Wybrane Zagadnienia Algebry - Sprawozdanie 1

Rafał Włodarczyk 279762

Kwiecień 2026

1 Zadanie 1 - Pierścień Liczb Gaussa

Pierścień Liczb Gaussa ($\mathbb{Z}[i]$ - *Gaussian Integers*) to zbiór liczb:

$$\mathbb{Z}[i] = \{a + bi : a, b \in \mathbb{Z}\}, \quad i^2 = -1$$

Wyróżniamy dwie składowe $z \in \mathbb{Z}[i]$ - rzeczywista a i urojona b. Dla zadanych danych wystarczające jest zdefiniowanie struktury danych, w oparciu o typ całkowity `int64_t`:

```
class Gauss {
  int64_t real;
  int64_t imaginary;
}
```

Klasa `Gauss` zawiera następujące metody:

- Norma zgodna z definicją $N(a + bi) = a^2 + b^2$. Dla zadanych danych $a + bi = 2 + 7i$:

  ```
  Norm(2+7i)
  -> N = 53
  ```

- Dzielenie z resztą X/Y: $X = QY + R$, gdzie $N(R) < N(Y)$. Dzielenie odbywa się w $\mathbb{C}$, a następnie jest zaokrąglane do najbliższych liczb całkowitych.

  $$Q = \frac{a+bi}{c+di} \cdot \frac{c-di}{c-di} = \left\lfloor \frac{ac+bd}{N(y)} \right\rceil + i \left\lfloor \frac{bc-ad}{N(y)} \right\rceil$$
  $$R = X - QY$$

  Na podstawie funkcji $\lfloor \cdot \rceil$ (zaokrąglenie do najbliższej liczby całkowitej) można zdefiniować 4 przypadki dzielenia, poprawne jeśli $N(R) < N(Y)$. Dla zadanych danych $X = (c+a)+(d+b)i = 11 + 14i$ oraz $Y = 6 + 2i$:

  ```
  Div(11+14i,6+2i), N(Y)=40
  -> Q = 2+1i ; R = 1+4i, N(R)=17
  -> Q = 2+2i ; R = 3-2i, N(R)=13
  -> Q = 3+1i ; R = -5+2i, N(R)=29
  -> Q = 3+2i ; R = -3-4i, N(R)=25
  ```

  Co zgadza się z oczekiwaniami ponieważ $N(R) < N(Y)$ dla wszystkich 4 przypadków oraz w $\mathbb{C}$ wynik dzielenia jest równy $2.35 + 1.55i$, widzimy jednocześnie że naturalne zaokrąglenie (od 0.5 w górę) daje najmniejszą normę reszty - wynik najbliższy temu w liczbach zespolonych.

- Wyznaczenie NWD oraz NWW. NWD jest wyznaczany za pomocą standardowego Algorytmu Euklidesa, wykorzystującego wyżej zdefiniowane dzielenie z resztą. NWW jest wyznaczany na podstawie NWD, zgodnie z zależnością $LCM(X,Y) = \frac{XY}{GCD(X,Y)}$. Dla zadanej listy $(a+bi, c+di, e+di)$:

  ```
  Array=(2+7i,9+7i,6+7i)
  -> GCD = 1+0i, -1+0i, 0+1i, 0-1i
  -> LCM = -245-725i, 245+725i, -245+725i, 245-725i
  ```

  Jeśli $GCD(X,Y) = 1$, to każdy *unit* $\in \{1, -1, i, -i\}$ dzieli X i Y wobec tego znów mamy 4 przypadki NWD i NWW. Wywołania dla listy pustej oraz jednoelementowej dają odpowiednio:

  ```
  GCD () = 0+0i
  GCD (2+7i) = 2+7i
  ```

2 Zadanie 2

Pierścień wielomianów $\mathbb{R}[x]$ to zbiór wielomianów o współczynnikach rzeczywistych:

$$\mathbb{R}[x] = \{a_n x^n + a_{n-1}x^{n-1} + ... + a_1 x + a_0 : n \in \mathbb{N}, a_i \in \mathbb{R}\}$$

Reprezentacja wielomianów jest realizowana za pomocą wektora współczynników:

```
class Poly {
  std::vector<double> c;
}
```

Gdzie `c[i]` to współczynnik przy x^i. Najprostszą optymalizacją pamięciową byłoby wykorzystanie struktury sparse vector, przy czym nie jest ona potrzebna do wykonania obliczeń dla zadanych niewielkich danych. Klasa `Poly` zawiera następujące metody:

- Norma $N(w) = N(a_n x^n + \cdots + a_0) = \max_i i : a_i > 0 = \mathrm{st}(w)$ - stopień wielomianu. Dla zadanych danych $(cx^a + b)$:

  ```
  Norm(9x^2 + 7)
  2
  ```

- Dzielenie z resztą - algorytm dzielenia w słupku, gdzie dzielnik jest odejmowany od dzielnej dopóki norma reszty jest większa lub równa normie dzielnika. Dla zadanych danych $X = cx^a + b = 9x^2 + 7$ oraz $Y = x + 1$:

  ```
  Div(9x^2 + 7;x + 1)
  Q = 9x - 9
  R = 16
  ```

- NWD,NWW - analogicznie do zadania 1, z wykorzystaniem algorytmu Euklidesa oraz zależności $LCM(X,Y) = \frac{XY}{GCD(X,Y)}$.
- EE - rozszerzony algorytm Euklidesa - dla wielomianów V,W znajduje v,W takie że $vV + wW = GCD(v,W)$. Dla zadanych danych $v(x) = ax^3 + bx^2 + cx + d$ oraz $w(x) = dx^3 + ex^2 + fx$ zachodzi:

```
Extended GCD(2x^3 + 7x^2 + 9x + 7, 7x^3 + 6x^2 + 2x)
GCD = 1
v = 7.2819148936169995x^2 - 45.545168304985815x + 30.627202839034926
w = -2.0805471124619994x^2 + 7.5143164327749465x + 21.585676291793167
```

Na tej podstawie jesteśmy w stanie znaleźć $g : 1 \notin GCD(v(x), w(x)+g)$ - czyli $v(x), w(x)+g$ muszą współdzielić pierwiastek. Wyznaczmy numerycznie pierwiastek $v(\alpha) = 0$ (metodą bisekcji), następnie wyznaczmy g takie że $w(\alpha) + g = 0$:

$$g = -w(\alpha)$$

Dla zadanych danych:

```
Root of v: -2.1693702696445882
w at root of v: -47.56767332520533
GCD(2x^3 + 7x^2 + 9x + 7, 7x^3 + 6x^2 + 2x + 47.56767332520533)
GCD = x + 2.1693702700163935
LCM(2x^3 + 7x^2 + 9x + 7, 7x^3 + 6x^2 + 2x + 47.56767332520533)
LCM = 14x^5 + 30.628816219770492x^4 + 42.554756687034x^3 +
      + 119.8183226457778x^2 + 133.04340632545495x + 153.48864962269494
```

3 Zadanie 3

Porządek produktowy $\leqslant$ na $\mathbb{N}^n$ jest definiowany jako:

$$(a_1, a_2, \ldots, a_n) \leqslant (b_1, b_2, \ldots, b_n) \iff (\forall i = 1, 2, \ldots, n)\, a_i \leqslant b_i$$

Wykonajmy sprawdzenie dla zadanych par $(a, b), (c, d), (e, f) = (2, 7), (9, 7), (6, 2)$ (każdy z każdym):

```
compare([2, 7], [2, 7]) is ==
compare([2, 7], [9, 7]) is <=
compare([2, 7], [6, 2]) is no comparison possible
compare([9, 7], [2, 7]) is >=
compare([9, 7], [9, 7]) is ==
compare([9, 7], [6, 2]) is >=
compare([6, 2], [2, 7]) is no comparison possible
compare([6, 2], [9, 7]) is <=
compare([6, 2], [6, 2]) is ==
```

Wykonajmy sprawdzenie dla trójek $(a, c, e), (b, d, f) = (2, 9, 6), (7, 7, 2)$:

```
compare([2, 9, 6], [7, 7, 2]) is no comparison possible
```

Następnie tworzymy algorytm znajdujący dla danego zbioru punktów $A \subset \mathbb{N}^n$ zbiór elementów $\leqslant$-minimalnych:

$$\mathrm{Min}(A) = \{a \in A : \nexists b \in A : b < a\}$$

Algorytm naiwny sprawdza dla każdego punktu $a \in A$ czy istnieje punkt $b \in A$ taki że $b < a$. Jeśli tak, to a nie jest $\leqslant$-minimalny. Jego złożoność można ocenić na $\mathcal{O}(n^2)$, gdyż dla każdego z n punktów sprawdzamy pozostałe $n - 1$ punkty. Algorytm ten jest jednak zupełnie nieefektywny dla zadanego zestawu danych, dlatego proponowany algorytm wykorzystuje sprytniejsze sprawdzanie.

```
for (p in A)
    is_min = true
    for (q in MIN)
        if (p < q)
            MIN.remove(q)
            is_min = false
            break
        else if (q < p)
            is_min = false
            break
    if (is_min)
        MIN.add(p)
```

Musimy sprawdzić każdy element p z A. Następnie budujemy zbiór MIN, który będzie zawierał $\leqslant$-minimalne elementy. Jeśli znajdziemy taki element p, że istnieje $q \in \text{MIN}$ taki że $p < q$, to usuwamy q ze zbioru MIN, ponieważ q nie jest już $\leqslant$-minimalny. Jeśli zachodzi $q < p$ to na pewno p nie jest $\leqslant$-minimalny, więc możemy przejeść dalej. Złożoność tego algorytmu zależy od rozmiaru zbioru $|\text{MIN}| = m$ i wynosi $\mathcal{O}(nm)$. W najgorszym przypadku, gdy wszystkie elementy są $\leqslant$-minimalne, złożoność będzie $\mathcal{O}(n^2)$, ale w praktyce jest znacznie lepsza, ponieważ $|\text{MIN}| \ll |A|$ dla zadanych danych (szacujemy takim bounding box-em, w granicach rozsądku - kwadratu wokoł okręgu i jego dolnego zarysu).

- Szukamy elementów $\leqslant$-minimalnych w zbiorze $A = \{(x, y) \in \mathbb{N}^2 : (x-a)^2 + (y-b)^2 < 5\}$, gdzie $a = 2$ i $b = 7$.

  ```
  Count Min elements of A: 3
  Min elements of A:
  [0, 7],[1, 6],[2, 5]
  ```

- Następnie szukamy elementów $\leqslant$-minimalnych w zbiorze $B = \{(x_1, x_2, x_3, x_4) \in \mathbb{N}^4 : (x_1 - c)^2 + (x_2 - d)^2 + (x_3 - e)^2 + (x_4 - f)^2 > 224\}$, gdzie $c = 9$, $d = 7$, $e = 6$ i $f = 2$.

  ```
  Count Min elements of B: 30
  Min elements of B:
  [0, 0, 0, 10], [0, 0, 13, 9], [0, 0, 14, 8], [0, 0, 15, 6], [0, 0, 16, 0]
  [0, 15, 0, 9], [0, 15, 13, 8], [0, 15, 14, 6], [0, 15, 15, 0], [0, 16, 0, 8]
  [0, 16, 13, 6], [0, 16, 14, 0], [0, 17, 0, 5], [0, 17, 13, 0], [0, 18, 0, 0]
  [19, 0, 0, 9], [19, 0, 13, 8], [19, 0, 14, 6], [19, 0, 15, 0], [19, 15, 0, 7]
  [19, 15, 13, 6], [19, 15, 14, 0], [19, 16, 0, 5], [19, 16, 13, 0], [19, 17, 0, 0]
  [20, 0, 0, 7], [20, 0, 13, 5], [20, 0, 14, 0], [20, 15, 0, 0], [21, 0, 0, 0]
  ```