Uniwersytet Warszawski
Wydział Matematyki, Informatyki i Mechaniki

Marysia Nazarczuk
Nr albumu: 417755

KONSTRUKCJE PALEYA DLA MACIERZY HADAMARDA

Streszczenie

Macierze Hadamarda są kwadratowymi macierzami o elementach równych ± 1, których wiersze są ortogonalne. Konstrukcja takich macierzy stanowi jedno z kluczowych zagadnień w kombinatoryce i teorii macierzy, ze względu na ich liczne zastosowania. Jedną z klasycznych metod budowy macierzy Hadamarda jest konstrukcja Paleya, która opiera się na szczególnych własnościach ciał skończonych i kwadratów resztowych. W tej pracy analizujemy szczegóły konstrukcji Paleya oraz prezentujemy jej warianty. Omówimy również przykłady oraz dowody istnienia macierzy Hadamarda o rozmiarach opartych na ciałach skończonych, ilustrując tym samym, jak algebraiczne struktury wspomagają efektywne tworzenie tych macierzy. Praca zawiera przegląd teoretyczny, jak i praktyczne aspekty implementacji konstrukcji Paleya.

Praca wykonana pod kierunkiem
dr. Adam Malinowski
Wydział MIM

Warszawa, Styczeń 2025

Spis treści

1. Wprowadzenie

Macierze Hadamarda [1] to kwadratowe macierze o wymiarach $m \times m$, których elementy wynoszą ± 1, a wiersze (lub kolumny) są wzajemnie ortogonalne, co oznacza, że ich iloczyny skalarne wynoszą zero. Formalnie, macierz Hadamarda spełnia zależność $H^T H = mI$, gdzie H^T oznacza macierz transponowaną, a I jest macierzą jednostkową.

Jedną z najprostszych metod konstrukcji macierzy Hadamarda jest rekurencyjna konstrukcja dla wymiarów będących potęgami liczby 2, na przykład:

$$H_2 = \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}, \quad H_{2m} = \begin{bmatrix} H_m & H_m \\ H_m & -H_m \end{bmatrix}.$$

W ogólności, jeśli macierz H jest macierzą Hadamarda, to macierz transponowana H^T również nią jest, co oznacza, że ortogonalność zachodzi zarówno dla wierszy, jak i kolumn.

Jednym z fundamentalnych problemów związanych z macierzami Hadamarda jest określenie, dla jakich rzędów m istnieją takie macierze. Hipoteza Hadamarda sugeruje, że macierz Hadamarda istnieje dla każdego m podzielnego przez 4, choć dla niektórych wartości m, takich jak 668, istnienie macierzy Hadamarda pozostaje niepotwierdzone.

W niniejszej pracy skoncentrujemy się na generowaniu macierzy Hadamarda różnymi sposobami, by móc uzyskiwać coraz to różne rzędy.

2. Notacja i definicje

W celu zrozumienia konstrukcji macierzy Hadamarda zaproponowanej przez Paleya [2], konieczne jest wprowadzenie kluczowych pojęć związanych z ciałami skończonymi. Konstrukcja ta bazuje na algebraicznych własnościach ciał skończonych, w szczególności na charakterze kwadratowym, który odgrywa fundamentalną rolę w budowie macierzy spełniających warunki Hadamarda. W tym rozdziale przedstawiamy podstawowe definicje i notacje, które będą wykorzystywane w dalszej części artykułu.

2.1. Charakter kwadratowy i macierz Jacobsthala

Charakter kwadratowy jest kluczowym narzędziem w teorii ciał skończonych i znajduje zastosowanie w wielu dziedzinach matematyki i informatyki, takich jak teoria kodowania czy kryptografia. W przypadku konstrukcji Paleya służy on do określania relacji pomiędzy elementami ciała skończonego.

Definicja 2.1.1. Charakter kwadratowy [3] $\chi(a)$ w ciele skończonym $\mathrm{GF}(q)$ określa, czy element a jest zerem, kwadratem innego elementu ciała (resztą kwadratową), czy też resztą niekwadratową. Formalnie definiujemy go jako:

$$\chi(a) = \begin{cases} 0 & \text{jeśli } a = 0, \\ 1 & \text{jeśli } a = b^2 \text{ dla pewnego niezerowego } b \in \mathrm{GF}(q), \\ -1 & \text{jeśli } a \text{ nie jest kwadratem żadnego elementu w } \mathrm{GF}(q). \end{cases}$$

Powyższa definicja pozwala na systematyczne klasyfikowanie elementów ciała skończonego na trzy kategorie, co ma kluczowe znaczenie przy budowie macierzy Jacobsthala.

Definicja 2.1.2. Macierz Jacobsthala [3], oznaczana przez Q, jest macierzą kwadratową wymiaru $q \times q$, której wiersze i kolumny indeksowane są elementami ciała skończonego $\mathrm{GF}(q)$. Element na przecięciu wiersza a i kolumny b definiuje się jako:

$$Q[a, b] = \chi(a - b),$$

gdzie χ jest charakterem kwadratowym określonym w poprzedniej definicji.

Macierz Jacobsthala jest narzędziem pozwalającym w elegancki sposób odwzorować charakter kwadratowy do struktury macierzowej. Stanowi ona podstawę dla konstrukcji bardziej złożonych macierzy, takich jak macierze Hadamarda.

2.2. Konstrukcje Paleya

Konstrukcje Paleya stanowią klasyczny przykład zastosowania teorii ciał skończonych w teorii macierzy Hadamarda. Poniżej opisujemy dwie główne konstrukcje, zależnie od wartości $q \pmod 4$.

Definicja 2.2.1. Jeżeli $q \equiv 3 \pmod 4$, można skonstruować macierz Hadamarda H rzędu $q + 1$ za pomocą wzoru [3]:

$$H = I + \begin{bmatrix} 0 & j^T \\ -j & Q \end{bmatrix},$$

gdzie:

- j jest wektorem kolumnowym wypełnionym jedynkami o długości q,

- I to macierz jednostkowa wymiaru $(q+1) \times (q+1)$,
- Q to macierz Jacobsthala.

Macierz ta spełnia własność skośnosymetryczną, co oznacza, że:

$$H + H^T = 2I.$$

Definicja 2.2.2. Jeżeli $q \equiv 1 \pmod 4$, macierz Hadamarda H rzędu $2(q+1)$ można skonstruować jako:

$$H = \begin{bmatrix} C + I_{q+1} & C - I_{q+1} \\ C - I_{q+1} & -(C + I_{q+1}) \end{bmatrix},$$

gdzie:

- C to macierz zdefiniowana jako:

$$C = \begin{bmatrix} 0 & j^T \\ j & Q \end{bmatrix},$$

- j jest wektorem kolumnowym wypełnionym jedynkami o długości q,
- I_{q+1} to macierz jednostkowa wymiaru $(q+1) \times (q+1)$,
- Q to macierz Jacobsthala.

Tak skonstruowana macierz H jest symetryczną macierzą Hadamarda, co oznacza, że $H^T = H$.

3. Przygotowania

3.1. Iloczyn Kroneckera

Iloczyn Kroneckera jest narzędziem, które pozwala na tworzenie większych macierzy Hadamarda z mniejszych. Twierdzenie poniżej pokazuje, że taka operacja zachowuje własności macierzy Hadamarda.

Twierdzenie 3.1.1. Iloczyn Kroneckera macierzy Hadamarda jest macierzą Hadamarda.

Dowód: Niech H_a i H_b będą macierzami Hadamarda rzędu a oraz b. Zauważmy, że macierz $H_a \otimes H_b$ (gdzie $\otimes$ oznacza iloczyn Kroneckera) spełnia:

$$(H_a \otimes H_b)^T \cdot (H_a \otimes H_b) = (H_a^T \otimes H_b^T) \cdot (H_a \otimes H_b).$$

Korzystając z własności iloczynu Kroneckera, mamy:

$$(H_a^T \cdot H_a) \otimes (H_b^T \cdot H_b) = aI \otimes bI = ab \cdot I.$$

Stąd wynika, że $H_a \otimes H_b$ również jest macierzą Hadamarda. □

Operacja ta umożliwia efektywną budowę dużych macierzy Hadamarda.

3.2. Pomocnicze lematy

Zanim przejdziemy do dowodu głównych konstrukcji, wprowadzimy kilka lematów, które okażą się przydatne w analizie własności macierzy Jacobsthala oraz macierzy Hadamarda. Każdy z lematów dotyczy specyficznych właściwości charakteru kwadratowego oraz wynikających z niego struktur algebraicznych macierzy.

Lemat 3.2.1. Jeśli $q \equiv 3 \pmod 4$, to -1 nie jest kwadratem oraz macierz Q jest skośnosymetryczna.

Dowód: Załóżmy przeciwnie, że istnieje $b \in \mathrm{GF}(q)$, takie że $-1 = b^2$. Niech $q = 4k + 3$. Wówczas mamy:

$$-1 \equiv (-1)^{2k+1} \equiv (-1)^{\frac{q-1}{2}} \equiv (b^2)^{\frac{q-1}{2}} = b^{q-1} = 1.$$

Otrzymujemy sprzeczność, więc -1 nie może być kwadratem. Ponadto, dla dowolnych $a, b \in \mathrm{GF}(q)$ zachodzi:

$$\chi(a - b) = -\chi(b - a),$$

co oznacza, że $Q^T = -Q$. □

Wnioski z powyższego lematu mają kluczowe znaczenie dla konstrukcji macierzy Hadamarda w przypadku, gdy $q \equiv 3 \pmod 4$. Właściwość skośnosymetryczności macierzy Q odgrywa istotną rolę w późniejszych dowodach.

Lemat 3.2.2. Jeśli $q \equiv 1 \pmod 4$, to -1 jest kwadratem oraz macierz Q jest symetryczna.

Dowód: Ponieważ $q \equiv 1 \pmod 4$, zachodzi $q - 1 = 4k$ dla pewnego $k \in \mathbb{Z}$. Grupa multiplikatywna $\mathrm{GF}(q)^\times$ ma rząd $q - 1$. Niech g będzie generatorem w tej grupie. W zapisie cyklicznym element -1 możemy wyrazić jako $= 1 = g^m$. Skoro $(-1)^2 = 1$, to $2m \equiv 0(\pmod q) - 1)$, zatem $-1 = g^{\frac{q-1}{2}}$. Ponieważ $\frac{q-1}{2} = 2k$ jest liczbą parzystą, wynika, że -1 jest kwadratem w $\mathrm{GF}(q)$. Dalej dla dowolnych $a, b \in \mathrm{GF}(q)$ zachodzi:

$$\chi(a - b) = \chi(b - a),$$

co oznacza, że $Q^T = Q$. □

□

W przypadku $q \equiv 1 \pmod 4$, symetria macierzy Q pozwala na wprowadzenie konstrukcji macierzy Hadamarda o różnych wymiarach i szczególnych własnościach algebraicznych.

Lemat 3.2.3. Dla $b \neq 0$ zachodzi $\sum_a \chi(a) \cdot \chi(a+b) = -1$.

Dowód: Rozważmy sumę:

$$\sum_a \chi(a) \cdot \chi(a+b).$$

Jeżeli $a = 0$, to $\chi(0) \cdot \chi(b) = 0$. Dla $a \neq 0$ istnieje $z \neq 1$, takie że $a + b = az$, ponieważ $z = 1 + b \cdot a^{-1}$. Wtedy:

$$\sum_a \chi(a) \cdot \chi(a+b) = \sum_a \chi(a)^2 \cdot \chi(z) = \sum_z \chi(z) - \chi(1) = 0 - 1 = -1.$$

□

Lemat ten wprowadza istotne informacje o wzajemnych relacjach między elementami w macierzy Jacobsthala. W szczególności wynik ten jest kluczowy dla uzasadnienia właściwości iloczynów elementów macierzy Q.

Lemat 3.2.4. $Q^T Q = qI - J$, gdzie I jest macierzą jednostkową, a J jest macierzą wypełnioną jedynkami.

Dowód: Niech $B = Q^T Q$. Wtedy dla $i \neq j$:

$$b_{ij} = \sum_k \chi(a_i - a_k) \cdot \chi(a_j - a_k) = -1,$$

co wynika z lematu 3.2.3. Dla $i = j$, elementy w kolumnie odpowiadają 0 oraz $q - 1$ wartościom ± 1, więc:

$$b_{ii} = q - 1.$$

Stąd $B = qI - J$. □

Ostateczny wynik tego lematu pokazuje, że macierz $Q^T Q$ przyjmuje prostą postać macierzy diagonalnej z odpowiednimi korektami w przypadku macierzy wypełnionej jedynkami. Ta właściwość stanowi podstawę dla konstrukcji i analizy macierzy Hadamarda.

4. Dowody konstrukcji Paleya

4.1. Dowód konstrukcji I

Przejdźmy do pierwszej konstrukcji macierzy Hadamarda. W tym przypadku założenie $q \equiv 3 \pmod 4$ pozwala na wykorzystanie własności skośnosymetrycznej macierzy Q, jak zostało wykazane w poprzednich lematów. Konstrukcja oparta jest na macierzy blokowej, która łączy macierz jednostkową, wektor jedynek oraz macierz Q.

Twierdzenie 4.1.1. Niech $q \equiv 3 \pmod 4$ oraz niech

$$H = I + \begin{bmatrix} 0 & j^T \\ -j & Q \end{bmatrix},$$

gdzie j to wektor kolumnowy wypełniony jedynkami o długości q, a I to macierz jednostkowa wymiaru $(q+1) \times (q+1)$. Wówczas macierz H jest macierzą Hadamarda rzędu $q+1$.

Dowód: Musimy udowodnić, że $H^T H = m \cdot I$ dla pewnego m. Mamy:

$$H^T H = \left(I + \begin{bmatrix} 0 & -j^T \\ j & Q \end{bmatrix} \right) \cdot \left(I + \begin{bmatrix} 0 & j^T \\ -j & Q \end{bmatrix} \right).$$

Rozpisując mnożenie blokowe, otrzymujemy:

$$H^T H = I + \begin{bmatrix} 0 & 0 \\ 0 & Q^T + Q \end{bmatrix} + \begin{bmatrix} j^T \cdot j & -j^T \cdot Q \\ Q^T \cdot j & Q^T Q + J \end{bmatrix}.$$

Teraz sprawdzimy właściwości poszczególnych składników:

- $Q^T = -Q$, więc druga macierz w powyższej sumie jest zerowa.
- $j^T \cdot j = q$, ponieważ j ma długość q.
- $-j^T \cdot Q = 0$ oraz $Q^T \cdot j = 0$, co wynika z równowagi między liczbą kwadratów i niekwadratów w $GF(q)$.
- $Q^T Q = qI - J$, zgodnie z lematem 3.2.4.

Ostatecznie:

$$H^T H = I + \begin{bmatrix} q & 0 \\ 0 & qI \end{bmatrix} = (q+1) \cdot I.$$

Zatem macierz H jest macierzą Hadamarda rzędu $q+1$. □

4.2. Dowód konstrukcji II

W drugiej konstrukcji wykorzystujemy założenie $q \equiv 1 \pmod 4$. Dzięki temu macierz Q jest symetryczna, co umożliwia budowę bardziej złożonej struktury blokowej. Konstrukcja ta pozwala uzyskać macierz Hadamarda o podwojonym wymiarze w porównaniu do pierwszej metody.

Twierdzenie 4.2.1. Niech $q \equiv 1 \pmod 4$. Wówczas macierz

$$H = \begin{bmatrix} C + I_{q+1} & C - I_{q+1} \\ C - I_{q+1} & -(C + I_{q+1}) \end{bmatrix}$$

jest macierzą Hadamarda rzędu $2(q+1)$.

Dowód: Musimy wykazać, że $H^T H = m \cdot I$, dla pewnego m. Skoro C jest symetryczna (lemat 3.2.2), a I_{q+1} jest macierzą jednostkową, to transpozycja macierzy H jest równa jej samej:

$$H^T = H.$$

Obliczamy teraz $H^T H$ jako:

$$H^T H = \begin{bmatrix} C + I_{q+1} & C - I_{q+1} \\ C - I_{q+1} & -(C + I_{q+1}) \end{bmatrix} \begin{bmatrix} C + I_{q+1} & C - I_{q+1} \\ C - I_{q+1} & -(C + I_{q+1}) \end{bmatrix}.$$

Po rozwinięciu otrzymujemy:

$$H^T H = \begin{bmatrix} K & L \\ M & N \end{bmatrix},$$

gdzie:

$$\begin{aligned} K &= (C + I_{q+1})(C + I_{q+1}) + (C - I_{q+1})(C - I_{q+1}), \\ L &= (C + I_{q+1})(C - I_{q+1}) - (C - I_{q+1})(C + I_{q+1}), \\ M &= (C - I_{q+1})(C + I_{q+1}) - (C + I_{q+1})(C - I_{q+1}), \\ N &= (C - I_{q+1})(C - I_{q+1}) + (C + I_{q+1})(C + I_{q+1}). \end{aligned}$$

Wyniki są następujące:

- $K = N = 2C^2 + 2I_{q+1}$,
- $L = M = 0$, ponieważ wszystkie odpowiednie składniki się znoszą.

Wiemy, że $C^2 = q \cdot I_{q+1}$, więc:

$$K = N = 2(q + 1) \cdot I_{q+1}.$$

Ostatecznie:

$$H^T H = 2(q + 1) \cdot I,$$

co dowodzi, że H jest macierzą Hadamarda rzędu $2(q + 1)$. □

5. Generowanie macierzy Jacobsthala

5.1. Konstrukcja ciała $\mathbb{F}_{p^m}$

Aby skonstruować ciało $\mathbb{F}_{p^m}$, gdzie $q = p^m$, p jest liczbą pierwszą, a m jest liczbą naturalną, postępujemy według poniższych kroków:

1. **Ciało podstawowe** $\mathbb{F}_p$: Ciało $\mathbb{F}_p$ to ciało skończone z p elementami, gdzie p jest liczbą pierwszą. Elementy tego ciała to $\{0, 1, 2, \ldots, p-1\}$, a działania w $\mathbb{F}_p$ są wykonywane modulo p.

2. **Wielomiany nad** $\mathbb{F}_p$: Rozważamy pierścień wielomianów $\mathbb{F}_p[x]$, czyli wielomiany dowolnego stopnia o współczynnikach w $\mathbb{F}_p$.

3. **Wielomian nierozkładalny**: Wybieramy wielomian nierozkładalny $f(x)$ stopnia m w $\mathbb{F}_p[x]$. Wielomian $f(x)$ jest nierozkładalny, jeśli nie da się go rozłożyć na iloczyn wielomianów o niższych stopniach w $\mathbb{F}_p[x]$.

4. **Rozszerzenie ciała**: Tworzymy rozszerzenie ciała $\mathbb{F}_{p^m}$ jako pierścień ilorazowy:

$$\mathbb{F}_{p^m} = \mathbb{F}_p[x]/(f(x)),$$

   gdzie $(f(x))$ oznacza ideał generowany przez $f(x)$. Elementy $\mathbb{F}_{p^m}$ można reprezentować jako klasy reszt wielomianów stopnia mniejszego niż m.

5. **Konstrukcja elementów**: Elementy $\mathbb{F}_{p^m}$ to klasy reszt z dzielenia wielomianów przez $f(x)$. Ciało $\mathbb{F}_{p^m}$ ma dokładnie p^m elementów.

W praktyce, ciałem będą wszystkie wielomiany stopnia mniejszego niż m o współczynnikach w $\mathbb{F}_p$ (jest ich p^m). Działaniem dodawania będzie zwykłe dodawanie wielomianów modulo p. Mnożenie będzie polegało na wymnożeniu wielomianów modulo p, a następnie wzięciu reszty przy dzieleniu przez nierozkładalny wielomian $f(x)$.

Poniżej znajduje się funkcja `generateFieldElements`, która generuje wszystkie elementy ciała GF(p^m).

```
vector<Polynomial> generateFieldElements(int p, int m) {
    vector<Polynomial> elements;
    int numElements = pow(p, m);
    for (int i = 0; i < numElements; ++i) {
        Polynomial poly(m, 0);
        int value = i;
        for (int j = 0; j < m; ++j) {
            poly[j] = value % p;
            value /= p;
        }
        elements.push_back(poly);
    }
    return elements;
}
```

Funkcja `generateFieldElements` tworzy wszystkie możliwe wielomiany reprezentujące elementy ciała skończonego GF(p^m).

5.2. Obliczanie reszt kwadratowych dla każdego elementu ciała $\mathbb{F}_{p^m}$

W celu stworzenia tablicy reszt kwadratowych `remainders`, dla każdego elementu ciała s, liczymy jej kwadrat za pomocą funkcji `squarePolynomial`, a następnie dzielimy wynik z resztą przez wielomian nierozkładalny $f(x)$ za pomocą funkcji `divideWithRemainder`.

```
Polynomial squarePolynomial(const Polynomial &s, int p) {
    int deg_s = s.size() - 1;
    Polynomial result(2 * deg_s + 1, 0);
    for (int i = 0; i <= deg_s; ++i) {
        for (int j = 0; j <= deg_s; ++j) {
            result[i + j] += s[i] * s[j];
            result[i + j] %= p;
        }
    }
    return result;
}
```

Funkcja `squarePolynomial` oblicza kwadrat wielomianu w ciele GF(p), zwracając wynikowy wielomian.

```
Polynomial divideWithRemainder(const Polynomial &w, const Polynomial &q, int p) {
    Polynomial remainder = w;
    for (int i = w.size() - 1; i >= int(q.size()) - 1; --i) {
        if (remainder[i] != 0) {
            int coef = remainder[i] / q.back();
            for (int j = 0; j < q.size(); ++j) {
                remainder[i - j] -= coef * q[q.size() - 1 - j];
                remainder[i - j] %= p;
                if (remainder[i - j] < 0) remainder[i - j] += p;
            }
        }
    }
    while (!remainder.empty() && remainder.back() == 0) {
        remainder.pop_back();
    }
    return remainder;
}
```

Funkcja `divideWithRemainder` wykonuje dzielenie wielomianów w ciele modularnym GF(p), zwracając resztę z dzielenia. Operacje są wykonywane z redukcją modulo p.

5.3. Generowanie macierzy

Konstrukcja macierzy Jacobsthala za pomocą `generateJacobianMatrix` polega na tym, że dla każdego elementu `matrix[i][j]` obliczamy różnicę wielomianów: i-ty element ciała $\mathbb{F}_q$ oraz j-ty element ciała $\mathbb{F}_q$. Następnia dla wyniku sprawdzane jest, za pomocą funkcji `isQuadraticResidue`, czy jest on kwadratem pewnego elementu $\mathbb{F}_q$. Jeśli tak to umieszczamy w komórce 1 w przeciwnym razie -1. Jeśli natomiast $i = j$, to w komócre umieszczamy element 0.

```
vector<vector<int>> generateJacobianMatrix(const vector<Polynomial> &fieldElements,
                                           const vector<Polynomial> &remainders,
                                           int p) {
    int n = fieldElements.size();
    vector<vector<int>> matrix(n, vector<int>(n, 0));

    for (int i = 0; i < n; ++i) {
        for (int j = 0; j < n; ++j) {
            if (i == j) {
                matrix[i][j] = 0;
            } else {
                Polynomial diff = fieldElements[i];

                for (int k = 0; k < diff.size(); ++k) {
                    if (k < fieldElements[j].size()) {
                        diff[k] -= fieldElements[j][k];
                    }
                    diff[k] %= p;
                    if (diff[k] < 0) diff[k] += p;
                }

                while (!diff.empty() && diff.back() == 0) {
```

```
                diff.pop_back();
            }

            matrix[i][j] = isQuadraticResidue(diff, remainders, n, p) ? 1 : -1;
        }
    }
}

    return matrix;
}
```

Funkcja `generateJacobianMatrix` generuje macierz Jacobsthala dla ciała `fieldElements` oraz wielomianu nierozkładalnego `remainers`.

```
bool isQuadraticResidue(const Polynomial &diff, const vector<Polynomial> &remainders,
                        int n, int p) {

    for (const Polynomial &r : remainders) {
        if (diff == r && n % 4 == 3) {
            return true;
        }

        if (n % 4 == 1) {
            Polynomial negated_r = negatePolynomial(r, p);
            if (diff == r || diff == negated_r) {
                return true;
            }
        }
    }
    return false;
}
```

Funkcja `isQuadraticResidue` sprawdza czy wielomian jest kwadratem jakiegoś innego wielomianu.

```
Polynomial negatePolynomial(const Polynomial &poly, int p) {
    Polynomial negated = poly;

    for (int &coeff : negated) {
        coeff = (p - coeff) % p;
        if (coeff < 0) coeff += p;
    }

    return negated;
}
```

Funkcja `negatePolynomial` wyznacza wielomian przeciwny do danego.

6. Generowanie macierzy Hadamarda

Przy pomocy funkcji `construct_hadamard` sprawdzamy, który przypadek mamy do wygenerowania: $q = 2^m$ lub konstrukcje Paleya. Jeśli $q \neq 2^m$, to rozkładamy q na iloczyn liczb 2^k i $4 \cdot p^m$ i dla każdej z nich generujemy macierz Hadamarda, by następnie korzystając z iloczynu Kroneckera, wygenerować macierz rzędu q.

6.1. Przypadek $q = 2^m$

W przypadku gdy q jest potęgą liczby 2, to macierz Hadamarda generujemy przy użyciu funkcji `construct_power_of_two_hadamard`.

```
vector<vector<int>> construct_power_of_two_hadamard(int n) {
    vector<vector<int>> H = {{1}};

    while (H.size() < n) {
        int m = H.size();
        vector<vector<int>> new_H(2 * m, vector<int>(2 * m));

        for (int i = 0; i < m; i++) {
            for (int j = 0; j < m; j++) {
                new_H[i][j] = H[i][j];
                new_H[i][j + m] = H[i][j];
                new_H[i + m][j] = H[i][j];
                new_H[i + m][j + m] = -H[i][j];
            }
        }

        H = move(new_H);
    }

    return H;
}
```

6.2. Iloczyn Kroneckera macierzy

Funkcja `kroneckerProduct` liczy iloczyn Kroneckera macierzy A i B.

```
vector<vector<int>> kroneckerProduct(const vector<vector<int>> &A,
                                     const vector<vector<int>> &B) {
    int rowsA = A.size();
    int colsA = A[0].size();
    int rowsB = B.size();
    int colsB = B[0].size();

    int rowsResult = rowsA * rowsB;
    int colsResult = colsA * colsB;

    vector<vector<int>> result(rowsResult, vector<int>(colsResult, 0));

    for (int i = 0; i < rowsA; ++i) {
        for (int j = 0; j < colsA; ++j) {
            for (int k = 0; k < rowsB; ++k) {
                for (int l = 0; l < colsB; ++l) {
                    result[i * rowsB + k][j * colsB + l] = A[i][j] * B[k][l];
                }
            }
        }
    }
```

```
    return result;
}
```

6.3. Konstrukcja Paleya typu I

Funkcja `construct_q_plus_one_hadamard` generuje macierz Hadamarda o rozmiarze $n \times n$, w poniższy sposób:

1. **Generowanie elementów ciała** $GF(p^m)$: Funkcja tworzy wszystkie elementy $GF(p^m)$, gdzie p jest liczbą pierwszą, a m wymiarem rozszerzenia.
2. **Wybór nieredukowalnego wielomianu**: Znajduje nieredukowalny wielomian q, który definiuje rozszerzenie $GF(p^m)$.
3. **Obliczanie reszt kwadratowych**: Dla każdego elementu $s \in GF(p^m)$, funkcja oblicza s^2 (kwadrat wielomianu) i wyznacza resztę z dzielenia przez q.
4. **Tworzenie macierzy Jacobsthala**: Na podstawie elementów ciała $GF(p^m)$ i reszt kwadratowych generowana jest macierz Q.
5. **Budowanie macierzy Hadamarda**: Funkcja inicjalizuje macierz Hadamarda H wartościami 1, a następnie wypełnia jej wiersze i kolumny zgodnie z macierzą Q.

```
vector<vector<int>> construct_q_plus_one_hadamard(int n, int p, int m) {
    vector<Polynomial> fieldElements = generateFieldElements(p, m);

    Polynomial q = find_irreducible_polynomial(p, m);

    vector<Polynomial> remainders;
    for (const Polynomial &s : fieldElements) {
        Polynomial w = squarePolynomial(s, p);
        Polynomial remainder = divideWithRemainder(w, q, p);
        remainders.push_back(remainder);
    }

    vector<vector<int>> Q = generateJacobianMatrix(fieldElements, remainders, p);

    vector<vector<int>> H(n, vector<int>(n, 1));
    for (int i = 1; i < n; i++) {
        H[0][i] = 1;
        H[i][0] = -1;
        for (int j = 1; j < n; j++) {
            H[i][j] = Q[i - 1][j - 1];
        }
        H[i][i] = 1;
    }

    return H;
}
```

6.4. Konstrukcja Paleya typu II

Funkcja `construct_two_q_plus_one_hadamard` generuje macierz Hadamarda o rozmiarze $2(q+1) \times 2(q+1)$, gdzie $q = p^m$, w poniższy sposób:

1. **Generowanie elementów ciała** $GF(p^m)$: Funkcja tworzy wszystkie elementy $GF(p^m)$, gdzie p jest liczbą pierwszą, a m wymiarem rozszerzenia.

2. **Wybór nieredukowalnego wielomianu**: Znajduje nieredukowalny wielomian q, który definiuje rozszerzenie $GF(p^m)$.

3. **Obliczanie reszt kwadratowych**: Dla każdego elementu $s \in GF(p^m)$, funkcja oblicza s^2 (kwadrat wielomianu) i wyznacza resztę z dzielenia przez q.

4. **Tworzenie macierzy Jacobsthala** Q: Na podstawie elementów ciała $GF(p^m)$ i reszt kwadratowych generowana jest macierz Q, która odzwierciedla strukturę algebraiczną $GF(p^m)$. Macierz Q ma wymiary $(q+1) \times (q+1)$.

5. **Budowanie macierzy** C: Macierz C jest rozszerzeniem macierzy Q, gdzie wartości w pierwszym wierszu i kolumnie są ustawiane na 1, z wyjątkiem elementu $C[0][0] = 0$.

6. **Tworzenie macierzy jednostkowej** I: Funkcja generuje macierz jednostkową I o wymiarach $(q+1) \times (q+1)$, z jedynkami na przekątnej.

7. **Budowa końcowej macierzy Hadamarda** H: Macierz Hadamarda H jest budowana na podstawie macierzy C i I w następujący sposób:

   - Lewy górny kwadrat: $C + I$,
   - Prawy górny kwadrat: $C - I$,
   - Lewy dolny kwadrat: $C - I$,
   - Prawy dolny kwadrat: $-(C + I)$.

```
vector<vector<int>> construct_two_q_plus_one_hadamard(int n, int p, int m) {
    vector<Polynomial> fieldElements = generateFieldElements(p, m);

    Polynomial q = find_irreducible_polynomial(p, m);

    vector<Polynomial> remainders;
    for (const Polynomial &s : fieldElements) {
        Polynomial w = squarePolynomial(s, p);
        Polynomial remainder = divideWithRemainder(w, q, p);
        remainders.push_back(remainder);
    }

    vector<vector<int>> Q = generateJacobianMatrix(fieldElements, remainders, p);
    int sizer = pow(p, m) + 1;

    vector<vector<int>> C(sizer, vector<int>(sizer));
    for (int i = 1; i <= sizer - 1; i++) {
        for (int j = 1; j <= sizer - 1; j++) {
            C[i][j] = Q[i - 1][j - 1];
        }
    }
    for (int i = 1; i <= sizer - 1; i++) {
        C[0][i] = C[i][0] = 1;
    }
    C[0][0] = 0;

    vector<vector<int>> I(sizer, vector<int>(sizer, 0));
    for (int i = 0; i < sizer; i++) {
        I[i][i] = 1;
    }

    vector<vector<int>> H(2 * sizer, vector<int>(2 * sizer));
    for (int i = 0; i < sizer; i++) {
        for (int j = 0; j < sizer; j++) {
            H[i][j] = C[i][j] + I[i][j];
            H[i][j + sizer] = C[i][j] - I[i][j];
            H[i + sizer][j] = C[i][j] - I[i][j];
            H[i + sizer][j + sizer] = -(C[i][j] + I[i][j]);
```

```
            }
        }

        return H;
    }
```

7. Podsumowanie

Macierze Hadamarda są kwadratowymi macierzami o wymiarach $m \times m$, których elementy wynoszą ± 1, a wiersze są wzajemnie ortogonalne. Formalnie spełniają zależność $H^T H = mI$, gdzie H^T to macierz transponowana, a I jest macierzą jednostkową. Ich konstrukcja jest kluczowym zagadnieniem w kombinatoryce i pozostaje nadal nierozstrzygnięte w ogólności.

Jednym z klasycznych sposobów budowy tych macierzy jest konstrukcja Paleya, która opiera się na właściwościach ciał skończonych i kwadratów resztowych. W artykule przedstawiono dwa warianty konstrukcji Paleya: dla przypadków $q \equiv 3 \pmod 4$ oraz $q \equiv 1 \pmod 4$. Dla obu przypadków zaprezentowano dowody istnienia i szczegółowe omówienie procedur generowania macierzy, w tym wykorzystanie macierzy Jacobsthala oraz iloczynu Kroneckera.

Artykuł zawiera także przegląd teoretyczny i praktyczne aspekty implementacji konstrukcji, w tym algorytmy generowania ciał skończonych $GF(p^m)$, obliczania reszt kwadratowych i budowy macierzy Jacobsthala. Przytoczone dowody matematyczne oraz lematy podkreślają, jak algebraiczne struktury wspomagają efektywną budowę macierzy Hadamarda różnych rzędów.

Pełna implementacja wraz z dokumentacją znajduje się na githubie [4].

Bibliografia

[1] Jacques Hadamard. Resolution d'une question relative aux determinants. *Bull. des sciences math.*, 2:240–246, 1893.

[2] R. E. A. C. Paley. On orthogonal matrices. *Journal of Mathematics and Physics*, 12, 1933.

[3] Wikipedia contributors. Paley construction — wikipedia, the free encyclopedia, 2024.

[4] Marysia Nazarczuk. Hadamard-matrix, 2024. `https://github.com/mar2000/Hadamard-Matrix`.