Paradigmas de Solução de Problemas

Divisão e Conquista – Transformada Rápida de Fourier

Prof. Edson Alves - UnB/FGA

2020

Sumário

Transformada Rápida de Fourier

DFT em $O(N \log N)$

- A divisão e conquista pode ser aplicada no cálculo da DFT para reduzir sua complexidade assintótica
- Na etapa de divisão o sinal é dividido em duas partes de tamanhos aproximadamente iguais
- A conquista acontece quando o sinal tem uma única amostra: neste caso a transformada discreta coincide com a própria amostra
- A fusão permite o cálculo da DFT do sinal a partir das DFTs das duas partes
- Se a fusão for feita em $O(N)$, a recorrência se torna

$$f(N) = 2f(N/2) + (N)$$

- O Teorema Mestre nos diz que a complexidade da transformada passa a ser $O(N \log N)$
- Esta versão da DFT é denominada *Fast Fourier Transform* (FFT)

Decomposição do sinal FFT

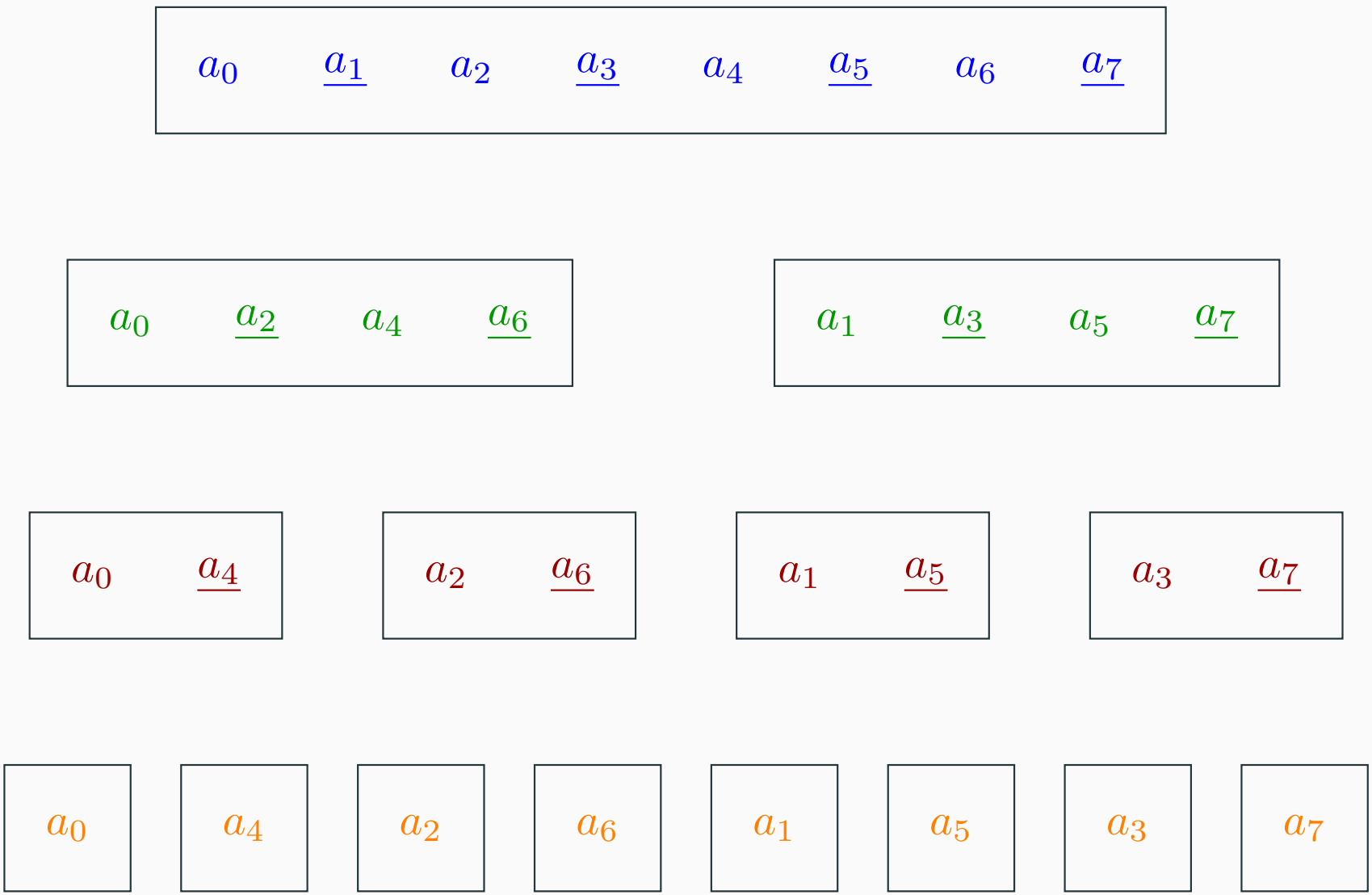
- Considere o sinal $(a_k) = a_0, a_1, \ldots, a_{N-1}$
- Assuma, sem perda de generalidade, que $N = 2^t$, para algum t natural
- Se N não for uma potência de dois, basta adicionar um número suficiente de amostras $a_i = 0$ ao sinal até que N se torne uma potência de dois
- A etapa de divisão, também denominada decomposição do sinal, o sinal é separado em duas partes de tamanho $N/2$: as amostras cujos índices são pares (e_k) e as amostras cujos índices são ímpares (o_k)
- Assim,

$$(e_k) = a_0, a_2, a_4, \ldots, a_{N-2}$$

e

$$(o_k) = a_1, a_3, a_5, \ldots, a_{N-1}$$

Visualização da decomposição do sinal



Decomposição $\times$ ordenação

- Gerando a decomposição por meio da alocação de novos dois subvetores com as cópias dos elementos de índices pares e ímpares permite uma implementação *top-down* da FFT
- Para uma implementação *bottom-up*, é preciso entender o padrão subjacente que surge desta decomposição
- De fato, os elementos que ocupam as folhas nas árvores de decomposição tem índices que correspondem à ordenação dos números $\{0, 1, 2, \ldots, N-1\}$ usando como critério a inversão de sua representação binária
- Assim, por meio de um comparador customizado o este ordenação pode ser feita com complexidade $O(N \log N)$, o que não modifica a complexidade da FFT como um todo

Visualização da ordenação por padrão binário invertido

Índice	Padrão invertido	Padrão original
0	000	000
4	001	100
2	010	010
6	011	110
1	001	100
5	101	101
3	110	011
7	111	111

Implementação da ordenação por padrão binário

```cpp
#include <bits/stdc++.h>

using namespace std;

int reversed(int x, int bits)
{
    int res = 0;

    for (int i = 0; i < bits; ++i)
    {
        res <<= 1;
        res |= (x & 1);
        x >>= 1;
    }

    return res;
}

```

Implementação da ordenação por padrão binário

```
19 template<typename T> vector<T> sortByBits(const vector<T>& xs)
20 {
21     int N = (int) xs.size(), bits = 1;
22
23     while ((1 << bits) != N)
24         ++bits;
25
26     vector<int> is(N);
27     iota(is.begin(), is.end(), 0);
28
29     sort(is.begin(), is.end(), [&bits](int x, int y) {
30         return reversed(x, bits) < reversed(y, bits);
31     });
32
33     vector<T> ans(N);
34
35     for (int i = 0; i < N; ++i)
36         ans[i] = xs[is[i]];
37
38     return ans;
39 }
```

Referências

Referências

1. **CHEEVER**, Erick. The Fourier Series, acesso em 12/08/2020.
2. CP Algorithms. Fast Fourier Transform, acesso em 13/08/2020.
3. **SMITH**, Steven W. The Scientist and Engineer's Guide to Digital Signal Processing, acesso em 17/08/2020.
4. Standford. Lecture 11 – The Fourier Transform, acesso em 13/08/2020.
5. Wikipédia. Discrete Fourier Transform, acesso em 13/08/2020.
6. Wolfram. Fourier Series, acesso em 12/08/2020.
7. Wolfram. Fourier Transform, acesso em 13/08/2020.