Szybka transformata Fouriera

Problem Chcemy pomnożyć dwa wielomiany A: $\sum_{j=0}^{n-1} a_j x^j$ oraz B: $\sum_{j=0}^{n-1} b_j x^j$ stopnia co najwyżej n.

O ile operację dodawania wielomianów w postaci współczynnikowej możemy przeprowadzać w czasie $\Theta(n)$, operacja mnożenia wielomianów wymaga przemnożenia każdego współczynnika wielomianu A przez każdy współczynnik wielomianu B. Obarczone jest to zatem kosztem $\Theta(n^2)$:

$$C(x) = A(x) \cdot B(x) = \sum_{j=0}^{2n-2} \left(\sum_{k=0}^{j} a_k b_{j-k} \right) x^j$$

O ile w postaci współczynnikowej operacja mnożenia jest trudna obliczeniowo (w przeciwieństwie do operacji dodawania), to w postaci par dwuelementowych, gdy wielomian $W(x)$ jest skojarzony z n parami $(x_i, W(x_i))$ takimi, że $i = 0, 1, \ldots, n-1$ operacje podstawowe wykonywać możemy bardzo prosto w czasie $\Theta(n)$:

$$C_+(x) = A(x) + B(x) \to \{(x_i, A(x_i) + B(x_i))\} \quad i = 0, 1, \ldots, n-1$$
$$C_\cdot(x) = A(x) \cdot B(x) \to \{(x_i, A(x_i) \cdot B(x_i))\} \quad i = 0, 1, \ldots, n-1$$

Operację zamiany postaci współczynnikowej na postać par nazywamy **ewaluacją**. Problemem tej postaci jest jednak fakt, że nieznana jest nam metoda obliczania wartości $C(x')$ dla x' będącego różnym od któregokolwiek z x skojarzonych aktualnie z wielomianem inna niż dokonanie ponownego przekształcenia wielomianu na postać współczynnikową (**interpolacji**) i ewaluacji wartości wielomianu w x'.

Zauważmy, że operacja mnożenia daje w wyniku wielomian o stopniu wyższym niż wielomiany wejściowe: $\deg(C) = \deg(A) + \deg(B)$. Aby interpolacja była określona jednoznacznie, musimy jej dostarczyć tyle punktów, ile wynosi spodziewany stopień wielomianu ($\deg(C)$). Musimy to uwzględnić w naszych rozważaniach i dla wielomianów A, B o równym stopniu n ewaluować $2n$ punktów przed przystąpieniem do dalszych operacji.

W nauczanej na analizie numerycznej postaci, ewaluacji n punktów wielomianu o stopniu $O(n)$ dokonać możemy w czasie $O(n^2)$ używając schematu Hornera. Interpolacji przy użyciu wzoru na wielomian interpolacyjny Lagrange'a umiemy dokonać dla n punktów w czasie $\Theta(n^2)$. Pomysł aby dokonać przekształcenia wielomianu do formy par, dokonać szybkiego mnożenia i wrócić do formy współczynnikowej wydaje się zatem nieefektywny. Ale czy na pewno?

Okazuje się, że dla pewnych określonych wartości x_i możemy transformacji pomiędzy dwoma formami zapisu wielomianów dokonać szybciej, w czasie $\Theta(n \log n)$, używając szybkiej transformaty Fouriera. Metoda ta opiera się na szczególnych właściwościach zespolonych pierwiastków z jedynki, które umożliwiają zastosowanie strategii *dziel i zwyciężaj*.

Zespolone pierwiastki z jedności Zespolony pierwiastek stopnia n z jedności to liczba ω taka, że $\omega^n = 1$. Istnieje dokładnie n pierwiastków z jedności stopnia n:

$$e^{2\pi ik/n} \text{ dla } k = 0, 1, \ldots, n-1.$$

Po podstawieniu do wzoru Eulera:

$$e^{iu} = \cos(u) + i\sin(u)$$

odkrywamy, że wzór ten ma ładną interpretację geometryczną z uwagi na to, że część rzeczywista jest niewidoczna z części zespolonej (a część zespolona z rzeczywistej) - jeśli jedną osią układu współrzędnych będzie $Re(x)$ a drugą $Im(x)$ otrzymamy punkty rozłożone równomiernie po okręgu jednostkowym, nanosząc na ten układ pierwiastki z jedności otrzymane ze wzoru $e^{2\pi ik/n}$.

Ponieważ $\omega_n^n = \omega_n^0 = 1$ (bo $e^{2\pi in/n} = e^{2\pi i} = 1 = e^0$ ze wzoru Eulera) to $\omega_n^j \omega_n^k = \omega_n^{j+k} = \omega_n^{(j+k) \mod n}$, pierwiastki n-tego stopnia z jedności formują grupę z mnożeniem o identycznej strukturze jak grupa $(\mathbb{Z}_n, +_n)$.

Przydatnymi do celów naszego zadania właściwościami tych pierwiastków są:

- skracanie: $\omega_{dn}^{dk} = \left(e^{2\pi i/dn}\right)^{dk} = \left(e^{2\pi i/n}\right)^k = \omega_n^k$
- połowienie: $\left(\omega_n^{k+n/2}\right)^2 = \omega_n^{2k+n} = \omega_n^{2k}\omega_n^n = \omega_n^{2k} = \left(\omega_n^k\right)^2$

Z połowienia wynika, że ω_n^k i $\omega_n^{k+n/2}$ mają ten sam kwadrat, co jest właściwością kluczową dla podejścia *dziel i zwyciężaj* (nietrudno zauważyć, że mamy $n/2$ rozróżnialnych kwadratów n pierwiastków n-tego stopnia z jedności będących $n/2$ pierwiastkami stopnia $n/2$ z jedności).

Załóżmy, że chcemy ewaluować wielomian $A(x) = \sum_{j=0}^{n-1} a_j x^j$ stopnia n w punktach będących kolejnymi pierwiastkami z jedności stopnia n. Wtedy wielomian w postaci par przyjmie postać:

$$A(x) \to \{(\omega_n^k, \sum_{j=0}^{n-1} a_j \omega_n^{kj} = y_k\} \quad i = 0, 1, \ldots, n-1$$

Wektor $y = (y_0, y_1, \ldots, y_{n-1})$ nazywamy **dyskretną transformatą Fouriera (DFT)** wektora $a = (a_0, a_1, \ldots, a_{n-1})$. Piszemy, że $y = \mathrm{DFT}_n(a)$.

Szybka transformata Fouriera implementuje strategię dziel i zwyciężaj, poprzez obliczanie mniejszych wielomianów, stopnia $n/2$, złożonych tylko ze współczynników a_i o i parzystych, albo nieparzystych:

$$E(x) = a_0 + a_2x + a_4x^2 + \cdots + a_{n-2}x^{n/2-1}$$
$$O(x) = a_1 + a_3x + a_5x^2 + \cdots + a_{n-1}x^{n/2-1}$$

... wtedy $A(x) = E(x^2) + xO(x^2)$, a problem redukuje się do:

- ewaluowania wielomianów o stopniu $n/2$ $E(x)$ i $O(x)$ w punktach:

  $$(\omega_n^0)^2, (\omega_n^1)^2, \cdots, (\omega_n^n - 1)^2$$

  (zauważmy, że mamy dokładnie tyle rozróżnialnych wartości pierwiastków ile potrzebujemy - punktów jest tak na prawdę $n/2$ z uwagi na to, że wartości jest $n/2$ z połowienia, a nie ma sensu ewaluować wielomianu dwa razy dla tej samej wartości)

- połączenia wielomianów poprzez $A(x) = E(x^2) + xO(x^2)$:
  $y_k = y_k^E + \omega y_k^O$
  $y_{k+n/2} = y_k^E - \omega y_k^O$

Każde z wywołań wykorzystuje $\Theta(n)$ dodatkowych operacji na łączenie wyników $O(x)$ i $E(x)$, zatem całkowity czas działania FFT wynosi $\Theta(n \log n)$.

Przekształcenie odwrotne Pamiętamy, że DFT możęmy zapisać jako iloczyn macierzy $y = V_n a$ gdzie V_n jest macierzą Vandermonda zawierającą odpowiednie potęgi ω_n:

$$\begin{pmatrix} y_0 \\ y_1 \\ y_2 \\ \vdots \\ y_{n-1} \end{pmatrix} = \begin{pmatrix} 1 & 1 & 1 & \cdots & 1 \\ 1 & \omega_n & \omega_n^2 & \cdots & \omega_n^{n-1} \\ 1 & \omega_n^2 & \omega_n^4 & \cdots & \omega_n^{2n-2} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & \omega_n^{n-1} & \omega_n^{2n-2} & \cdots & \omega_n^{(n-1)(n-1)} \end{pmatrix} \begin{pmatrix} a_0 \\ a_1 \\ a_2 \\ \vdots \\ a_{n-1} \end{pmatrix}$$

(k, j)-ty element V_n to ω_n^{kj} dla $j, k = 0, 1, \ldots, n-1$.

Z tej postaci widzimy, że aby wykonać operację odwrotną do $\mathrm{DFT}(\cdot)$, czyli $a = \mathrm{DFT}_n^{-1}(y)$ musimy pomnożyć y przez macierz V_n^{-1} odwrotną do V_n. Jej (k, j)-ty element to ω_n^{-kj}/n dla $j, k = 0, 1, \ldots, n-1$. Dowód:

$$\begin{aligned} [V_n^{-1} V_n]_{jj'} &= \sum_{k=0}^{n-1} (\omega_n^{-kj}/n)(\omega_n^{kj'}) \\ &= \sum_{k=0}^{n-1} (\omega_n^{k(j'-j)}/n) \end{aligned}$$

Gdy $j = j'$ wyrażenie jest równe 1, a dla pozostałych j, j' mamy:

$$\begin{aligned} \sum_{k=0}^{n-1} (\omega_n^k)^j &= \frac{(\omega_n^k)^n - 1}{\omega_n^k - 1} \\ &= \frac{\omega_n^{nk} - 1}{\omega_n^k - 1} \\ &= \frac{(1)^k - 1}{\omega_n^k - 1} = 0 \end{aligned}$$

... czyli otrzymaliśmy $[V_n^{-1} V_n] = Id_n$. Wobec tego $\mathrm{DFT}_n^{-1}(y)$ wynosi:

$$a_j = \frac{1}{n} \sum_{k=0}^{n-1} y_k \omega_n^{-kj}$$

wystarczy zatem zmodyfikować FFT tak, aby zamienić a oraz y miejscami, zamienić ω_n na ω_n^{-1} i podzielić wynik przez n.

Powyższe kroki pokazują nam, że mnożenie wektorów współczynników a, b długości n takiego, że n jest potęgą liczby 2 można przedstawić: $a(x) \otimes b(x) = \mathrm{DFT}_{2n}^{-1}(\mathrm{DFT}_{2n}(a) \cdot \mathrm{DFT}_{2n}(b))$ gdzie wektory wejściowe są (być może) uzupełnione zerami z przodu do długości $2n$ a $\cdot$ opisuje iloczyn "po współrzędnych".

Haszowanie

Niech $S \subseteq U$ będzie ustalonym zbiorem oraz niech $h : U \to 0, \ldots, m-1$ będzie rodziną wybraną losowo z pewnej rodziny uniwersalnej. Kolizją nazywamy parę $x, y \in S$ taką, że $h(x) = h(y)$. Im większe wybierzemy m, tym, w przeciętnym przypadku kolizji będzie mniej. Zależność wygląda następująco:

$$\begin{aligned} E[\text{całkowita liczba kolizji}] &= E[\sum_{x,y \in S} 1_{h(x)=h(y)}] \\ &= \sum_{x,y \in S} E[1_{h(x)=h(y)}] \\ &= \sum_{x,y \in S} P[h(x) = h(y)] \\ &\leqslant \binom{n}{2} \frac{1}{m} < \frac{n^2}{2m} \end{aligned}$$

Nierówność Markowa: Dla każdej zmiennej losowej X o wartości oczekiwanej $E(X)$ i dla każdego $\epsilon > 0$ oraz $p > 0$:

$$P\left(|X| \geqslant \epsilon\right) \leqslant \frac{E(|X|^p)}{\varepsilon^p}$$

Korzystając z nierówności Markowa:

$$P[\text{całkowita liczba kolizji} \geqslant \frac{n^2}{m}] < \frac{1}{2}$$

Wnioski:

- Dla $m = n^2$ z prawdopodobieństwem co najmniej $\frac{1}{2}$ w ogóle nie będzie kolizji.
- Dla $m = n$ z prawdopodobieństwem co najmniej $\frac{1}{2}$ całkowita liczba kolizji będzie mniejsza niż n.

Słownik statyczny Chcemy zbudować słownik statyczny - strukturę danych, której nie planujemy modyfikować już po stworzeniu. Chcielibyśmy by umożliwiał on wykonywanie operacji:

- Build(S) - budowania struktury dla zbioru S.
- Lookup(x) - odpowiadającą na pytanie czy $x \in S$.

Wystarczy zatem zadeklarować i zainicjalizować tablicę T rozmiaru n^2 elementów zerami. Dla kolejno wylosowanych funkcji $h(x)$ z rodziny H haszujemy nimi kolejno wszystkie elementy S. W przypadku wykrycia kolizji, porzucamy h i wybieramy nową funkcję z rodziny, dopóki nie znajdziemy funkcji bezkolizyjnej.
Niech $|S| = n$. Z uwagi na to, że dla rozmiaru tablicy haszującej $m = n^2$ z prawdopodobieństwem co najmniej $\frac{1}{2}$ w ogóle nie będzie kolizji, potrzebujemy średnio 2 prób by znaleźć funkcję haszującą taką, która ich nie posiada. Czas funkcji Build(S) jest zatem rzędu $O(n)$.

Haszowanie doskonałe Niech H będzie uniwersalną rodziną funkcji haszujących postaci $h : U \to \{0, \dots, n-1\}$. Operacja Build($S$) wygląda wtedy tak: Losujemy h z rodziny H dopóki całkowita liczba kolizji nie będzie mniejsza niż n.
Dla każdego $i = 0, 1, \dots, n-1$ niech $S_i = \{x \in S | h(x) = i\}$. Zbudujmy statyczny słownik T_i dla każdego zbioru S_i algorytmem budowania słownika statycznego. Adres słownika T_i przechowujemy w komórce $T[i]$ tablicy $T[0..n]$.
Operacja Lookup(x) sprowadza się do wykonania Lookup(x) na $T_{h(x)}$, co działa w czasie stałym.
Rozmiar struktury jest podzielony między tablicę T rozmiaru $O(n)$, stałą pamięciowo reprezentację funkcji h oraz słowniki T_i. Ich łączny rozmiar to $O(\sum_{i=0}^{n-1} |S_i|^2)$, co możemy oszacować przez:

$$\sum_{i=0}^{n-1} |S_i|^2 = n + 2\sum_{i=0}^{n-1} \frac{|S_i|^2 - |S_i|}{2} = 2\sum_{i=0}^{n-1} \binom{|S_i|}{2} + n$$

Ponieważ $\sum_{i=0}^{n-1} \binom{|S_i|}{2}$ jest całkowitą liczbą kolizji elementów z S przy użyciu funkcji h, a o h zakładamy z treści naszego algorytmu, że liczba jej kolizji nie przekracza n, możemy powiedzieć, że $\sum_{i=0}^{n-1} \binom{|S_i|}{2} \leqslant n$, a zatem:

$$\sum_{i=0}^{n-1} |S_i|^2 \leqslant 3n$$

Tak jak w przypadku słownika statycznego, liczba prób w średnim wypadku wynosi 2. Ponieważ sprawdzić liczbę kolizji możemy w czasie $O(n)$, pasującą do naszych potrzeb funkcję h znajdziemy w czasie $O(n)$.

Ponieważ dla każdego i, słownik T_i budowany jest w oczekiwanym czasie $O(|S_i|^2)$, z liniowości wartości oczekiwanej, słowniki wybudujemy w czasie oczekiwanym $O(\sum_{i=0}^{n-1} |S_i|^2)$ który już oszacowaliśmy przez $O(n)$.