Desafios de Programação

Super Day One

Com o Rafa:

otimizamos o hardware

Com o Igor:

otimizamos a programação
para aproveitar o hardware

O que resta?

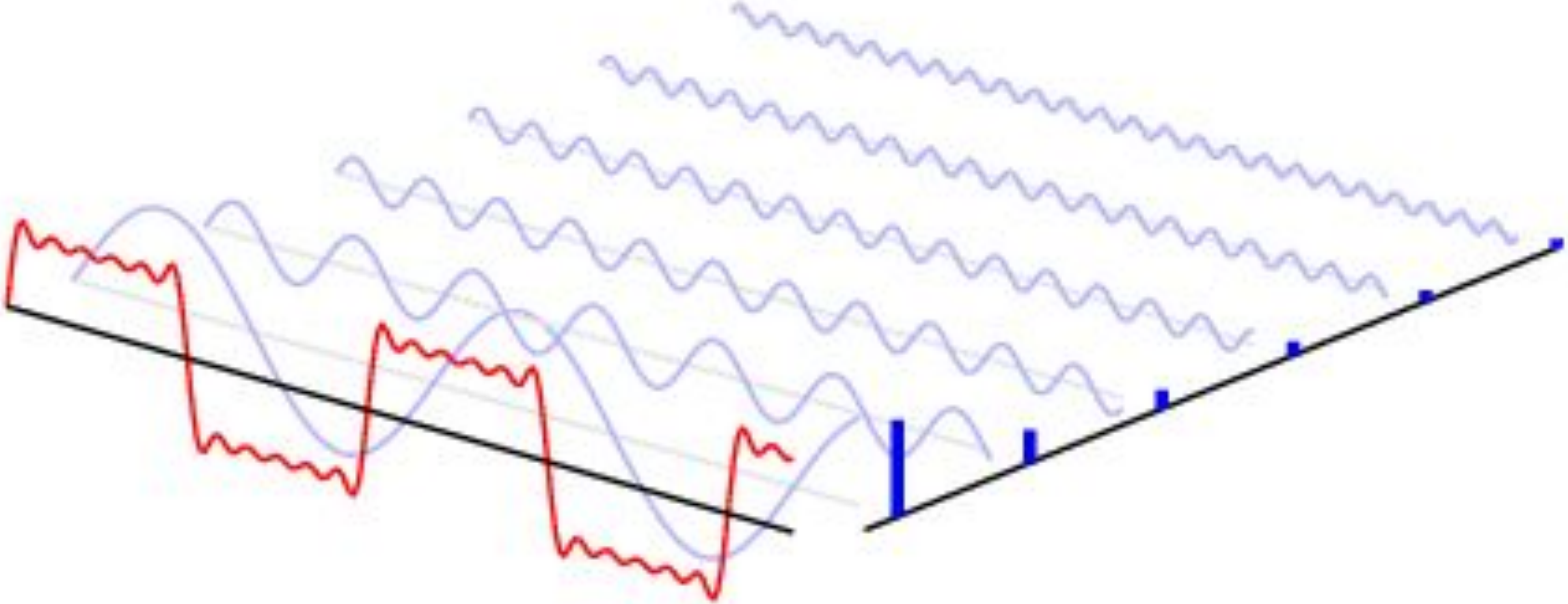
```
#define PARTE1
//#define PARTE2
//#define PARTE3
```

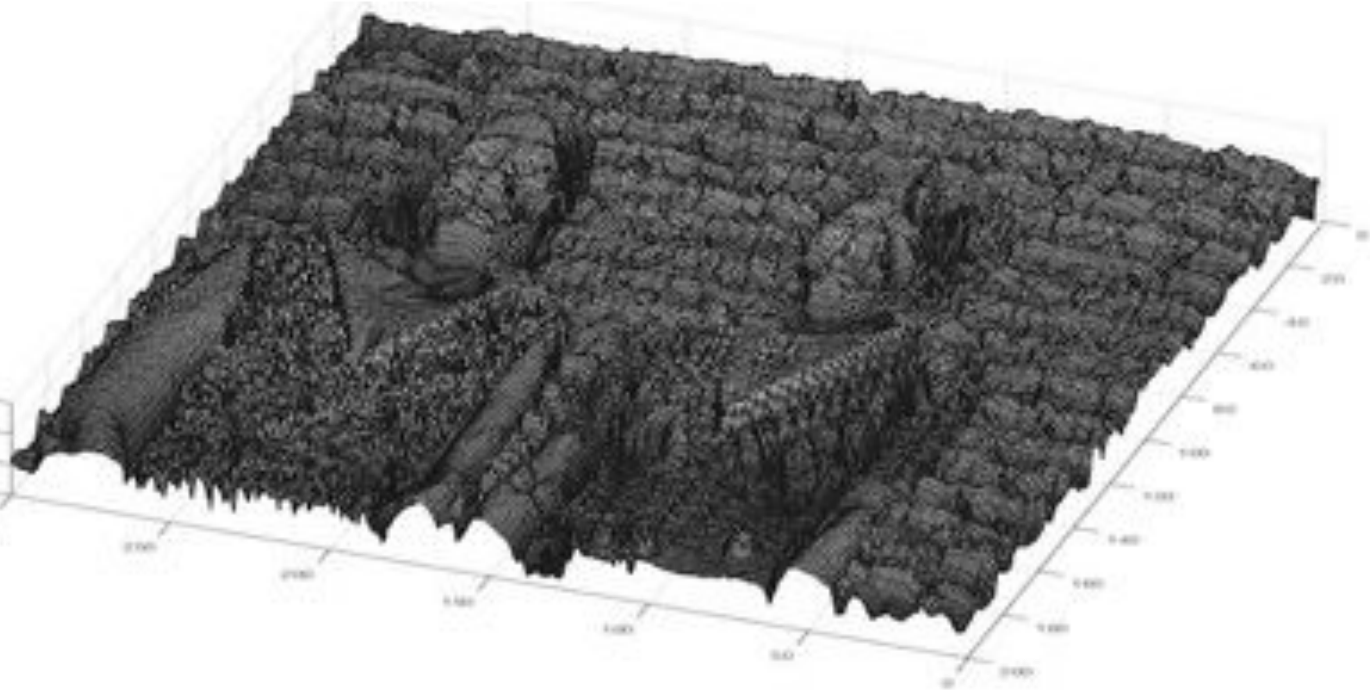
antes

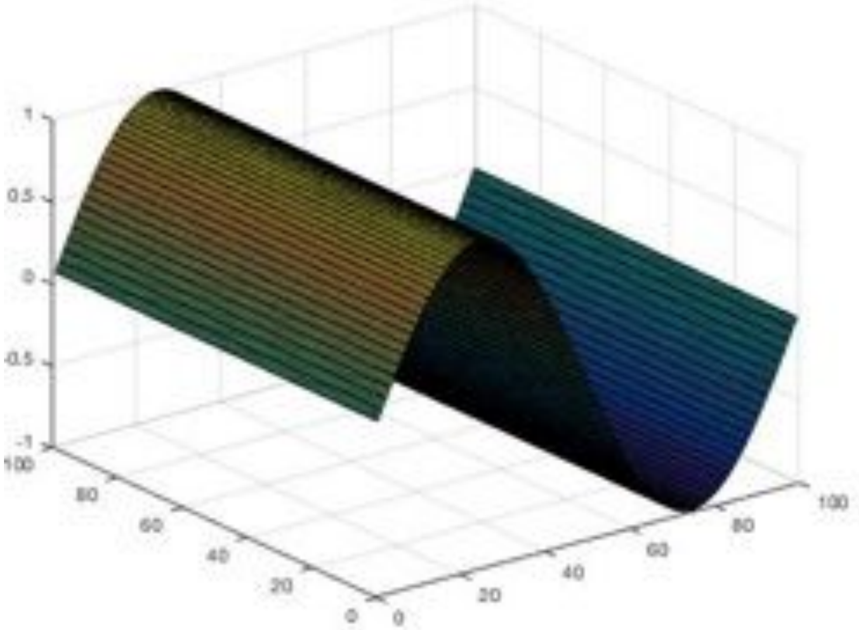
depois

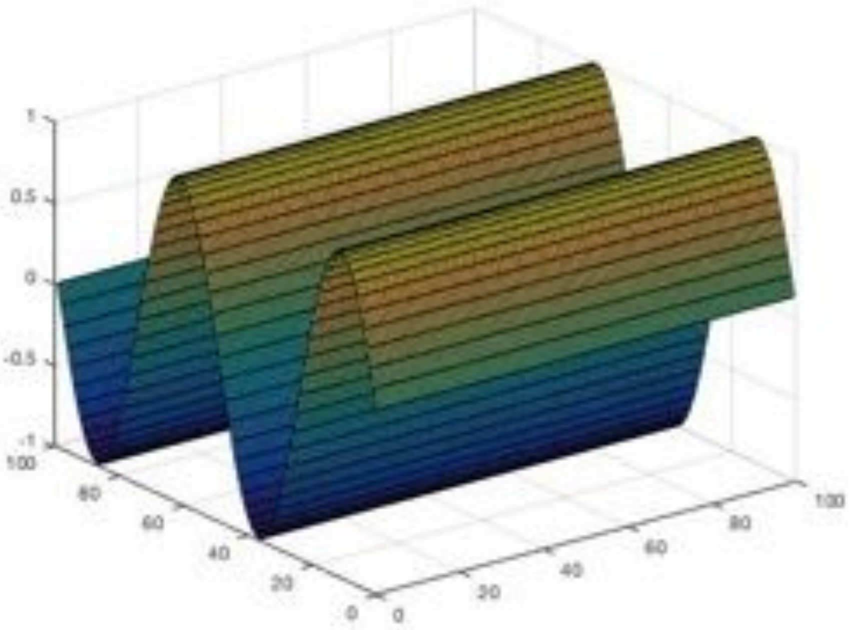
```
//#define PARTE1
#define PARTE2
//#define PARTE3
```

O que o botão faz?




3000
2000
1000
0
3000
2500
2000
1500
1000
500
0
0
200
400
600
800
1000
1200
1400
1600
1800
2000

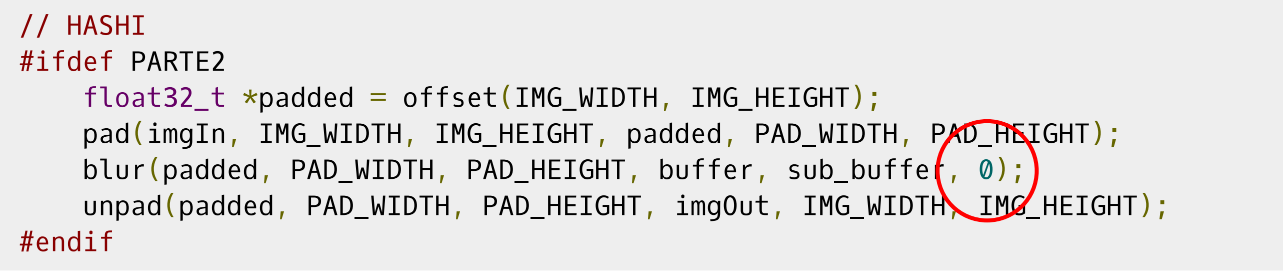

1
0.5
0
-0.5
-1
100
80
60
40
20
0
0
20
40
60
80
100


1
0.5
0
-0.5
-1
100
80
60
40
20
0
0
20
40
60
80
100

$$\boldsymbol{X}(\boldsymbol{k}) = \sum_{n=0}^{N-1} x(n) \left[\mathbf{cos}\left(\frac{2\pi kn}{N}\right) + \boldsymbol{i}\,\mathbf{sen}\left(\frac{2\pi kn}{N}\right)\right]$$

procure a função `dft` no arquivo `fourier.c`

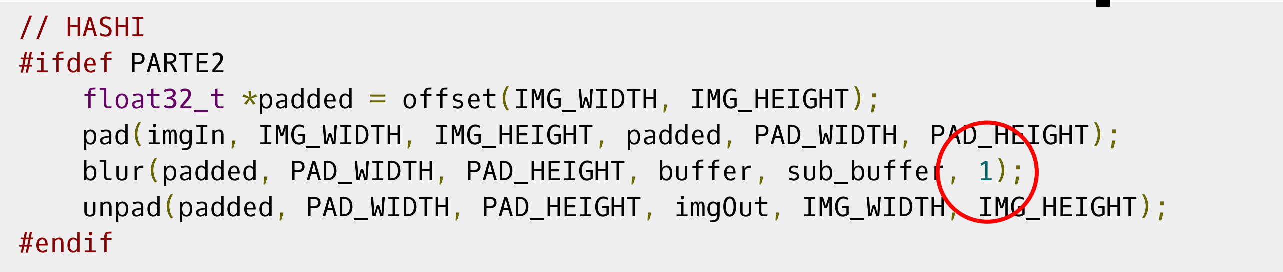
Uma pequena mudança...



```
// HASHI
#ifdef PARTE2
    float32_t *padded = offset(IMG_WIDTH, IMG_HEIGHT);
    pad(imgIn, IMG_WIDTH, IMG_HEIGHT, padded, PAD_WIDTH, PAD_HEIGHT);
    blur(padded, PAD_WIDTH, PAD_HEIGHT, buffer, sub_buffer, 0);
    unpad(padded, PAD_WIDTH, PAD_HEIGHT, imgOut, IMG_WIDTH, IMG_HEIGHT);
#endif
```

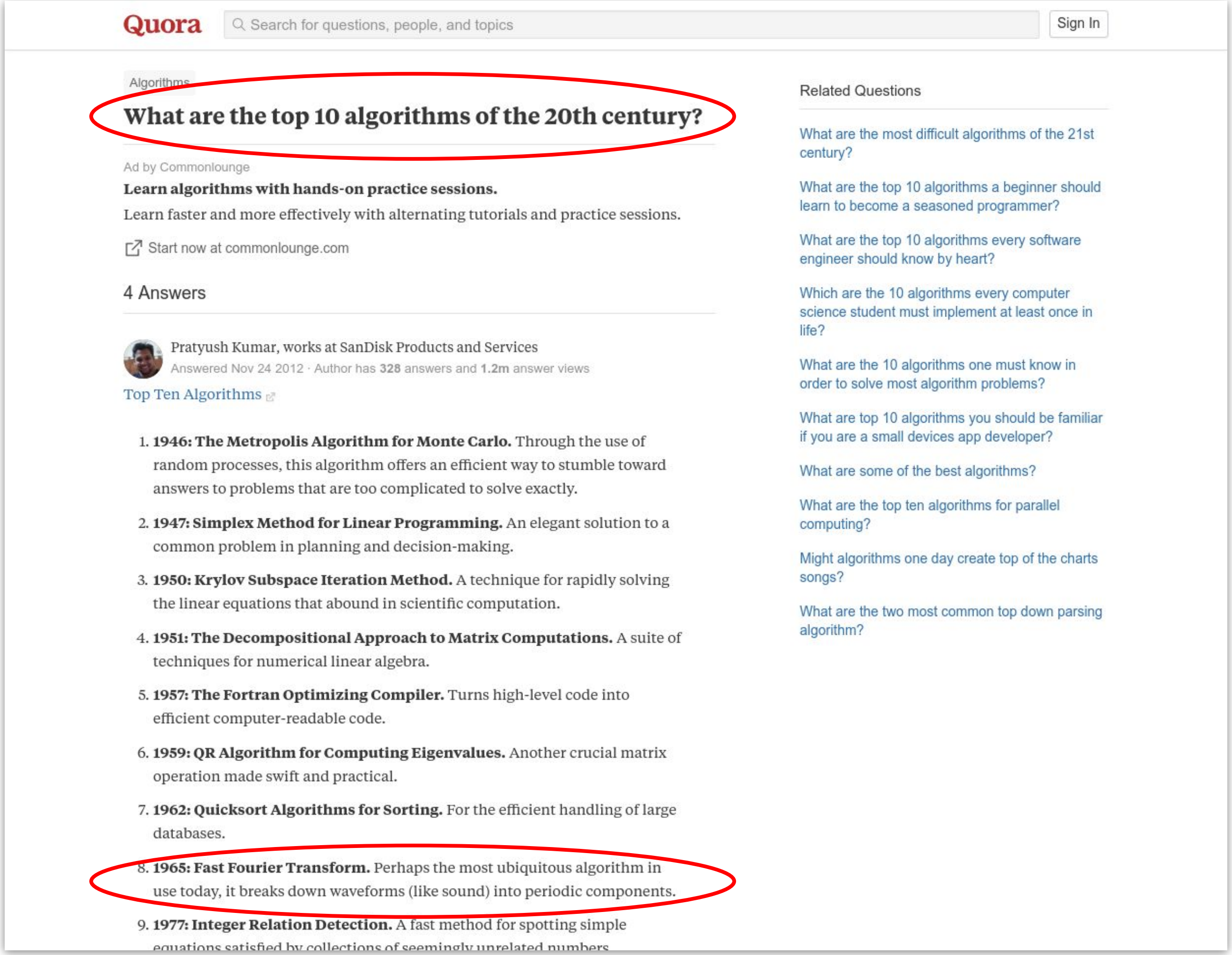
antes

depois



```
// HASHI
#ifdef PARTE2
    float32_t *padded = offset(IMG_WIDTH, IMG_HEIGHT);
    pad(imgIn, IMG_WIDTH, IMG_HEIGHT, padded, PAD_WIDTH, PAD_HEIGHT);
    blur(padded, PAD_WIDTH, PAD_HEIGHT, buffer, sub_buffer, 1);
    unpad(padded, PAD_WIDTH, PAD_HEIGHT, imgOut, IMG_WIDTH, IMG_HEIGHT);
#endif
```

wtf? O_o



Algorithms

What are the top 10 algorithms of the 20th century?

4 Answers

Pratyush Kumar, works at SanDisk Products and Services
Answered Nov 24 2012 · Author has **328** answers and **1.2m** answer views

Top Ten Algorithms

1. **1946: The Metropolis Algorithm for Monte Carlo.** Through the use of random processes, this algorithm offers an efficient way to stumble toward answers to problems that are too complicated to solve exactly.
2. **1947: Simplex Method for Linear Programming.** An elegant solution to a common problem in planning and decision-making.
3. **1950: Krylov Subspace Iteration Method.** A technique for rapidly solving the linear equations that abound in scientific computation.
4. **1951: The Decompositional Approach to Matrix Computations.** A suite of techniques for numerical linear algebra.
5. **1957: The Fortran Optimizing Compiler.** Turns high-level code into efficient computer-readable code.
6. **1959: QR Algorithm for Computing Eigenvalues.** Another crucial matrix operation made swift and practical.
7. **1962: Quicksort Algorithms for Sorting.** For the efficient handling of large databases.
8. **1965: Fast Fourier Transform.** Perhaps the most ubiquitous algorithm in use today, it breaks down waveforms (like sound) into periodic components.
9. **1977: Integer Relation Detection.** A fast method for spotting simple equations satisfied by collections of seemingly unrelated numbers

Related Questions

- What are the most difficult algorithms of the 21st century?
- What are the top 10 algorithms a beginner should learn to become a seasoned programmer?
- What are the top 10 algorithms every software engineer should know by heart?
- Which are the 10 algorithms every computer science student must implement at least once in life?
- What are the 10 algorithms one must know in order to solve most algorithm problems?
- What are top 10 algorithms you should be familiar if you are a small devices app developer?
- What are some of the best algorithms?
- What are the top ten algorithms for parallel computing?
- Might algorithms one day create top of the charts songs?
- What are the two most common top down parsing algorithm?

O que há de tão especial nesse algoritmo?

Por que a diferença entre os dois é tão grande?

Essa diferença pode ser compensada apenas pela camada de hardware ou ponte hardware-software?

Existem algoritmos perfeitos ou existem algoritmos mais adequados para a situação?

Não percam os próximos capítulos!

```
//#define PARTE1
#define PARTE2
//#define PARTE3
```

antes

depois

```
//#define PARTE1
//#define PARTE2
#define PARTE3
```