Efikasno računanje DFTa
Gercelov algoritam
FFT - Fast Fourier Transform

1

Diskretna Furijeova transformacija - DFT

$$X[k] = X\left[k\frac{2\pi}{N}\right] = \sum_{n=0}^{N-1} x[n]e^{-jk\frac{2\pi}{N}n} \quad k = 0,1,\dots,N-1$$

U direktnoj primeni, izračunavanju, izraza pojavljuju se četiri osnovne operacije:

- izračunavanje trigonometrijskih funkcija,
- množenje,
- sabiranje i
- izračunavanje indeksa (adresa podataka u memoriji).

Broj potrebnih operacija za izračunavanje svih N DFT odbiraka, u najopštijem slučaju kada je $x[n]$ kompleksna sekvenca, iznosi:

1. Broj izračunavanja trigonometrijskih funkcija – $2N^2$(za svako k i n, sin i cos)
2. Broj realnih množenja – $4N^2$
3. Broj realnih sabiranja –$2N(2N-1) \approx 4N^2$

Broj izračunavanja indeksa, kao i njihov uticaj na složenost izračunavanja, teško je proceniti jer zavisi od arhitekture računarskog sistema i programskog jezika koji se koristi.

Cilj -> smanjiti potreban broj izračunavanja

2

Računanje sin i cos

$$e^{-jk(n+1)\frac{2\pi}{N}} = e^{-jkn\frac{2\pi}{N}} \cdot e^{-jk\frac{2\pi}{N}}$$

$$e^{-jk\frac{2\pi}{N}} = \cos\left(k\frac{2\pi}{N}\right) - j\sin\left(k\frac{2\pi}{N}\right)$$

Rekurentna formula

$$\cos\left(k(n+1)\frac{2\pi}{N}\right) = \cos\left(kn\frac{2\pi}{N}\right)\cos\left(k\frac{2\pi}{N}\right) - \sin\left(kn\frac{2\pi}{N}\right)\sin\left(k\frac{2\pi}{N}\right)$$

$\cos(0) = 1 \quad \sin(0) = 0$

$$\sin\left(k(n+1)\frac{2\pi}{N}\right) = \cos\left(kn\frac{2\pi}{N}\right)\sin\left(k\frac{2\pi}{N}\right) + \sin\left(kn\frac{2\pi}{N}\right)\cos\left(k\frac{2\pi}{N}\right)$$

$\sin\left(k\frac{2\pi}{N}\right)$ i $\cos\left(k\frac{2\pi}{N}\right)$ se računaju samo jednom

Na ovaj način se $2N^2$ izračunavanja trigonometrijskih funkcija zamenjuje sa $2N$ izračunavanja trigonometrijskih funkcija putem $4N^2$ realnih množenja i $2N^2$ realnih sabiranja. Zauzeće memorije je isto kao kod direktnog postupka. Rekurzivni metod ima i jedan nedostatak koji se ogleda u akumulaciji grešaka tokom izračunavanja. Zbog toga se ovaj metod primenjuje samo na računarima opšte namene kod kojih se varijable predstavljaju u formatu sa pokretnom tačkom i velikim brojem bita u slučajevima kada dužina sekvence koja se transformiše N nije suviše velika.

3

Gercelov algoritam (Goertzel)

4

Kao što se vidi iz prethodnog, direktno izračunavanje DFT zahteva veliki broj računskih operacija čak i kada se koristi tablični ili rekurzivni pristup izračunavanju trigonometrijskih funkcija. Zbog toga će biti razmotreni efikasni postupci za izračunavanje svih odbiraka DFT. Međutim, takvi postupci nisu efikasni kada je potrebno odrediti mali broj DFT odbiraka. U takvim slučajevima pogodno je upotrebiti poboljšani direktni metod koji koristi osobinu periodičnosti eksponencijalnog faktora

$$e^{-jkn\frac{2\pi}{N}} = W_N^{kn}$$

$$W_N^{kN} = 1 = W_N^{-kN}$$

$$X[k] = \sum_{n=0}^{N-1} x[n]W_N^{kn} = W_N^{-kN}\sum_{n=0}^{N-1} x[n]W_N^{kn} = \sum_{n=0}^{N-1} x[n]W_N^{-k(N-n)}$$

što ima oblik konvolucije sekvence $x[n]$ sa sekvencom W_N^{-kn}.
Ako tako definišemo sekvencu $y_k[m]$ kao konvoluciju konačne sekvence $x[n]$ i sekvence W_N^{-kn}

$$y_k[m] = \sum_{n=0}^{N-1} x[n]W_N^{-k(m-n)}$$

$$X[k] = \sum_{n=0}^{N-1} x[n]W_N^{-k(N-n)} \Rightarrow X[k] = y_k[N]$$

Još samo da izračunamo ovu konvoluciju

$y_k[m] = \sum_{n=0}^{N-1} x[n]W_N^{-k(m-n)}$ odziv diskretnog sistema čiji je impulsni odziv W_N^{-kn} na pobudu $x[n]$,

$X[k] = y_k[N]$ a koji je izračunat u trenutku $m = N$.

Poznato je da je sistem čiji je impulsni odziv W_N^{-kn} opisan diferencnom jednačinom:

$$y_k[n] = W_N^{-k}y_k[n-1] + x[n]$$

$$y_k[-1] = 0$$

Digresija $h[n] = W_N^{-kn}$

$$H(z) = \sum_{n=0}^{\infty} h[n]z^{-n} = \sum_{n=0}^{\infty} W_N^{-kn}z^{-n} = \sum_{n=0}^{\infty}\left(W_N^{-k}z^{-1}\right)^n = \frac{1}{1-W_N^{-k}z^{-1}}$$

$$Y(z) = H(z)X(z)$$

$$Y(z) - Y(z)W_N^{-k}z^{-1} = X(z)$$

$$y[n] - y[n-1]W_N^{-k} = x[n]$$

1 $y_k[n] = W_N^{-k} y_k[n-1] + x[n]$

Nismo smanjili broj izračunavanja (za sada), ali

$y_k[n-1] = W_N^{-k} y_k[n-2] + x[n-1]$

2 $y_k[n-1](-W_N^k) = (W_N^{-k} y_k[n-2] + x[n-1])(-W_N^k)$

Sabiranjem 1 i 2

$$y_k[n] + y_k[n-1](-W_N^k) = W_N^{-k} y_k[n-1] + x[n] + (W_N^{-k} y_k[n-2] + x[n-1])(-W_N^k)$$

$$y_k[n] = (W_N^{-k} + W_N^k) y_k[n-1] - y_k[n-2] + x[n] - W_N^k x[n-1]$$

$$y_k[n] = 2\cos\left(k\frac{2\pi}{N}\right) y_k[n-1] - y_k[n-2] + x[n] - W_N^k x[n-1]$$

$$H(z) = \frac{1 - W_N^k z^{-1}}{1 - 2\cos\left(k\frac{2\pi}{N}\right) z^{-1} + z^{-2}}$$

Mogli i ovako $H(z) = \frac{1}{1 - W_N^{-k} z^{-1}} = \frac{1 - W_N^k z^{-1}}{1 - W_N^k z^{-1}} \cdot \frac{1}{1 - W_N^{-k} z^{-1}} = \frac{1 - W_N^k z^{-1}}{1 - 2\cos\left(k\frac{2\pi}{N}\right) z^{-1} + z^{-2}}$

7

$$H(z) = \frac{1 - W_N^k z^{-1}}{1 - 2\cos\left(k\frac{2\pi}{N}\right) z^{-1} + z^{-2}} = \frac{1}{1 - 2\cos\left(k\frac{2\pi}{N}\right) z^{-1} + z^{-2}} \cdot \left(1 - W_N^k z^{-1}\right) = H_1(z) \cdot H_2(z) \cdot$$

Računamo u dva koraka

$H_1(z) \Rightarrow v_k[n] = 2\cos\left(k\frac{2\pi}{N}\right) v_k[n-1] - v_k[n-2] + x[n]$

$H_2(z) \Rightarrow y_k[n] = v_k[n] - W_N^k v_k[n-1]$

$v_k[-2] = v_k[-1] = 0$

Ovakvim preuređivanjem jednačine postignuto je da se nerekurzivni deo algoritma, izračunava samo jednom u trenutku $n = N$. U svakoj iteraciji rekurzivnog dela izvodi se jedno množenje i dva sabiranja. Ako je ulazna sekvenca $x[n]$ kompleksna, za izračunavanje jednog DFT odbirka $X[k]$, potrebno je $2(N + 2)$ realnih množenja i $4(N + 1)$ realnih sabiranja, odnosno, broj množenja je smanjen približno dva puta u odnosu na direktni metod. Ako je ulazna sekvenca $x[n]$ realna, broj realnih množenja iznosi $N + 1$, dok je broj realnih sabiranja $2N$. Zbog simetrije se u tom slučaju istovremeno određuje i $X[N - k]$.

8

Gercelov algoritam ima još neke prednosti. Za određivanje $X[k]$ potrebno je pamtiti samo vrednosti trigonometrijskih funkcija $\cos(2\pi k/N)$ i $\sin(2\pi k/N)$, koje se, ukoliko je potrebno više odbiraka, mogu i rekurzivno izračunavati.

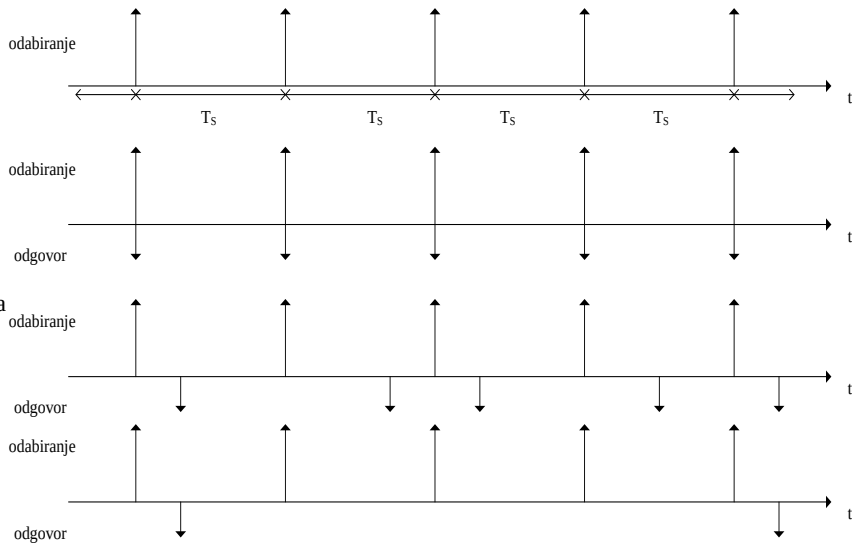
Druga prednost Gercelovog algoritma je što se rekurzivni deo izvodi stalno, a nerekurzivni deo samo po jednom na kraju algoritma. To je veoma pogodno za primene u realnom vremenu, gde se ne može čekati kraj sekvence da bi se započela njena obrada.

Sa porastom broja DFT odbiraka koje treba izračunati, M, Gercelov algoritam ostaje približno dvostruko efikasniji od direktnog algoritma, ali broj operacija raste srazmerno proizvodu MN . Znači, Gercelov algoritam je pogodan za izračunavanje malog broja DFT odbiraka, tj. kada je $M << N$, ili preciznije $M < \log_2 N$. Za izračunavanje većeg broja DFT odbiraka pogodniji su FFT algoritmi kod kojih je broj operacija srazmeran sa $N \log_2 N$, a koji će biti detaljnije opisani u izlaganjima koja slede.

9

RAD U REALNOM VREMENU – Potreba za „efikasnim“ računanjem

„Odgovor“ najkasnije u definisanom vremenskom intervalu.



Idealno
Vreme za proračun je nula

Realno - posle svakog odmerka
Vreme za proračun $< T_s$

Realno – posle n odmeraka
Vreme za proračun
po odmerku $< T_s$

10

FFT
Fast Fourier Transform

11

Periodičnost i simetričnost DFT koeficijenata može biti iskorišćena u još većoj meri za redukciju broja aritmetičkih operacija ako se primeni princip dekompozicije.

Pri direktnom izračunavanju DFT broj aritmetičkih operacija srazmeran sa $4N^2$.
Ako se ulazna sekvenca podeli na dve parcijalne sekvence, i za svaku od njih odredi DFT direktnom metodom, ukupan broj operacija množenja iznosi $2 \cdot \left(4 \cdot \left(\frac{N}{2}\right)^2\right) = 2N^2$, odnosno upola je manji nego za direktno izračunavanje DFT cele ulazne sekvence.

Naravno, dobijeni rezultati u oba slučaja nisu identični, pa je zato potrebno izvršiti dodatne aritmetičke operacije nad parcijalnim DFT da bi se dobio ispravan rezultat. Ukoliko je broj dodatnih množenja znatno manji od $2N^2$ dobija se značajna ušteda u broju potrebnih množenja, ako je dužina sekvence velika.

Princip dekompozicije se može dalje primeniti na parcijalne sekvence, sve dok se ne dođe do sekvence koja se ne može više deliti.

Na principu dekompozicije su zasnovani svi tzv. brzi algoritmi za izračunavanje DFT, poznati pod opštim nazivom Brza Furijeova Transformacija (Fast Fourier Transform).

12

FFT algoritam za $N = 2^p$ sa preuređivanjem u vremenu (DIT – decimation in time)
(radix 2)

Pretpostavimo da je dužina ulazne sekvence $N = 2^p$. Ovo ograničenje nije strogo, jer se sekvenca uvek može dopuniti potrebnim brojem nula, a znatno doprinosi jednostavnosti izvođenja i implementacije algoritma zbog toga što omogućava više uzastopnih podela sekvence na dve podsekvence iste dužine.

Na primer, koeficijenti kojima se množe članovi ulazne sekvence sa parnim indeksima isti su za članove sa indeksima k i $k + (N\ 2)$, gde je $0 \le k \le (N\ 2) - 1$. Za neparne vrednosti indeksa k koeficijenti imaju istu apsolutnu vrednost ali su suprotnog znaka. Zbog toga se ulazna sekvenca može razbiti na dve parcijalne sekvence: $x_{10}[n]$ koju čine elementi sa parnim indeksima i $x_{11}[n]$ koju čine elementi sa neparnim indeksima.

$$X[k] = \sum_{n=0}^{N-1} x[n]W_N^{kn} = \sum_{n=2i} x[n]W_N^{kn} + \sum_{n=2i+1} x[n]W_N^{kn}$$

Periodičnost i simetričnost rotirajućih faktora

$$W_N^{kn} = W_N^{k(n+N)} \qquad W_N^{k+\frac{N}{2}} = W_N^{k+\frac{N}{2}+\frac{N}{2}-\frac{N}{2}} = -W_N^k$$

$$W_N^2 = W_{N/2}^1 \qquad W_N^{k(N-n)} = W_N^{-kN} = \left(W_N^{kN}\right)^*$$

13

$$X[k] = \sum_{n=0}^{N-1} x[n]W_N^{kn} = \sum_{n=2i} x[n]W_N^{kn} + \sum_{n=2i+1} x[n]W_N^{kn}$$

$$X[k] = \sum_{i=0}^{N/2-1} x_{10}[i]W_N^{2ki} + \sum_{i=0}^{N/2-1} x_{11}[i]W_N^{2ki+k} = \sum_{i=0}^{N/2-1} x_{10}[i]W_{N/2}^{ki} + W_N^k \sum_{i=0}^{N/2-1} x_{11}[i]W_{N/2}^{ki}$$

Iz originalne sekvence sa parnim n — Iz originalne sekvence sa neparnim n

pojavljuju se dve sume koje predstavljaju DFT sekvenci $x_{10}[n]$ i $x_{11}[n]$ čija je dužina $N/2$ članova.
I ovo smo već radili kada smo gledali primenu DFTa.

$$X[k] = X_{10}[k] + W_N^k X_{11}[k]$$

Da li treba da računamo za $k = 0,1,\dots,N-1$ $X_{10}[k]$ i $X_{11}[k]$ ili je dovoljno upola manje $k = 0,1,\dots,N/2-1$

$$X\left[k + \frac{N}{2}\right] = X_{10}\left[k + \frac{N}{2}\right] + W_N^{k+N/2} X_{11}\left[k + \frac{N}{2}\right]$$

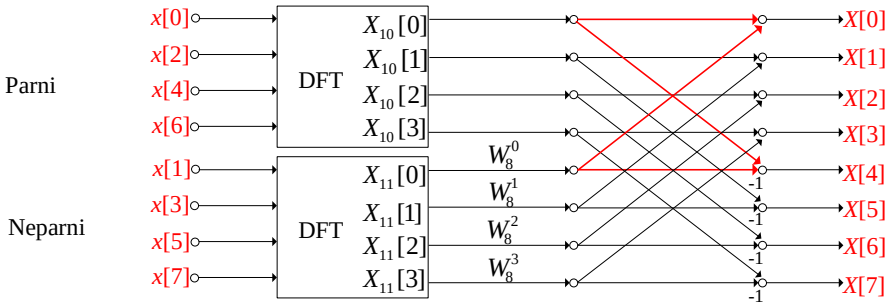
$$X\left[k + \frac{N}{2}\right] = X_{10}[k] - W_N^k X_{11}[k]$$

$X_{10}[k]$ i $X_{11}[k]$ su periodični sa periodom $N/2$

14

$$X[k] = X_{10}[k] + W_N^k X_{11}[k] \qquad k = 0,1,\dots,N/2-1$$

$$X\left[k+\frac{N}{2}\right] = X_{10}[k] - W_N^k X_{11}[k] \qquad k = 0,1,\dots,N/2-1$$



Izlazna sekvenca dobra

„Preuređivanje ulazne sekvence“

Strelice putanja računanja.
Više strelice ulaze u čvor -> sabiranje.
Koeficijent na strelici -> množenje.

Leptir operacija $X[k]$ i $X[k+N/2]$ na osnovu $X_{10}[k]$ i $X_{11}[k]$

Direktno računanje DFTa

1. Broj izračunavanja trigonometrijskih funkcija – $2N^2$(za svako k i n, sin i cos)
2. Broj realnih množenja – $4N^2$
3. Broj realnih sabiranja –$2N(2N-1) \approx 4N^2$

Podela na dve sekvence

Uočava se da u leptir operaciji postoji samo jedno kompleksno množenje i 2 kompleksna sabiranja, odnosno 4 realna množenja i 6 realnih sabiranja. Množenje sa eksponencijalnim faktorom W_N^k menja samo argument kompleksnog broja $X_{11}[k]$, pa se W_N^k naziva *rotacioni faktor*. U leptiru se takođe izračunavaju i dve trigonometrijske funkcije.

Pošto je za izračunavanje svih DFT odbiraka sekvence od $N/2$ tačaka potrebno $4(N/2)^2$ množenja, ukupno je potrebno $2N^2$ množenja za izračunavanje sekvenci $X_{10}[k]$ i $X_{11}[k]$ i dodatnih $4(N/2) = 2N$ množenja za kombinovanje sekvenci $X_{10}[k]$ i $X_{11}[k]$.

Za izračunavanje DFT podelom sekvence na parni i neparni deo ukupno je potrebno $2N(N+1)$ realnih množenja, što je manje od $4N^2$ ako je $N > 2$.

Pošto je po pretpostavci dužina ulazne sekvence $N = 2^p$, sekvence $x_{10}[n]$ i $x_{11}[n]$ imaju paran broj elemenata pa se mogu, u cilju daljeg smanjenja broja operacija, ponovo podeliti na parcijalne sekvence dužine $N/4$ po osnovu parnosti indeksa.

$$X_{10}[k] = \sum_{i=0}^{N/4-1} x_{10}[2i]W_{N/2}^{2ki} + \sum_{i=0}^{N/4-1} x_{10}[2i+1]W_{N/2}^{2ki+k} = \sum_{i=0}^{N/4-1} x_{20}[i]W_{N/4}^{ki} + W_N^{2k}\sum_{i=0}^{N/4-1} x_{21}[i]W_{N/2}^{ki}$$
$$= X_{20}[k] + W_N^{2k}X_{21}[k]$$

$$X_{11}[k] = \sum_{i=0}^{N/4-1} x_{11}[2i]W_{N/2}^{2ki} + \sum_{i=0}^{N/4-1} x_{11}[2i+1]W_{N/2}^{2ki+k} = \sum_{i=0}^{N/4-1} x_{22}[i]W_{N/4}^{ki} + W_N^{2k}\sum_{i=0}^{N/4-1} x_{23}[i]W_{N/2}^{ki}$$
$$= X_{22}[k] + W_N^{2k}X_{23}[k]$$

$$X_{10}\left[k + \frac{N}{4}\right] = X_{20}[k] - W_N^{2k}X_{21}[k]$$

$$X_{11}\left[k + \frac{N}{4}\right] = X_{22}[k] - W_N^{2k}X_{23}[k]$$

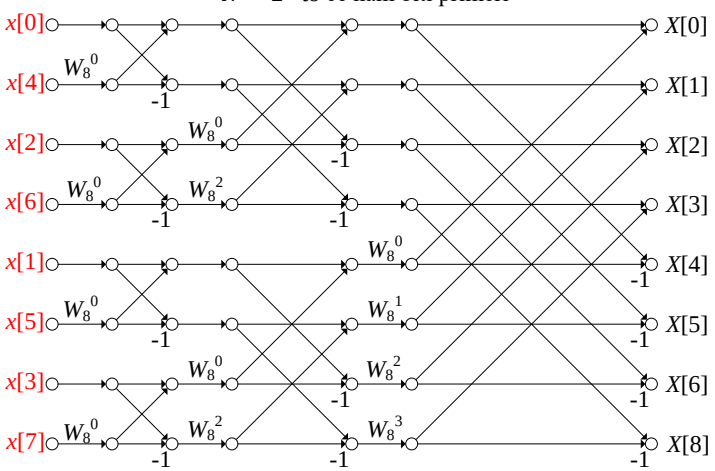
Može i dalje. Dok ne dođemo do jedinične sekvence. $N = 2^p$. Delimo p puta.

$$X[k] = X_{10}[k] + W_N^k X_{11}[k]$$
$$X_{10}[k] = X_{20}[k] + W_N^{2k} X_{21}[k]$$
$$X_{20}[k] = X_{30}[k] + W_N^{4k} X_{31}[k]$$
$$\dots$$

17

$N = 2^3$ to će nam biti primeri

Preuređeni podaci



Korektan redosled

Ukupno $p = \log_2 N$ stepeni

Ukupno $N_B = \frac{N}{2}\log_2 N$ leptir operacija

18

Potrebnih operacija

$$N_M = 2N\log_2 N$$

$$N_A = 3N\log_2 N$$

$$N_T = N\log_2 N$$

Primer N=4096 potreban broj množenja
Direktno 33554432
FFT 49152
Odnos 49152/33554432=0.00146=0.146%

		DFT			DIT FFT		
p	N	N_T	N_M	N_A	N_T	N_M	N_A
2	4	32	64	48	8	16	24
3	8	128	256	224	24	48	72
4	16	512	1024	960	64	128	192
5	32	2048	4096	3968	160	320	480
6	64	8192	16384	16128	384	768	1152
7	128	32768	65536	65024	896	1792	2688
8	256	131072	262144	261120	2048	4096	6144
9	512	524288	1048576	1046528	4608	9216	13824
10	1024	2097152	4194304	4190208	10240	20480	30720
11	2048	8388608	16777216	16769024	22528	45056	67584
12	4096	33554432	67108864	67092480	49152	98304	147456

Uočite

U poslednjem p-tom stepenu nam trebaju rotacioni faktori W_N^0, W_N^1,…, $W_N^{N/2-1}$ „u donjoj polovini“

U pretposlednjem (p-1) stepenu nam trebaju rotacioni faktori $W_{N/2}^0$, $W_{N/2}^1$,…, $W_{N/2}^{N/4-1}$ -> W_N^0, W_N^2,…, $W_N^{N/2-2}$
*dva puta grupe „prazno, ima, prazno, ima“

…

U (p-k) stepenu nam trebaju rotacioni faktori $W_{N/2^k}^0$, $W_{N/2^k}^1$,…, $W_{N/2^k}^{N/2^{k+1}-1}$ -> W_N^0, $W_N^{2^k}$,…, $W_N^{N/2-2^k}$
*2^k puta grupe „prazno, ima, prazno, ima, …“

…

U 1. stepenu nam trebaju rotacioni faktori W_N^0 *2^{N-1} puta grupe „prazno, ima, prazno, ima, …“

Čest se javlja na ispitu da se nacrtaju dijagrami računanja FFTa preko leptira.
Pamtite po smislu a ne „napamet“

Da razjasnimo onda šta su nam bili oni indeksi

$$X[k] = \sum_{i=0}^{N/2-1} x_{10}[i]W_N^{2ki} + \sum_{i=0}^{N/2-1} x_{11}[i]W_N^{2ki+k} = \sum_{i=0}^{N/2-1} x_{10}[i]W_{N/2}^{ki} + W_N^k \sum_{i=0}^{N/2-1} x_{11}[i]W_{N/2}^{ki}$$

…

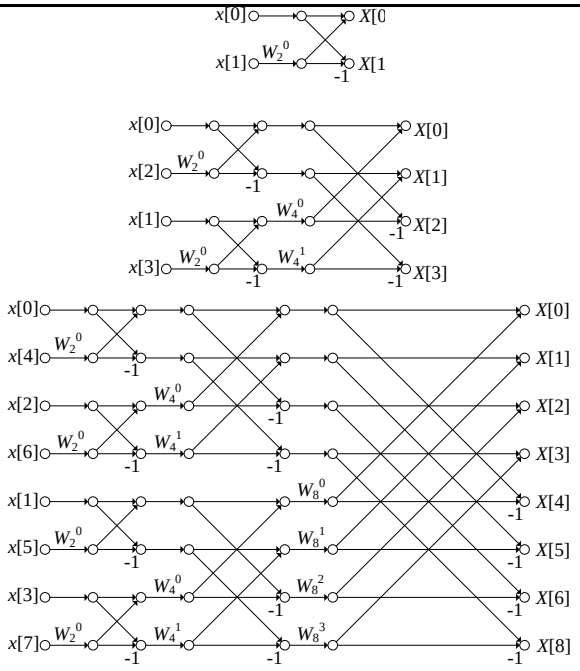
x_{10} sekvenca dobijena iz orginalne tako što su uzeti odmerci čiji indeksi deljenjem sa 2^1 daju ostatak 0 (parni)
x_{11} sekvenca dobijena iz orginalne tako što su uzeti odmerci čiji indeksi deljenjem sa 2^1 daju ostatak 1 (neparni)
x_{20} sekvenca dobijena iz orginalne tako što su uzeti odmerci čiji indeksi deljenjem sa 2^2 daju ostatak 0 (parni parni)
x_{21} sekvenca dobijena iz orginalne tako što su uzeti odmerci čiji indeksi deljenjem sa 2^2 daju ostatak 1 (parni neparni)
x_{22} sekvenca dobijena iz orginalne tako što su uzeti odmerci čiji indeksi deljenjem sa 2^2 daju ostatak 2 (neparni parni)
x_{23} sekvenca dobijena iz orginalne tako što su uzeti odmerci čiji indeksi deljenjem sa 2^2 daju ostatak 3 (neparni neparni)

…

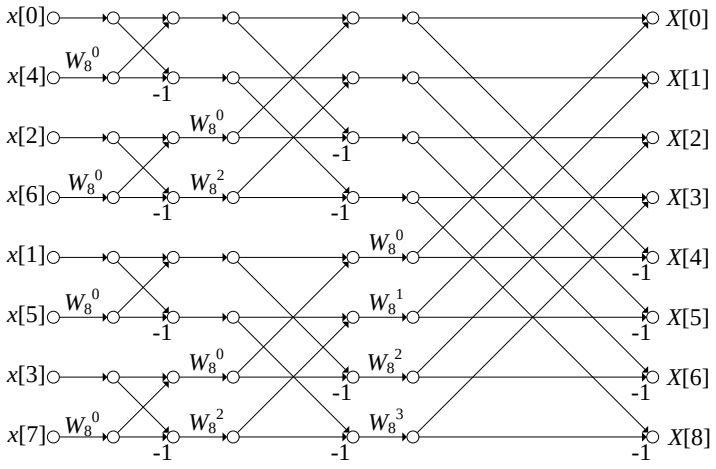
Kako smo onda mogli pamtiti redosled na dijagramu sa *N*=8
Idemo redom $x[0], x[1], \ldots, x[6], x[7]$
uzimamo prvo one koji deljenjem sa 4 daju ostatak 0,
zatim one koji deljenjem sa 4 daju ostatak 2
zatim one koji deljenjem sa 4 daju ostatak 1
zatim one koji deljenjem sa 4 daju ostatak 3

I ovo možemo generalizovati za bilo koje $N = 2^p$

Copy
Paste



N=8, da sredimo rotacione faktore

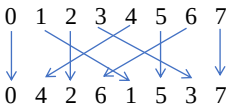


Na ulaz u algoritam moramo da dovedemo preuređenu sekvencu podataka

Preuređenje podataka - Bit inverzija

Redosled sekvence: 0 1 2 3 4 5 6 7
Redosled koji je potreban: 0 4 2 6 1 5 3 7

preuređenje



i tu postoji simetrija

Orginal	000	001	010	011	100	101	110	111
1.korak	000	010	100	110	001	011	101	111
	000	010	100	110	001	011	101	111
2.korak	000	100	010	110	001	101	011	111
Preuređena	000	100	010	110	001	101	011	111
Nova pozicija	000	001	010	011	100	101	110	111

$$b_{n-1}\ b_{n-2}b_{n-3}\ldots b_1\ b_0 \xrightarrow{ide\ na\ mesto} b_0\ b_1b_2\ldots b_{n-2}\ b_{n-1}$$

Pošto se u opisanom algoritmu preuređuje ulazna sekvenca, tj, preuređivanje se vrši u vremenskom domenu, ovakav FFT algoritam naziva se *FFT algoritam sa preuređivanjem u vremenu*. U literaturi na engleskom jeziku ovaj algoritam naziva *decimation-in-time* (*DIT*) *algoritam*, pa se i u našoj terminologiji odomaćio naziv *decimacija* (*desetkovanje*) *u vremenu*. Kako se u poslednje vreme u literaturi iz obrade signala pod decimacijom uglavnom podrazumeva redukcija broja odbiraka, pravilnije je i pogodnije koristiti naziv algoritam sa preuređivanjem u vremenu, ili kraće, DIT algoritam.

$$b_{n-1}\, b_{n-2} b_{n-3} \dots b_1\, b_0 \xrightarrow{ide\ na\ mesto} b_0\, b_1 b_2 \dots b_{n-2}\, b_{n-1}$$

$$b_{n-1}2^{n-1} + b_{n-2}2^{n-2} + \cdots + b_1\, 2^1 + b_0 2^0 \xrightarrow{ide\ na\ mesto} b_0 2^{n-1} + \ b_1 2^{n-2} + \cdots + b_{n-2}\, 2^1 + b_{n-1}2^0$$

Ovako smo delili
if (b(i)==0) ide u "donju polovinu indeksa" X_{10}
else ide u "gornju polovinu indeksa" X_{11}

```
nova_pozicija = 0;
pomeraj =2^(n-1);
for (i = 0, i<n, i++){
        if (b(i) ==1) nova_pozicija += pomeraj;
        pomeraj=pomeraj/2;}
```

25

Da razjasnimo onda šta su nam bili oni indeksi

$$X[k] = \sum_{i=0}^{N/2-1} x_{10}[i] W_N^{2ki} + \sum_{i=0}^{N/2-1} x_{11}[i] W_N^{2ki+k} = \sum_{i=0}^{N/2-1} x_{10}[i] W_{N/2}^{ki} + W_N^k \sum_{i=0}^{N/2-1} x_{11}[i] W_{N/2}^{ki}$$

…

x_{10} sekvenca dobijena iz orginalne tako što su uzeti odmerci čiji indeksi deljenjem sa 2^1 daju ostatak 0 (parni)
x_{11} sekvenca dobijena iz orginalne tako što su uzeti odmerci čiji indeksi deljenjem sa 2^1 daju ostatak 1 (neparni)
x_{20} sekvenca dobijena iz orginalne tako što su uzeti odmerci čiji indeksi deljenjem sa 2^2 daju ostatak 0 (parni parni)
x_{21} sekvenca dobijena iz orginalne tako što su uzeti odmerci čiji indeksi deljenjem sa 2^2 daju ostatak 1 (parni neparni)
x_{22} sekvenca dobijena iz orginalne tako što su uzeti odmerci čiji indeksi deljenjem sa 2^2 daju ostatak 2 (neparni parni)
x_{23} sekvenca dobijena iz orginalne tako što su uzeti odmerci čiji indeksi deljenjem sa 2^2 daju ostatak 3 (neparni neparni)

…

Kako smo onda mogli pamtiti redosled na dijagramu sa N=8
Idemo redom $x[0], x[1], \dots, x[6], x[7]$
uzimamo prvo one koji deljenjem sa 4 daju ostatak 0,
zatim one koji deljenjem sa 4 daju ostatak 2
zatim one koji deljenjem sa 4 daju ostatak 1
zatim one koji deljenjem sa 4 daju ostatak 3

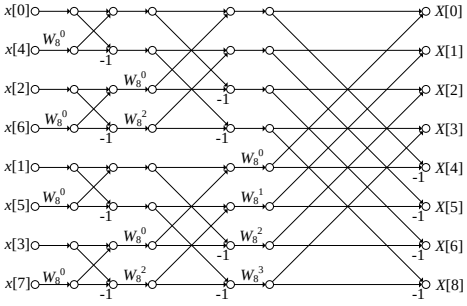
I ovo možemo generalizovati za bilo koje $N = 2^p$

26

Potrebna memorija za izračunavanje DFT

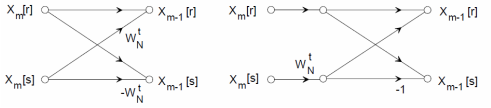
Ulazna sekvenca kompleksna
Broj potrebnih lokacija $N+N$
Izlazna je "sigurno" kompleksna
Broj potrebnih lokacija $N+N$
Sve zajedno $4N$

Ne deluje puno gledajući primere koje smo mi radili N=32. Međutim kako tehnologija napreduje ako rastu i „apetiti". Nije neuobičajeno da radimo sa $N = 2^{20}$ tačaka (1M) ili čak sa $N = 2^{30}$ (1G) tačaka. Povećamo memoriju, porastu „apetiti" pa nam opet postaje problem manjak memorije itd… Pogotovo ako ove algoritme realizujemo hardverski.

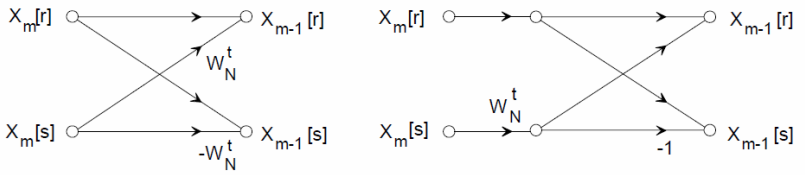


$$X_{m-1}[r] = X_m[r] + W_N^t X_m[s]$$

$$X_{m-1}[s] = X_m[r] - W_N^t X_m[s]$$



27



Dakle, u bilo kom stepenu izvršavanja algoritma, podaci čiji su indeksi r i s potrebni su za izračunavanje samo dva nova podatka na istim lokacijama. Ako se izračunavanje obavlja kolonu po kolonu, $2N$ memorijskih lokacija u kojima se nalazi ulazna sekvenca može se koristiti za smeštanje međurezultata i izlazne DFT sekvence. Ustvari, potrebna je još samo jedna (dve zbog kompleksnih vrednosti) dodatna memorijska lokacija za smeštanje međurezultata kod izvršavanja svake od leptir operacija. Ovakav efikasan postupak za korišćenje memorije naziva se *izračunavanje u mestu* (engl. *in-place computation*) i veoma je važan kada je memorijski prostor ograničen

$$P = X_m[s]W_N^t \quad vrednost X_m[s]\ nije\ više\ potrebna$$

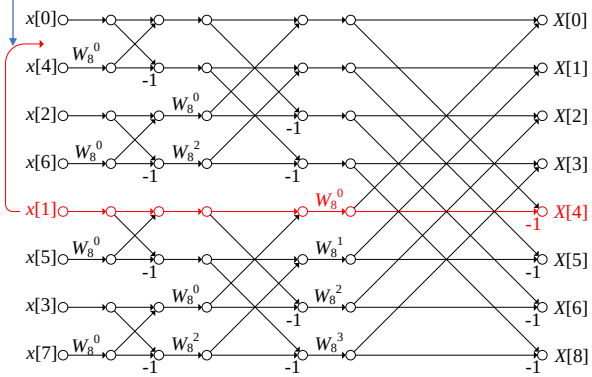
$$X_m[r] - P \xrightarrow{na\ memorijsku\ lokaciju} X_m[s]$$

$$X_m[r] + P \xrightarrow{na\ memorijsku\ lokaciju} X_m[r]$$

28

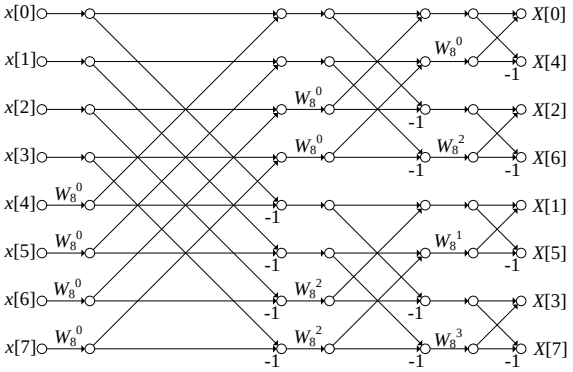
Modifikacije DIT algoritma
(modifikacija grafa)

Povlačimo celu horizontalnu liniju na gore, tako da ulazni podatak zauzme svoju pravu poziciju. Ne diramo čvorove, koeficijente …
I tako redom …



Ulazna sekvenca u „prirodnom“ redosledu

Izlazna sekvenca u „bit inverznom“ redosledu

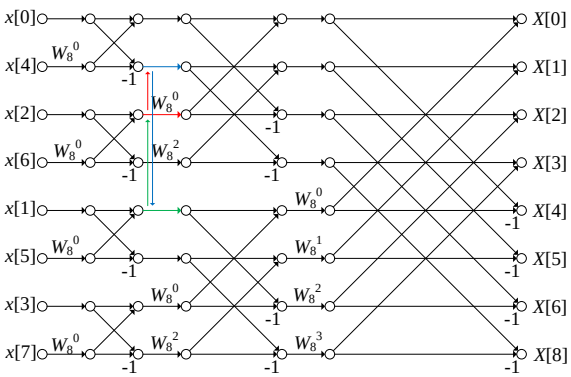


Moguće „in place“

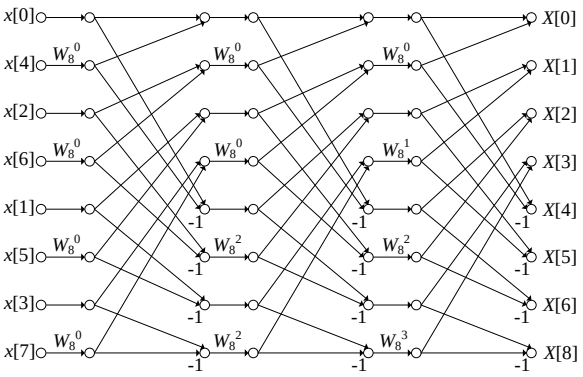
29

Modifikacije DIT algoritma
(modifikacija grafa)

„menjamo mesto“ samo putanjama između leptira



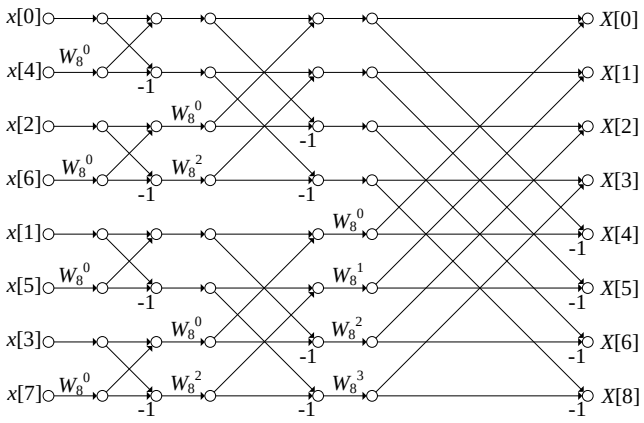
Ista geometrija leptira



Nije moguće direktno „in place“

30

Pseudo kod



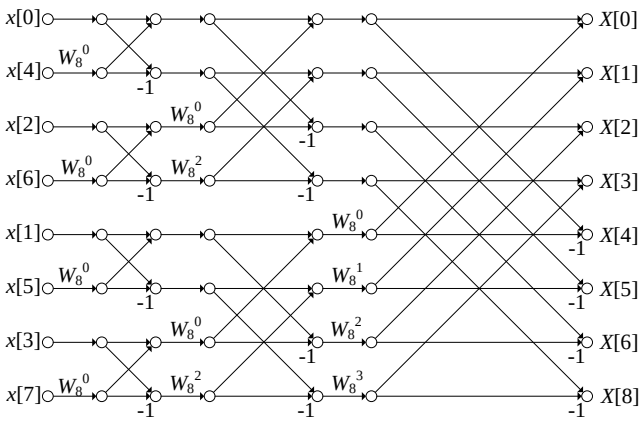
Može u okviru višestrukih petlji

```
s=3;
n=2^s;
for (i=0, i<n/2, i++){W(i)=cos(2*pi*i/n)+j*sin(2*pi*i/n);}

/* 1.stepen*/
blok=2;
pomerajW=n/blok;
for (i=0, i<n, i+=blok){
        for (j = 0, j<blok/2, j++{
                pom=x(i+j+blok/2)*W(J*pomerajW);
                x(i+j+blok/2)=x(i+j)-pom;
                x(i+j)=x(i+j)+pom;
        }
}
/*2. stepen*/
blok=blok*2;
pomerajW=n/blok;
for (i=0, i<n, i+=blok){
        for (j = 0, j<blok/2, j++{
                pom=xi(i+j+blok/2)*W(J*pomerajW);
                x(i+j+blok/2)=x(i+j)-pom;
                x(i+j)=x(i+j)+pom;
        }
}
/*3. stepen*/
blok=blok*2;
pomerajW=n/blok;
for (i=0, i<n, i+=blok){
        for (j = 0, j<blok/2, j++{
                pom=x(i+j+blok/2)*W(J*pomerajW);
                x(i+j+blok/2)=x(i+j)-pom;
                x(i+j)=x(i+j)+pom;
        }
}
```


31

Pseudo kod



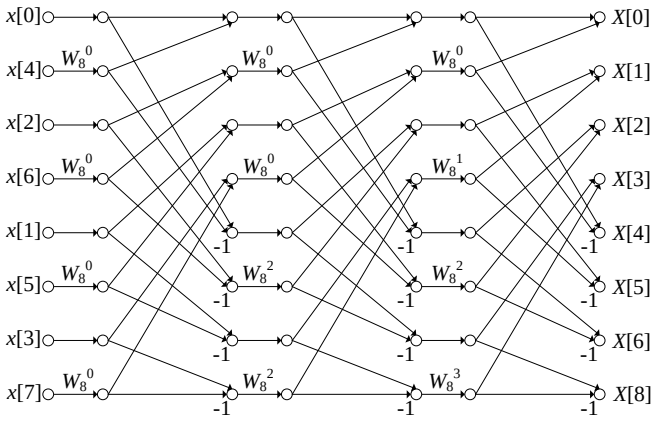

```
s=3;
n=2^s;
for (i=0, i<n/2, i++){W(i)=cos(2*pi*i/n)+j*sin(2*pi*i/n);}

for (k=1, k<s+1, k++){
        blok=2^k;
        pomerajW= 2^(s-k);

        for (i=0, i<n, i+=blok){
                for (j = 0, j<blok/2, j++){
                        pom=x(i+j+blok/2)*W(J*pomerajW);
                        x(i+j+blok/2)=x(i+j) -pom;
                        x(i+j)=x(i+j)+pom;
                }
        }
}
```


32

Pseudo kod




```
s=3;
n=2^s;
for (i=0, i<n/2, i++){W(i)=cos(2*pi*i/n)+j*sin(2*pi*i/n);}

for (j=0, j<s, j++){
        blok= 2^(s-j);
        pomerajW=blok/2; /*n/2,n/4,n/8 …*/

        for (i=0, i<n, i+=2){
                indexW=int(i/blok)*pomerajW;
                x(i+1)=x(i+1)* W(indexW);
        }

        for (i=0, i<n, i+=2){
                y(i/2)=x(i)+x(i+1);
                y(i/2+n/2)= x(i)-x(i+1);
        }

        for (i=0, i<n, i++){x(i)=y(i);}
}
```

„lako“ za hardverske realizacije


```
k=3;
n=2^k;
for (i=0, i<n/2, i++){W(i)=cos(2*pi*i/n)+j*sin(2*pi*i/n);}

for (l=1, l<k+1, l++){
        blok=2^l;
        pomerajW=n/blok;

        for (i=0, i<n, i+=blok){
                for (j = 0, j<blok/2, j++){
                        pom=x(i+j+blok/2)*W(J*pomerajW);
                        x(i+j+blok/2)=x(i+j) -pom;
                        x(i+j)=x(i+j)+pom;
                }
        }
}
```

VS

```
k=3;
n=2^k;
for (i=0, i<n/2, i++){W(i)=cos(2*pi*i/n)+j*sin(2*pi*i/n);}

for (j=0, j<k, j++){
        blok= 2^(k-j);
        pomerajW=blok/2; /*n/2,n/4,n/8 …*/

        for (i=0, i<n, i+=2){
                indexW=int(i/blok)*pomerajW;
                x(i+1)=x(i+1)* W(indexW);
        }

        for (i=0, i<n, i+=2){
                y(i/2)=x(i)+x(i+1);
                y(i/2+n/2)= x(i)-x(i+1);
        }

        for (i=0, i<n, i++){x(i)=y(i);}
}
```

„dužina petlje promenljiva, inkrement indeksa promenljiv“

„fiksne petlje, fiksni indeksi“

FFT algoritam $N = 2^p$ sa preuređivanjem u frekvencijskom domenu (DIF – decimation in frequency)

$$X[k] = \sum_{n=0}^{N-1} x[n] W_N^{kn}$$

$$X[2k] = \sum_{n=0}^{N-1} x[n] W_N^{2kn} = \sum_{n=0}^{N/2-1} x[n] W_N^{2kn} + \sum_{n=N/2}^{N-1} x[n] W_N^{2kn} = \sum_{n=0}^{N/2-1} x[n] W_N^{2kn} + \sum_{n=0}^{N/2-1} x\left[n+\frac{N}{2}\right] W_N^{2k\left(n+\frac{N}{2}\right)}$$

Periodičnost i simetričnost rotirajućih faktora

$$W_N^{kn} = W_N^{k(n+N)} \qquad W_N^{k+\frac{N}{2}} = W_N^{k+\frac{N}{2}+\frac{N}{2}-\frac{N}{2}} = -W_N^k$$

$$W_N^2 = W_{N/2}^1 \qquad W_N^{k(N-n)} = W_N^{-kN} = \left(W_N^{kN}\right)^*$$

$$W_N^{2k\left(n+\frac{N}{2}\right)} = W_N^{2kn} W_N^{2k\frac{N}{2}} = W_N^{2kn}$$

$$X[2k] = \sum_{n=0}^{N/2-1} \left(x[n] + x\left[n+\frac{N}{2}\right]\right) W_N^{2kn} = \sum_{n=0}^{N/2-1} \left(x[n] + x\left[n+\frac{N}{2}\right]\right) W_{N/2}^{kn}$$

$$X[2k+1] = \sum_{n=0}^{N-1} x[n] W_N^{(2k+1)kn} = \sum_{n=0}^{N/2-1} x[n] W_N^{(2k+1)n} + \sum_{n=N/2}^{N-1} x[n] W_N^{(2k+1)n}$$
$$= \sum_{n=0}^{N/2-1} x[n] W_N^{(2k+1)n} + \sum_{n=0}^{N/2-1} x\left[n+\frac{N}{2}\right] W_N^{(2k+1)\left(n+\frac{N}{2}\right)}$$

$$W_N^{(2k+1)\left(n+\frac{N}{2}\right)} = W_N^{2kn} W_N^{2k\frac{N}{2}} W_N^{n+\frac{N}{2}} = -W_N^{2kn} W_N^n$$

$$X[2k+1] = \sum_{n=0}^{N/2-1} \left(x[n] - x\left[n+\frac{N}{2}\right]\right) W_N^{2kn} W_N^n = \sum_{n=0}^{N/2-1} \left(x[n] - x\left[n+\frac{N}{2}\right]\right) W_{N/2}^{kn} W_N^n$$

$$X[2k] = \sum_{n=0}^{N/2-1} \left(x[n] + x\left[n+\frac{N}{2}\right]\right) W_{N/2}^{kn}$$

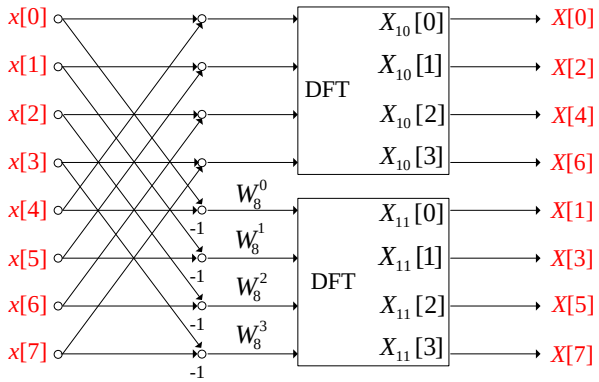
$$X[2k+1] = \sum_{n=0}^{N/2-1} \left(x[n] - x\left[n+\frac{N}{2}\right]\right) W_{N/2}^{kn} W_N^n$$

DFT sekvence $x_{10}[n]$ od $N/2$ članova koja se dobija sabiranjem odgovarajućih članova prve i druge polovine ulazne sekvence

DFT sekvence $x_{20}[n]$ od $N/2$ članova, koja se dobija oduzimanjem odgovarajućih članova iz druge polovine od članova prve polovine ulazne sekvence i množenjem dobijene razlike sa W_N

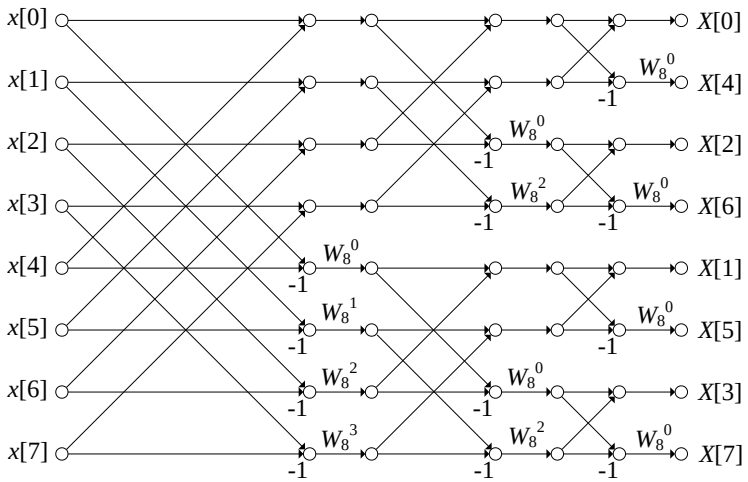
$$X[2k] = \sum_{n=0}^{N/2-1} \left(x[n] + x\left[n+\frac{N}{2}\right]\right) W_{N/2}^{kn} = \sum_{n=0}^{N/2-1} x_{10}[n] W_{N/2}^{kn} \qquad x_{10}[n] = x[n] + x\left[n+\frac{N}{2}\right]$$

$$X[2k+1] = \sum_{n=0}^{N/2-1} \left(x[n] - x\left[n+\frac{N}{2}\right]\right) W_{N/2}^{kn} W_N^n = \sum_{n=0}^{N/2-1} x_{11}[n] W_{N/2}^{kn} \qquad x_{11}[n] = x[n] - x\left[n+\frac{N}{2}\right] W_N^n$$



37

Na isti način kao kod DIFa, daljom podelom sekvenci na primeru $N = 2^3 = 8$

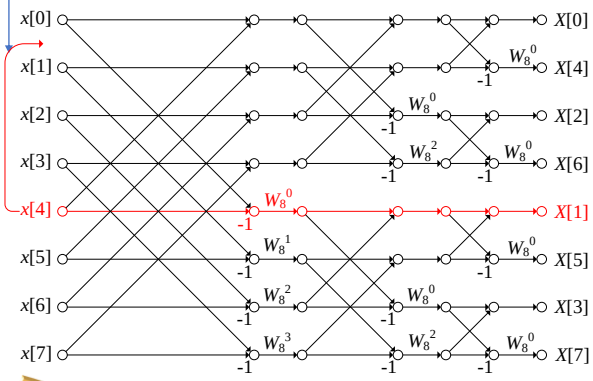


Moguće „in place“

38

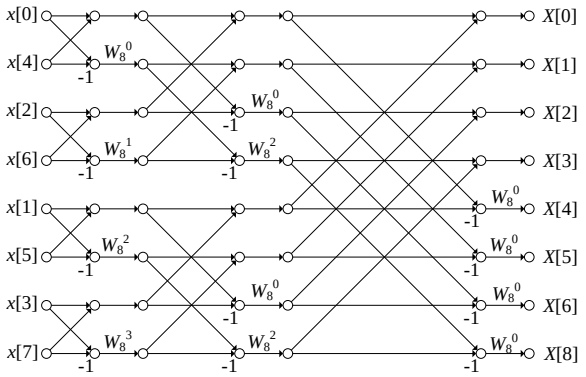
Modifikacije DIF algoritma
(modifikacija grafa)

Povlačimo celu horizontalnu liniju na gore, tako da izlazni podatak zauzme svoju pravu poziciju. Ne diramo čvorove, koeficijente …
I tako redom …



Ulazna sekvenca u
„bit inverznom“ redosledu

Izlazna sekvenca
u „prirodnom“ redosledu

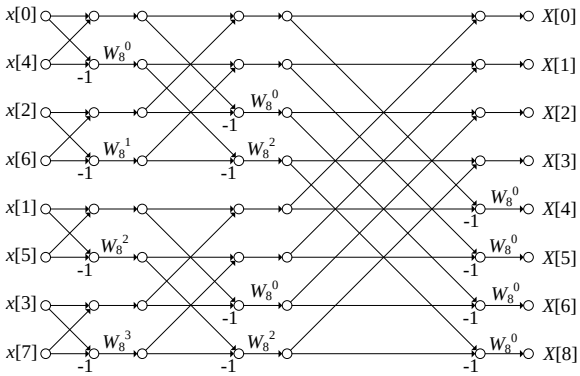


Moguće „in place“

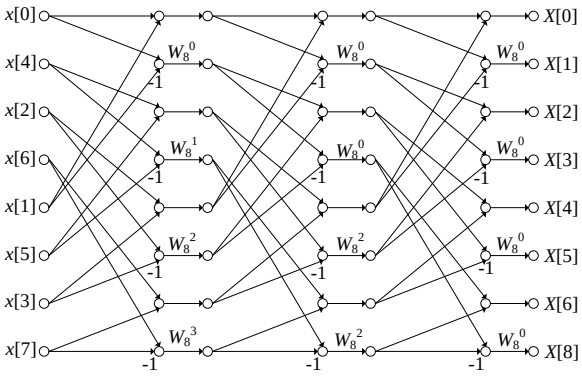
39

Modifikacije DIF algoritma
(modifikacija grafa)

„menjamo mesto“ samo putanjama između leptira



Ista geometrija leptira

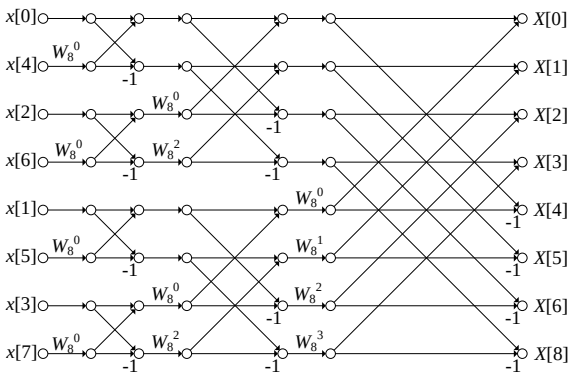


Nije moguće direktno „in place“

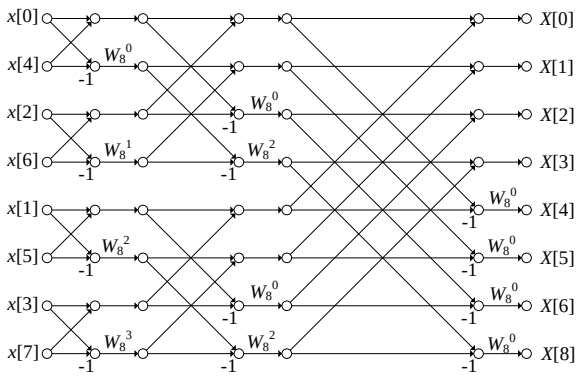
40

VS

DIT



DIF

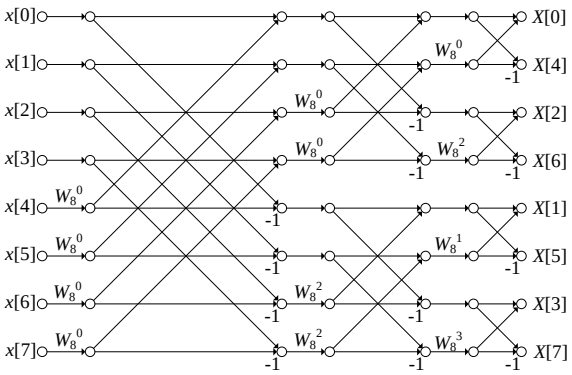


Konstrukcija leptira – rotirajući faktori
Računanje „isto“

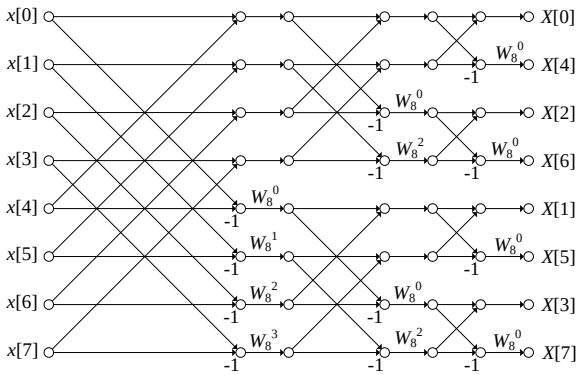
41

VS

DIT



DIF



Konstrukcija leptira – rotirajući faktori
Računanje „isto“

42

FFT algoritmi sa preuređivanjem u vremenu i po frekvenciji koji su opisani bili su razvijeni za dužinu sekvence $N = 2^p$ pa se stoga nazivaju *algoritmi sa osnovom* 2, ili, prema engleskoj terminologiji, *radiks*-2 *algoritmi*.

U literaturi o efikasnom izračunavanju DFT poznati su i drugačiji načini dekompozicije. Na primer, sekvenca se može deliti na četiri podsekvence (ako je $N = 4^p$) ili na osam podsekvenci (ako je $N = 8^p$). Kako se time mogu postići izvesne uštede u broju izračunavanja, interesantne su i klase *radiks*-4 i *radiks*-8 algoritama.
Izlaze van okvira ovog kursa.

U prethodnom izlaganju dužina sekvence čija se DFT određuje bila je ograničena uslovom $N = r^p$, (R = 2,4,8).
Ovakvo ograničenje omogućilo je jednostavnu dekompoziciju i razvoj efikasnih FFT algoritama. Međutim, u praksi postoje slučajevi kada ulazna sekvenca ne zadovoljava ograničenje $N = r^p$ i ne sme se proširivati nulama.
U literaturi o izračunavanju DFT razvijeni su postupci efikasnog izračunavanja koji su primenljivi na mnogo širi krug slučajeva, a mogu dovesti i do efikasnijih rešenja.
U osnovi svih metoda opet leži princip dekompozicije sekvence, samo što je taj princip u opštem slučaju zamenjen preslikavanjem jednodimenzionalne sekvence u višedimenzionalnu sekvencu.
Izlaze van okvira ovog kursa.

Drugi primer je efikasno kombinovanja radiks-2 i radiks-4 algoritama i predstavlja
Algoritam sa dvostrukim radiksom (engl. *split-radix algorithm*)
Izlazi van okvira ovog kursa.