 = r rodadas, com 2ˆk = j jogos cada,** como desejado.

**(b)**

Algoritmo em pseudocódigo:

função geraTorneio(k, i){

}

Algoritmo em Python:

def geraTorneio(k, start):

**(c)**