UNIVERSIDADE FEDERAL DE VIÇOSA

DEPARTAMENTO DE INFORMÁTICA

(DPI)

TRABALHO PRATICO 1

Rafael Zardo Crevelari – ES105468

Natascha Siqueira Martinez Palhares - ES105460

Uma imagem contendo Logotipo

Descrição gerada automaticamenteDisciplina: Matemática Discreta

Professor: André Gustavo Dos Santos

13 de julho 2022

**RESPOSTAS:**

**Exercício 1: (Feito por Natascha)**Note que para um número ser escrito como a soma de cubos de inteiros positivos, os números escolhidos para serem elevados ao cubo tem que ser menor ou igual a raiz cúbica do número final. Ou seja, a raiz cúbica de 1729 é aproximadamente 12,0023, então tem que ser escolhidos números menores ou iguais a 12.

Como vimos em sala que o primeiro número que pode ser escrito como a soma de cubos inteiros positivos de duas maneiras diferentes é o 1729, então vamos começar pelo número 1730.

Utilizando o programa exercicio1.cpp, temos que  
  
4104 atende as propriedades: 2^3 + 16^3 = 9^3 + 15^3  
13832 atende as propriedades: 2^3 + 24^3 = 18^3 + 20^3  
20683 atende as propriedades: 10^3 + 27^3 = 19^3 + 24^3  
32832 atende às propriedades: 4^3 + 32^3 = 18^3 + 30^3  
39312 atende as propriedades: 2^3 + 34^3 = 15^3 + 33^3  
40033 atende as propriedades: 9^3 + 34^3 = 16^3 + 33^3  
46683 atende às propriedades: 3^3 + 36^3 = 27^3 + 30^3  
64232 atende as propriedades: 17^3 + 39^3 = 26^3 + 36^3  
65728 atende as propriedades: 12^3 + 40^3 = 31^3 + 33^3  
110656 atende às propriedades: 4^3 + 48^3 = 36^3 + 40^3

O código exercicio1.cpp para depois de encontrar 10 números que podem ser escritos como a soma de cubos inteiros positivos de duas maneiras diferentes. Porém, se quiser encontrar mais números, basta aumentar a restrição do laço de repetição while. Ou seja, se quiser saber 15 números que atendem a essas propriedades basta mudar o cont < 10 para cont < 15, por exemplo.

**Exercício 2: (Feito por Rafael)**

Para responder esse exercício foi criado o software **exercicio2.cpp**, para obter os números **Inválidos**, ou seja, os que não podem ser escritos como a soma de quartas potenciais de 18 valores inteiros. O software gera os números até o tamanho máximo de uma variável int, depois disso a linguagem C++ não consegue computar os próximos números. A tabela da direta foi extraída a partir do software em questão, vale ressaltar que todos os dados obtidos pelo software não estão na tabela, uma vez que encontrar valores para números maiores que 200/300 demandas uma elevada quantidade de tempo pelo software. Logo, podemos perceber na tabela a direita que os números 159, 239 são inválidos, assim temos outros números além do 79 com a propriedade de não poderem ser escritos como a soma de quartas potenciais de 18 valores inteiros.

|  |  |
| --- | --- |
| **Sequência** | |
| **Valor** | **Situação** |
| 0 | Valido |
| ... | ... |
| 78 | Valido |
| **79** | **Invalido** |
| 80 | Valido |
| ... | ... |
| 158 | Valido |
| **159** | **Invalido** |
| 160 | Valido |
| ... | ... |
| 238 | Valido |
| **239** | **Invalido** |

**Exercício 3: (Feito por Natascha)**Utilizando o programa exercicio3.cpp, temos que  
Os 11 primeiros números de Fibonacci que são divisíveis por 5 são:

0, 5, 55, 610, 6765, 75025, 832040, 9227465, 102334155, 1134903170, 512559680, 1820529360

Note que, a partir do 3 termo, conseguimos chegar no próximo divisível por 5, multiplicando o anterior por 11 e somando a esse valor o número anterior ao anterior. Assim, tomamos por definição o primeiro termo como 0 e o segundo como 5.

Dessa forma conseguimos achar os próximos números divisíveis por 5 seguindo essa conjectura:

0  
5  
55 = (11\*5) +0  
610 = (11\*55) +5  
6765 = (11\*610) + 55   
75025 = (11 \* 6765) + 610

E assim por diante....

Além disso, note que, os números de Fibonacci são divisíveis por 5, se e somente se, seu índice for divisível por 5 (posição na sequência de Fibonacci). Assim, temos que,

5 ⇒ índice 5   
55 ⇒ índice 10   
610 ⇒ índice 15

E assim por diante....

Obs: por questões de tamanho do número, o código exercicio3.cpp foi feito para descobrir apenas os 11 primeiros números.

**Exercício 4: (Feito por Rafael)**

Para responder esse exercício foi criado o software **exercicio4.cpp**, para obter os números de Fibonacci. O software gera os números da sequência de Fibonacci até o tamanho máximo de uma variável long long int, depois disso a linguagem C++ não consegue computar os próximos números de Fibonacci. A tabela da direta foi extraída a partir do software em questão, vale ressaltar que todos os dados obtidos pelo software não estão na tabela, apenas estão uma quantidade suficiente para o entendimento da conjectura, caso queira ver os dados completos é recomendado rodar o software.

Com os dados da tabela, podemos formar uma conjectura a respeito dos números de Fibonacci divisíveis por 3. Analisando a tabela, temos os 28 primeiros números de Fibonacci, caso queiramos encontrar os divisíveis por 3 dentro dessa lista, basta dividir o número em questão por 3 e obter o resto 0. Assim, baseado no obtido pelo nosso software, os números com fundo cinza e destacados com negrito são os números divisíveis por 3 dos 28 primeiros números de Fibonacci. A partir disso, podemos perceber um padrão na nossa tabela, em que a cada 3 elementos, o próximo é um número divisível por 3. Ou seja, a partir disso podemos notar outro fato importante, o qual é, todo elemento divisível por 3 nos 28 primeiros números de Fibonacci possui índice divisível por 4, como podemos ver no número 3, o qual possui índice 4 e 4 mod 4 = 0, também fica evidente no número 21, o qual possui índice 8 e 8 mod 4 = 0 e assim se repete nos índices divisíveis por 4 posteriores. Assim, para ver se essa aplicação é valida basta rodar o software e analisar o resto dos dados. Fazendo isso, podemos afirmar que de fato é verdadeiro e podemos concluir uma conjectura que, um número de Fibonacci é divisível por 3 se, e somente se, seu índice é divisível por 4. Outra conjectura que pode ser observada é, note que, conseguimos chegar no próximo divisível por 3, multiplicando o anterior por 7 e diminuindo a esse valor o número anterior do anterior. Assim, tomamos por definição o primeiro termo como 3 e o segundo como 21. Podemos ver isso no exemplo 21(Número divisível por 3 anterior) \* 7 - 3(Número divisível por 3 anterior do anterior) = 144 (Número divisível por 3 atual), e isso é verdadeiro conforme mostrado na tabela a direita.

|  |  |
| --- | --- |
| **Números de Fibonacci** | |
| **Índice** | **Número** |
| 1 | 1 |
| 2 | 1 |
| 3 | 2 |
| **4** | **3** |
| 5 | 5 |
| 6 | 8 |
| 7 | 13 |
| **8** | **21** |
| 9 | 34 |
| 10 | 55 |
| 11 | 89 |
| **12** | **144** |
| 13 | 233 |
| 14 | 377 |
| 15 | 610 |
| **16** | **987** |
| 17 | 1597 |
| 18 | 2584 |
| 19 | 4181 |
| **20** | **6765** |
| 21 | 10946 |
| 22 | 17711 |
| 23 | 28657 |
| **24** | **46368** |
| 25 | 75025 |
| 26 | 121393 |
| 27 | 196418 |
| **28** | **317811** |

**Exercício 5: (Feito por Natascha)**

Queremos verificar que (2n)! / ((n!) (n!)) É divisível por um número primo ao quadrado.

Vamos considerar n > 2, uma vez que,

Para n = 1 ⟹ (2\*1)! / (1! 1!) = 2 e para n = 2 ⟹ (2\*2)! / (2! 2)! = 6

O menor primo é o 2 e o segundo menor é 3, 2^2 = 4 > 2, 3^2 = 9 > 6, assim 2 não é divisível por 4 e 6 não é divisível por 4 nem 9.

Utilizando o programa exercicio5.cpp, temos que

Quando n = 3, então é divisível por 4 = 2^2

Quando n = 5, então é divisível por 4 = 2^2

Quando n = 6, então divisível por 4 = 2^2

Quando n = 7, então é divisível por 4 = 2^2

Quando n = 8, então é divisível por 9 = 3^3

Quando n = 9, então é divisível por 4 = 2^2

Quando n = 10, então é divisível por 4 = 2^2

Quando n = 11, então é divisível por 4 = 2^2

Obs: por questões de tamanho do número, o código exercicio5.cpp foi feito para ir até n=11.

**Exercício 6: (Feito por Rafael)**

Para responder esse exercício foi criado o software **exercicio6.cpp**, para obter os números **Válidos**, ou seja, inteiros ímpares N ≤ 200, tais que não sejam (n ⌊n/2⌋) divisíveis pelo quadrado de um número primo. O software gera os números até o tamanho máximo de uma variável long long int, depois disso a linguagem C++ não consegue computar os próximos números. A tabela da direta foi extraída a partir do software em questão, vale ressaltar que todos os dados obtidos pelo software não estão na tabela, apenas estão uma quantidade suficiente para o entendimento de uma conjectura, caso queira ver os dados completos é recomendado rodar o software.

Com isso, podemos perceber que se N é valido, então N é ímpar e primo, conforme destacado de cinza na tabela a direita. Assim, para ver se essa aplicação é verdadeira basta rodar o software e analisar o resto dos dados. Fazendo isso, podemos afirmar que de fato é verdadeiro e podemos concluir uma conjectura que, se N é valido, então N é ímpar e primo.

|  |  |
| --- | --- |
| Sequência | |
| Valor de N | Situação |
| 1 | Valido |
| 3 | Valido |
| 5 | Valido |
| 7 | Valido |
| 9 | Invalido |
| 11 | Valido |
| 13 | Invalido |
| 15 | Invalido |
| 17 | Valido |
| 19 | Valido |
| 21 | Invalido |
| 23 | Valido |