Oblikovanje i analiza algoritama

Matej Mihelčić

Prirodoslovno-matematički fakultet, Sveučilište u Zagrebu

matmih@math.hr

11. siječnja, 2024.

Definicija diskretne Fourierove transformacije

- **Definicija:** Diskretna Fourierova transformacija reda n je pridruživanje $\vec{a} \mapsto \vec{y}$:

$$y_k = A(\omega_n^k), \ \ k = 0, \ldots, n-1$$

Pripadnu transformaciju označavamo s $\vec{y} = DFT_n(\vec{a})$, a pripadnu Vandermondeovu matricu s:

$$V_n = V_n(\omega_n^0, \omega_n^1, \ldots, \omega_n^{n-1})$$

Inverzna diskretna Fourierova transformacija je obratno pridruživanje $\vec{y} \mapsto \vec{a}$, označavamo ju s $\vec{a} = DFT_n^{-1}(\vec{y})$.

- Računanje diskretne Fourierove transformacije reda n za zadani vektor $\vec{a}$ je problem evaluacije ili izračunavanja u svim n-tim korijenima iz jedinice. Uočite da po definiciji vrijedi:

$$\vec{y} = DFT_n(\vec{a}) \Leftrightarrow \vec{y} = V_n(\vec{a})$$

Definicija diskretne Fourierove transformacije

- Analogno, računanje inverzne transformacije reda n za zadani vektor $\vec{y}$ je problem interpolacije u svim n-tim korijenima iz jedinice, što možemo zapisati kao

$$\vec{a} = DFT_n^{-1}(\vec{y}) \Leftrightarrow \vec{y} = V_n(\vec{a}) \Leftrightarrow \vec{a} = V_n^{-1}(\vec{y})$$

- Pokazali smo da se V_n^{-1} može vrlo jednostavno izraziti preko V_n, što nam omogućava da problem interpolacije svedemo na problem evaluacije s malo drugačijom Vandermondeovom matricom. Praktično, to znači da za oba problema možemo koristiti isti algoritam (uz minimalne modifikacije).

Primjer FFT

- Neka je $A(z) = 2 - 3z + 4z^2 + 7z^3 - 3z^4 + 9z^5$
  Polinom A možemo zapisati kao:
  $A(z) = (2 + 4z^2 - 3z^4) + z(-3 + 7z^2 + 9z^4)$,
  $A(z) = A^{[0]}(z) + zA^{[1]}(z)$
- Odnosno: $A(z) = A^{[0]}(z^2) + zA^{[1]}(z^2)$
- Primijetimo da je: $A(-z) = A^{[0]}(z^2) - zA^{[1]}(z^2)$
- Cilj nam je izračunati vrijednost polinoma

$$A(z) = \sum_{j=0}^{n-1} a_j z^j$$

  reda n u svim n-tim korijenima jedinice.
- Bez smanjenja općenitosti možemo pretpostaviti da je $n = 2^m$, za neki $m \in \mathbb{N}_0$.

Brza Fourierova transformacija

- Uvijek možemo povećati red polinoma dodavanjem vodećih članova s koeficijentima jednakim nula.
- Ovim postupkom n se može povećati, ali sigurno za faktor manji od 2.
- Osnovni oblik FFT-a dobivamo rastavom polinoma A na parne i neparne dijelove prema indeksu koeficijenata (ili eksponenata od z):

  $A(z) = A^{[0]}(z^2) + zA^{[1]}(z^2)$, gdje su:
  $A^{[0]}(z^2) = a_0 + a_2 z^2 + \cdots + a_{n-2} z^{\frac{n}{2}-1}$
  $A^{[1]}(z^2) = a_1 + a_3 z^2 + \cdots + a_{n-1} z^{\frac{n}{2}-1}$
  "parni" i "neparni" polinom za A.
- Označimo s $\vec{a}^{[0]}$ i $\vec{a}^{[1]}$ pripadne vektore koeficijenata polinoma:

$$\vec{a}^{[0]} = (a_0, a_2, \ldots, a_{n-2}),$$

$$\vec{a}^{[1]} = (a_1, a_3, \ldots, a_{n-1}).$$

Brza Fourierova transformacija

- Dakle, problem reda n svodimo na dva problema reda $n/2$.
- Treba uspostaviti vezu između $DFT_n(\vec{a})$ i $DFT_{n/2}(\vec{a})$, nizova $\vec{a}^{[0]}$ i $\vec{a}^{[1]}$.
- Označimo s $\vec{y}^{[0]}$ i $\vec{y}^{[1]}$ pripadne diskretne Fourierove transformacije za nizove $\vec{a}^{[0]}$ i $\vec{a}^{[1]}$:

$$\vec{y}^{[0]} = DFT_{n/2}(\vec{a}^{[0]})$$

$$\vec{y}^{[1]} = DFT_{n/2}(\vec{a}^{[1]})$$

- Vrijedi (dokažite - dz):

$$y_k = y_k^{[0]} + \omega_n^k y_k^{[1]}, k = 0, \ldots, \frac{n}{2} - 1,$$

$$y_{k+n/2} = y_k^{[0]} - \omega_n^k y_k^{[1]}, k = 0, \ldots, \frac{n}{2} - 1.$$

Brza Fourierova transformacija

$$
\begin{aligned}
k \to \quad & \left[V_{n/2} \begin{pmatrix} a_0 \\ a_2 \\ \vdots \\ a_{n-2} \end{pmatrix} + \omega_n^k V_{n/2} \begin{pmatrix} a_1 \\ a_3 \\ \vdots \\ a_{n-1} \end{pmatrix} \right. \\
k + n/2 \to \quad & \left. V_{n/2} \begin{pmatrix} a_0 \\ a_2 \\ \vdots \\ a_{n-2} \end{pmatrix} - \omega_n^k V_{n/2} \begin{pmatrix} a_1 \\ a_3 \\ \vdots \\ a_{n-1} \end{pmatrix} \right]
\end{aligned}
$$

FFT - rekurzivni algoritam

```
1  function FFT(ā, ω){
2    /*ulaz - kompleksni vektor ā
3    izlaz - ȳ = DFT_n(ā)
4    pretpostavka - n = 2^m, m ∈ ℕ_0, ne provjerava se*/
5    if(ω == 1) return ā; //ȳ = ā za n = 1
6    else{
7          ā^[0] = (a_0, a_2, ..., a_{n-2});
8          ā^[1] = (a_1, a_3, ..., a_{n-1});
9          ȳ^[0] = FFT(ā^[0], ω^2);
10         ȳ^[1] = FFT(ā^[1], ω^2);
11         ω_n = exp(2πi/n);
12         ω = 1;
13         for(k = 0; k<=n/2-1;k++){
14               t = ω · y_k^[1]; //t = w_n^k · y_k^[1]
15               y_k = y_k^[0] + t;
16               y_{k+n/2} = y_k^[0] − t;
17               ω = ω · ω_n; } }
18   return ȳ;//kompleksni vektor duljine n
19 }
```

Složenost *FFT*-a

- Rekurzivni algoritam je prikladan za osnovnu analizu složenosti.
- Rekurzivna varijanta *FFT* algoritma se vrlo rijetko koristi u praksi. Iterativna verzija algoritma je brža (vremenska složenost je istog reda veličine uz manju multiplikativnu konstantu).
- Gotovi potprogrami iz raznih biblioteka su optimizirani (ponekad do razine arhitekture). Preporučljivo koristiti takve biblioteke (npr. `FFTW`).
- Napravit ćemo grubu procjenu vremenske složenosti rekurzivnog *FFT* algoritma, uz pretpostavku sekvencijalnog izvršavanja svih naredbi (naročito rekurzivnih poziva).

Složenost *FFT*-a

- **Teorem**: Vremenska složenost rekurzivnog izvršavanja *FFT* algoritma, uz pretpostavku sekvencijalnog izvršavanja svih naredbi, za vektor duljine $n = 2^m$ je $T(n) \in \Theta(nlog_2 n)$, uz pretpostavku da su sve operacije na komponentama vektora elementarne.
- **Dokaz**: Rekurzivni dio algoritma ima 2 poziva *FFT*-a na nizovima duljine $n/2$. $T(n) = 2T(n/2) + \mathcal{O}(n) \Rightarrow T(n) \in \Theta(nlog_2 n)$.
- Dva poziva funkcije *FFT* nad poljem duljine $n/2$ su nezavisni (možemo ih izvršavati paralelno bez korištenja sinkronizacijskih mehanizama).
- **Teorem**: Vremenska složenost rekurzivnog izvršavanja *FFT* algoritma za vektor duljine $n = 2^m$ na barem $n/2$ procesora je $T(n) \in \Theta(log_2 n)$, uz pretpostavku da su sve operacije na komponentama vektora elementarne.
- *FFT* se može gotovo idealno paralelizirati jer paralelno trajanje odgovara sekvencijalnom podijeljenom s brojem procesora.

Složenost *FFT*-a

- **Zadatak** Izračunajte aritmetičku složenost *FFT*-a: broj aditivnih i multiplikativnih ($A(n)$ i $M(n)$) kompleksnih aritmetičkih operacija.
- **Teorem**: Vremenska složenost množenja dvaju polinoma stupnja $n-1$ je $T(n) \in \Theta(nlog_2 n)$.
- **Dokaz**: