



# UMCS

UNIwersytet Marii Curie-Skłodowskiej  
w Lublinie

Wydział Matematyki, Fizyki i Informatyki

Kierunek: **Informatyka**

**Patryk Popek**

nr albumu: 303843

## **Analiza danych zakupowych i budowa systemu rekomendacji ułożenia produktów sklepowych z wykorzystaniem bazy grafowej**

Purchase data analysis and development of a product placement  
recommendation system using a graph database.

Praca magisterska

napisana w Katedrze Oprogramowania Systemów Informatycznych  
pod kierunkiem doktor Anny Sasak-Okoń

**Lublin rok 2025**



# Spis treści

<b>Wstęp</b>	<b>5</b>
<b>1 Perspektywa klienta</b>	<b>7</b>
1.1 Manipulacja klientem . . . . .	7
1.2 Istniejące aplikacje . . . . .	8
1.2.1 LEAFIO AL Retail . . . . .	8
1.2.2 Mobile Touch . . . . .	9
1.2.3 Rekomendacja . . . . .	9
1.3 Użyte technologie . . . . .	9
1.3.1 Java . . . . .	9
1.3.2 Neo4j . . . . .	11
1.3.3 Python . . . . .	12
<b>2 Algorytm rekomendacji</b>	<b>13</b>
2.1 Zarys algorytmu . . . . .	13
2.2 Wczytanie danych . . . . .	13
2.3 Algorytm Apriori . . . . .	15
2.3.1 Wprowadzenie . . . . .	15
2.3.2 Implementacja . . . . .	16
2.4 Generowanie reguł . . . . .	18
2.4.1 Wprowadzenie . . . . .	18
2.4.2 Implementacja . . . . .	18
2.5 Przetwarzanie za pomocą Neo4j . . . . .	19
2.6 Wyznaczenie Siły produktów . . . . .	23
2.6.1 Zachowanie ogólne . . . . .	23
2.6.2 Przypadki szczególne . . . . .	24
2.7 Algorytm Dijkstry . . . . .	25
2.7.1 Wprowadzenie . . . . .	25
2.7.2 Implementacja . . . . .	26
2.8 Rekomendacja . . . . .	27

<b>3</b>	<b>Aplikacja</b>	<b>31</b>
3.1	Okno główne . . . . .	31
3.2	Interfejs grafowy . . . . .	33
3.2.1	Funkcjonalności . . . . .	33
3.2.2	Szablony grafów . . . . .	35
3.3	Interfejs koszykowy . . . . .	38
3.3.1	Kategorie Produktów . . . . .	38
3.3.2	Koszyki zakupowe . . . . .	39
3.3.3	Interfejs reguł i Apriori . . . . .	42
<b>4</b>	<b>Analiza</b>	<b>45</b>
4.1	Przedstawienie zbiorów . . . . .	45
4.1.1	Zbiór pierwszy . . . . .	45
4.1.2	Zbiór drugi . . . . .	46
4.2	Zachowania Wsparcia . . . . .	48
4.2.1	Koszyki odstające . . . . .	48
4.2.2	Poziom minimalnego Wsparcia . . . . .	49
4.2.3	Średnia liczba produktów w koszyku . . . . .	51
4.2.4	Wydajność algorytmu . . . . .	52
4.3	Zachowania reguł . . . . .	54
4.3.1	Ufność . . . . .	54
4.3.2	Dźwignia . . . . .	56
4.3.3	Siła . . . . .	57
4.4	Układ sklepu . . . . .	59
	<b>Podsumowanie</b>	<b>61</b>
	<b>Literatura</b>	<b>64</b>

# Wstęp

Miliony osób codziennie wykonuje zakupy w hipermarketach. Kupują oni chociażby produkty pierwszej potrzeby, takie jak jedzenie czy ubrania. Jest to idealna sytuacja dla właścicieli sklepów. Znajdują oni coraz to więcej sposobów aby nakłonić klientów do kupna większej ilości produktów, co za tym idzie, do zostawienia większej ilości pieniędzy w budynku sklepu.

Celem niniejszej pracy jest stworzenie algorytmu rekomendacji ułożenia produktów wewnątrz sklepu. Rekomendacja ta tworzona jest z użyciem bazy grafowej za pomocą analizy wzorców zakupowych odczytanych na podstawie analizy danych o zawartości koszyków zakupowych klientów.

Pierwszy rozdział poświęcony jest wstępowi teoretycznemu. Przytoczone zostały techniki manipulacyjne stosowane przez właścicieli sklepów na konsumentach. Wspomniane zostały również gotowe rozwiązania dostępne na rynku oraz technologie, za pomocą których został zaprojektowany poniższy algorytm rekomendacyjny.

Drugi rozdział skupia się na budowie algorytmu. Opisane zostały poszczególne kroki składające się na system rekomendacji. Przybliżono budowę algorytmu Apriori oraz Dijkstry. Wprowadzono zestaw metryk opisujących reguły asocjacyjne oraz opisano ich znaczenie. Przybliżono zasady działania bazy grafowej Neo4j.

Trzeci rozdział dotyczy opisu powstałej aplikacji. Wyszczególnione zostały najważniejsze komponenty wraz z ich kluczowymi funkcjonalnościami i sposobem ich działania. Pochylnono się nad budową interfejsu grafowego, dzięki któremu modelowany jest układ sklepu.

Ostatni, czwarty rozdział, skupia się na analizie zachowań stworzonego algorytmu. Wykonana on została na bazie wybranych zbiorów danych dostępnych na platformie Kaggle oraz na danych losowo wygenerowanych za pomocą języka Python. Określono optymalne poziomy metryk dla badanych zbiorów. Zbadano między innymi: wpływ rozkładu danych koszykowych oraz średniej liczby produktów w koszyku na liczbę wyznaczanych wzorców.



# 1 Perspektywa klienta

## 1.1 Manipulacja klientem

Handel jest podstawą sprawnie działającego społeczeństwa. Może on polegać zarówno na wymianie konkretnych towarów z rąk do rąk[1], jak i na płatności gotówką za konkretny przedmiot/usługę, gdzie jeden sposób nie wyklucza drugiego.

W obecnych czasach każdy człowiek jest mimowolnie klientem. Musi on pracować, aby otrzymać pieniądze, za które będzie mógł kupić podstawowe produkty potrzebne do przeżycia takie jak chociażby jedzenie. Sytuacje tą chętnie wykorzystują różne hipermarkety, chcące zmaksymalizować zysk generowany na klientach. Używają one różnych sztuczek, celem zwiększenia lojalności oraz zaufania klienta[2] wobec danej marki.

Shmuel I. Becher oraz Yuval Feldman wskazują na 5 technik manipulacji klientami stosowanych przez sklepy[3]:

1. Kolor: Autorzy artykułu wskazują na fakt, iż barwa widzianego koloru może mieć wpływ na emocje klienta. Preferowanymi kolorami klientów okazują się barwy chłodne, takie jak niebieski, gdyż wywołują one uczucie relaksu i rozluźnienia w przeciwieństwie do barw ciepłych, takich jak czerwony, wywołujących pobudzenie. Podkreślona została również waga doboru odpowiedniego koloru do ceny produktu.
2. Muzyka: Odpowiednio dobrana muzyka może wpływać na zachowanie klienta. Spokojna, wolna muzyka zachęcała klientów do dłuższego pozostania na terenie sklepu. Dodatkowo, w ramach potwierdzenia postawionej tezy, przywołany został przykład restauracji fast-food oraz sklepu z kwiatami. W fast-foodzie odtwarzana była szybka, żywawa muzyka zachęcającą klientów do sprawniejszego konsumowania posiłku celem szybszego zwolnienia miejsca dla kolejnych osób. Kwaciarnia zaś skupiała się na odtwarzaniu muzyki miłosnej celem zachęcenia klienta do kupna większej ilości kwiatów.
3. Zapach: Trzecią techniką manipulacji jest dobór odpowiedniego zapachu. Podany został przykład temperatury zapachu. Klienci, czujący ciepły zapach, taki jak cynamon, odczuwali większe zatłoczenie, co za tym idzie presję związaną z sprawnym wykonywaniem zakupów, dlatego chętniej kupowali przedmioty mające podnieść ich status w oczach innych klientów. Podane zostały również przykłady uniwersalnych

zapachów mających pozytywny wpływ na klienta, takich jak świeży chleb, ziarna kawy, czy czekolada. Podkreślona została również waga odpowiedniego połączenia zapachu oraz muzyki, celem zwiększenia satysfakcji klienta, co za tym idzie lojalności wobec sklepu.

4. Celebryci: Firmy robiące kampanie reklamowe ze znanym celebrytą odnotowywały znaczny wzrost liczby sprzedanych produktów. Mimo, że klienci są świadomi tej formy manipulacji, to i tak sięgają po reklamowane w ten sposób produkty. Wspomniany został też proces zmiany wyglądu opakowań produktów, celem przyciągania uwagi dzieci, na przykład poprzez dodanie do opakowania postaci ze znanych kresekówek.
5. Reklama szczęścia: Punkt ten skupia się na formie reklamy. Autorzy wskazują na fakt, iż reklamy tworzone są w taki sposób, żeby klient uwierzył, że kupno reklamowego produktu zapewni mu szeroko rozumiane szczęście.

Dodatkowymi formami manipulacji mogą być również: wielkość koszyka zakupowego oraz sam układ sklepu[4]. Większa objętość koszyka zakupowego pozwala na włożenie do niego większej ilości produktów, co w połączeniu z pozostałymi technikami manipulacji sprawia, że klient zachęcony konkretnym produktem ma dodatkowe miejsce, aby go ze sobą zabrać. Strategiczne rozmieszczenie popularnych produktów zwiększa powierzchnię, po której musi poruszać się klient, co zwiększa szansę na kupno dodatkowych przedmiotów.

## 1.2 Istniejące aplikacje

Na rynku dostępne są komercyjne aplikacje zajmujące się szeroko pojętą optymalizacją ułożenia produktów sklepowych. Jednymi z najpopularniejszych dostępnych rozwiązań są LEAFIO AL Retail[5] oraz Mobile Touch[6].

### 1.2.1 LEAFIO AL Retail

LEAFIO AL Retail jest platformą stworzoną przez Leafio AL. Głównym celem aplikacji jest pomoc sklepom w zarządzaniu stanem dostępnego asortymentu oraz wyglądem półek sklepowych. Posiada ona wbudowane narzędzie do wykonywania planogramów dotyczących ułożenia towaru na półkach, co znacznie ułatwia proces zarządzania przestrzenią sklepu. Dodatkowym atutem aplikacji jest moduł wspomagający zarządzanie promocjami,



poprzez analizę ruchu klientów. Firma posiada dodatkowo własny blog[7], na którym dzieli się wynikami współprac oraz publikuje porady związane z zarządzaniem sklepem.

### **1.2.2 Mobile Touch**

Mobile Touch jest rozwiązaniem stworzonym przez Asseco Business Solution skierowanym bezpośrednio do producentów. Aplikacja służy do usprawnienia pracy przedstawicieli handlowych pracujących dla konkretnego producenta. Na podstawie zrobionego przez przedstawiciela zdjęcia program rozpoznaje produkty znajdujące się na półkach, co pozwala ustalić % powierzchni regału, na którym znajduje się towar producenta. Dzięki temu producent otrzymuje narzędzie pozwalające egzekwować, czy dany sklep przestrzega umowy dotyczącej powierzchni przeznaczonej na ekspozycję towaru producenta, na przykład podczas akcji promocyjnej. Dodatkowo posiadając informacje o obecnym rozmieszczeniu towaru na regale, producent otrzymuje możliwość zaproponowania zmiany obecnego planogramu półek w celu poprawienia pozycji własnych produktów.

### **1.2.3 Rekomendacja**

Powyżej opisane aplikacje skierowane są na konkretne segmenty rynku. Leafio skupia się tworzeniu planogramów konkretnych półek, gdzie Mobile Touch pomaga w zarządzaniu towarem konkretnego producenta. Pozostawia to możliwość do wypełnienia luki, w którą wpisuje się stworzony na rzecz tej pracy algorytm. Generuje on globalne ułożenie produktów sklepowych na podstawie ich popularności względem siebie poprzez znalezione w koszykach zakupowych wzorce.

## **1.3 Użyte technologie**

### **1.3.1 Java**

Do napisania algorytmu rekomendacji został wykorzystany język Java[8]. Podyktowane to zostało złożonością problemu oraz liczbą zależności między danymi, które w łatwy sposób mogły zostać przedstawione obiektowo. Dodatkowo język Java posiada dedykowaną bibliotekę do tworzenia interfejsów graficznych JavaFX[9]. Biblioteka ta jest jednym z najbardziej uniwersalnych narzędzi do generowania interfejsów aplikacji niewebowych. Głównym atutem tego rozwiązania jest możliwość dynamicznego generowania elementów

w kodzie, co zostało przedstawione na Listingu 1. Podejście to pozwala na natychmiastowe manipulowanie widocznymi elementami, w zależności od działań użytkownika, bez dodatkowej potrzeby ich szukania po nadanym identyfikatorze. Niweluje to problem rzutowania znalezionej elementu interfejsu z typu ogólnego Node na jego nadany typ np Button, co może prowadzić do błędów oraz trudności z modyfikacją.

```
public static Text createText(String t,String style,double width,double
↪ height,double x,double y) {
    Text text=new Text(t);
    text.getStyleClass().add(style);
    text.setLayoutX(x);
    text.setLayoutY(y);
    text.prefHeight(height);
    text.setWrappingWidth(width);
    return text;
}
```

Listing 1: Kod generujący element interfejsu

Do manipulacji wyglądem interfejsu użyty został CSS[10]. Jest to język służący do tworzenia warstwy wizualnej elementów deklarowanych w HTML oraz w rodzinie XML. Najczęściej zmienianymi parametrami są między innymi: rozmiar obiektu, obwód obiektu, kolor tła, wygląd czcionki czy centrowanie obiektów wewnątrz kontenera. Przykładowa definicja stylu została przedstawiona na Listingu 2. Styl ten zmienia kolor tła obiektu, ustawia rozmiar czcionki używanej wewnątrz obiektu oraz ustala wygląd obwodu.

```
.filterButton{
    -fx-background-color: #5fc9f3;
    -fx-font-size: 14px;
    -fx-border-width: 3px;
    -fx-border-color: #ffcab0;
    -fx-border-style: dashed;
}
```

Listing 2: Kod generujący element interfejsu

CSS pozwala dodatkowo na nasłuchiwanie elementów, pozwalając na dynamiczną zmianę wyglądu, np. w momencie najechania kursorem na przycisk. Style są domyślnie zapisywane w pliku o rozszerzeniu .css i mogą być wstrzykiwane do elementu interfejsu za pomocą funkcji .getStyleClass().add(); tak jak zostało to przedstawione na Listingu 1.

### 1.3.2 Neo4j

Neo4j jest nierelacyjnym systemem bazodanowym opartym na ważonym grafie skierowanym [11]. Grafem nazywamy parę zbiorów  $(V, E)$ , gdzie  $V$  nazywamy zbiorem wierzchołków, a  $E$  zbiorem krawędzi, z których każda łączy ze sobą 2 różne wierzchołki, wyznaczając tym samym relację między nimi [12]. Wagę krawędzi nazywamy liczbę powiązaną z nią, mogącą symbolizować między innymi koszt przejścia między wierzchołkami. Krawędź skierowana oznacza, że można przez nią przejść tylko w jedną stronę. Narzędzie to zostało wybrane, gdyż rozwiązania grafowe są używane do tworzenia systemów rekomendacji w miejscach takich jak media społecznościowe [13], czy portale filmowe [14]. Dodatkowo nadają się one dobrze do reprezentacji zależności w grupie, tworząc tym samym siatkę powiązań.

Neo4j posiada własny język zapytań Cypher [15]. Podstawą każdego zapytania jest składnia  $(:)-[:]->(:)$ .  $()$  oznaczają węzły, a  $[]$  panującą między nimi relację. Strzałka oznacza kierunek relacji. Prawa strona  $:"$  służy do podania nazwy rodzaju relacji/wierzchołka. Lewa strona  $:"$  służy do nazwania znalezionych danych, jeżeli wynik ten będzie używany w dalszej części zapytania. Najważniejsze polecenia zostały opisane poniżej:

1. **CREATE**: Jest to podstawowe polecenie tworzące nowe wierzchołki i krawędzie w grafie. Polecenie **CREATE (:Person{ name:"Tom"})-[:LIKE]->(:Person{ name:"Kate"})** stworzy 2 nowe wierzchołki typu Person, jeden z nazwą Tom, drugi z nazwą Kate i połączy je relacją LIKE w kierunku Tom -> Kate.
2. **MERGE**: Robi to samo co CREATE, ale przed wstawieniem czegokolwiek do bazy sprawdza, czy nie będzie to duplikatem. Jeżeli jest to nie tworzy nowego elementu w bazie i w przypadku tworzenia relacji, używa istniejącego już w bazie wierzchołka.
3. **UNWIND**: Polecenie to rozwija listę do pojedynczych elementów stosując dalszą część polecenie np. **MERGE** wszystkich elementów osobno.
4. **MATCH**: Jest to odpowiednik SELECT z SQL. Pozwala wyszukiwać dane w bazie. Polecenie **MATCH (p:Person)-[:LIKE]->(:Person{ name:"Kate"}) RETURN p.name AS personName** zwróci kolumnę personName z imionami wszystkich osób lubiących Kate
5. **WITH**: Służy do przekazywania wyników między kolejnymi poziomami zapytania.

### 1.3.3 Python

Do przeprowadzenia analizy zachowań napisanego algorytmu został wykorzystany Python. Język ten udostępnia biblioteki takie jak numpy[18] oraz pandas[17], pozwalające na swobodną manipulację danymi, dzięki czemu możliwym było stworzenie różnych scenariuszy testowych dla algorytmu rekomendacji. Dodatkowym atutem tego języka jest obecność biblioteki matplotlib[19], która jest dedykowanym narzędziem do generowania szeroko rozumianych wykresów. Umożliwia ona reprezentację otrzymanych wyników analizy w sposób przejrzysty dla czytelnika.

## 2 Algorytm rekomendacji

### 2.1 Zarys algorytmu

Algorytm rekomendacji ułożenia produktów sklepowych, będący podstawą niniejszej pracy, jest procesem złożonym oraz wielowarstwowym. Składa się on z kilku ściśle powiązanych ze sobą etapów przedstawionych na Listingu 3. Etapy te zostały szczegółowo opisane poniżej.

1. Wczytanie koszyków.
2. Wyznaczenie wzorców wśród wczytanych danych.
3. Wyznaczenie reguł na podstawie pozyskanych wzorców.
4. Stworzenie grafu powiązań wzorców na podstawie wyznaczonych reguł.
5. Zbadanie stopni centralności grafu za pomocą bazy Neo4j.
6. Ocena popularności produktów na bazie wyników Neo4j.
7. Wyznaczenie najkrótszych ścieżek w grafie reprezentującym sklep.
8. Przypisanie produktów do wierzchołków grafu

Listing 3: Zarys algorytmu rekomendacji

Celem stworzonego algorytmu jest zmaksymalizowanie powierzchni, po której musi poruszać się klient, aby odnaleźć i zakupić interesujące go produkty. Założenie to bazuje na wynikach badań[16] dotyczących preferencji zakupowych klientów. Większość badanych wskazała na fakt, iż nie planują oni wcześniej swojej listy zakupowej, więc co za tym idzie, przedmiotów, które chcą nabyć. Dlatego zwiększając powierzchnię, po której musi poruszać się klient, zwiększamy też szansę na kupienie przez niego dodatkowych produktów.

### 2.2 Wczytanie danych

Pierwszym etapem algorytmu jest wczytanie danych wejściowych do programu. Tymi danymi jest lista transakcji, czyli zawartość koszyków zakupowych klientów.

Rysunek 1 przedstawia format pliku z przykładowymi danymi. Jest to plik CSV składający się z dwóch kolumn: kolumny z identyfikatorem koszyka oraz kolumny z jego zawartością. Pojedyncze elementy wewnątrz jednej transakcji zostały rozdzielone za pomocą średnika. Pobranie danych następuje poprzez wczytanie odpowiedniego pliku do programu. Proces ten został przedstawiony na Listingu 4.

```

id,basket
1,Jogurt naturalny;Ser żółty;Pomidory;Mleko
2,Banany;Płatki owsiane;Makaron pełnoziarnisty;Jabłko
3,Ogórki;Makaron pełnoziarnisty;Oliwa z oliwek;Pomidory;Chleb pełnoziarnisty;
4,Ser żółty;Ogórki;Mleko;Makaron pełnoziarnisty
5,Jabłko;Ziemniaki;Płatki owsiane
6,Jogurt naturalny;Chleb pełnoziarnisty;Ziemniaki
7,Ser żółty;Ryż brązowy;Jabłko;Jajka;Chleb pełnoziarnisty
8,Banany;Jabłko;Ser żółty;Ziemniaki;Mleko
9,Oliwa z oliwek;Jajka;Pomidory;Chleb pełnoziarnisty
10,Ziemniaki;Płatki owsiane;Banany;Ser żółty;Oliwa z oliwek;Jabłko;Pomidory

```

Rysunek 1: Przykładowy plik reprezentujący koszyki

Użytkownik otrzymuje okno wyboru pliku. Nagłówki wybranego dokumentu są sprawdzane, aby uniknąć wczytania nieprawidłowych danych. Zapisywane są tylko zawartości koszyków, bez ich indeksów. Celem ujednolicenia formatu, nazwy zapisywane są za pomocą małych liter. Dodatkowo usuwane są potencjalne znaki białe. Pozyskane dane wyświetlane są w przygotowanym dla nich interfejsie graficznym.

```

public void loadFromCSV() {
    try {
        //okno wyboru pliku
        FileChooser fileChooser = new FileChooser();
        fileChooser.setTitle("Wybierz plik zawierający listę koszyków");
        File file = fileChooser.showOpenDialog(null);
        //...
        line = reader.readLine();
        while (line != null) {
            //zapis pojedynczego wiersza
            String[] b = line.split(",")[1].split(";");
            List<String> list = new ArrayList<>();
            for (String s : b)
                //ujednolicenie zapisu
                list.add(s.trim().toLowerCase());
            data.getData().add(list);
            line = reader.readLine();}
        //...
    }
}

```

Listing 4: Kod wczytujący koszyki

## 2.3 Algorytm Apriori

### 2.3.1 Wprowadzenie

Algorytm Apriori jest algorytmem służącym do wyszukiwania wzorców w danych. Popularnym zastosowaniem tego algorytmu jest wyznaczanie reguł asocjacyjnych podczas procesu analizy koszyków zakupowych [20]. Proces ten polega na znajdowaniu takich podzbiorów produktów, że wszystkie produkty znajdujące się wewnątrz podzbioru występują razem z pewną częstotliwością w rozważanych transakcjach. Dodatkowo badane jest oddziaływanie produktów na siebie nawzajem w ramach znalezionych podzbiorów. Taki podzbiór nazywany jest wzorcem i jego długość oznacza się jako  $K$ .

1. Zainicjuj algorytm zbiorem wzorców jednoelementowych
2. Wyznacz Wsparcie kandydujących wzorców.
3. Odrzuć kandydatów niespełniających progu minimalnego Wsparcia.
4. Pozostałych kandydatów oznacz jako wzorce częste.
5. Wygeneruj z nich zbiór kandydatów  $K+1$  elementowych.
6. Jeżeli zbiór nie jest pusty, wróć do punktu 2.

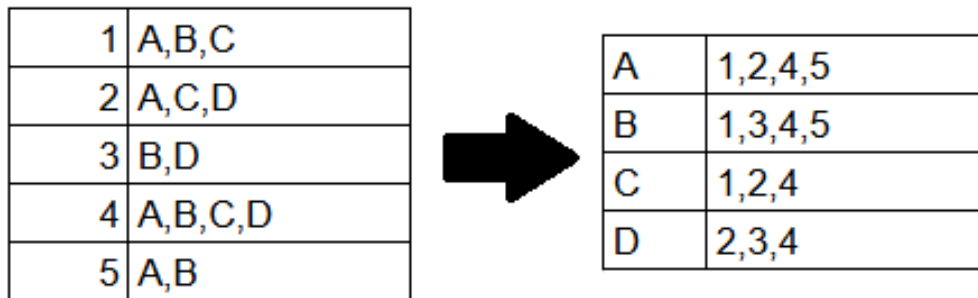
Listing 5: Poszczególne kroki algorytmu Apriori [21]

Listing 5 przedstawia podstawową wersję algorytmu. Poziom Wsparcia ustalany jest przez użytkownika. Oznacza on % transakcji, w którym znajduje się badany wzorec. Wzorec spełniający ustalony przez użytkownika minimalny poziom Wsparcia nazywany jest wzorcem częstym. Wzorcami częstymi dodatkowo są także wszystkie podwzorce powstałe z wzorca częstego.

Podstawowa wersja algorytmu, z powodu zaprezentowanego sposobu generowania kandydatów, jest nieoptymalna. W wersji najprostszej generowane są wszystkie możliwe dwuelementowe sumy zbiorów znalezionych częstych wzorców. Sprawia to, że tworzone są zduplikowane zbiory kandydatów, oraz kandydaci więksi niż  $K+1$ . Przez to, tak zaimplementowane Apriori staje się bardzo szybko niewydatne. Dlatego powstało wiele ulepszeń kodu skupiających się na optymalnym generowaniu zbioru kandydatów.

Jedno z ulepszeń Apriori, nazwane  $T\_Apriori$ [22], zostało zaproponowane przez Xiuli Yuan. Modyfikacja ta zmienia sposób wyliczania Wsparcia kandydatów. Tworzona jest specjalna baza informująca, w których koszykach znajduje się dany produkt. Wzorce znajdujące się w bazie są zbiorem kandydatów. W celu wyznaczenia Wsparcia obliczany

jest iloczyn zbiorów rozpatrywanej pary. Dla przykładu przedstawionego na Rysunku 2 i pary BD mamy:  $(1,3,4,5) \cap (2,3,4) = (2,4)$ . Jeżeli rozmiar zbioru spełnia próg minimalnego Wsparcia, to wzorzec zapisywany jest do bazy jako nowy wzorzec częsty. Podejście to znacznie przyspiesza wyliczanie Wsparcia z powodu mniejszej bazy operacyjnej. Algorytm kończy przebieg gdy liczba kandydatów jest mniejsza od rozpatrywanego obecnie rozmiaru K.



Rysunek 2: Inicjalizacja T\_Apriori

Innym zaproponowanym ulepszeniem jest M-Apriori[21]. Założenie działania algorytmu jest takie samo, jak przy T\_Apriori. Zmienia się sposób wyznaczania kandydatów. Pamiętana jest na stałe tabela dla  $K=1$ . Przy wyliczaniu Wsparcia wzorca sprawdzane jest, który element w rozpatrywanym zbiorze kandydata ma najmniejsze Wsparcie. Element ten wskazuje na transakcje, które należy przeszukać w celu wyliczenia Wsparcia. Algorytm działa do momentu wyzerowania listy kandydatów. Jest to wolniejsze rozwiązanie od T\_Apriori, ale nadal szybsze niż standardowa wersja z powodu szybszego procesu wyznaczania Wsparcia.

### 2.3.2 Implementacja

W pracy zaprezentowana została autorska modyfikacja algorytmu dotycząca sposobu generowania kandydatów. Modyfikacja ta przedstawiona została na Listingu 6. W zaproponowanym rozwiązaniu kandydaci kolejnych epok są generowani rekurencyjnie. Po każdym kroku algorytmu wyznaczana jest lista produktów, które weszły w skład wyznaczonych wzorców częstych ostatnio rozważanej epoki. Funkcja generuje wszystkie możliwe  $K+1$  elementowe zbiory bez powtórzeń podanej listy przedmiotów. Poprawia ona zaś znacząco tempo generowania zbioru kandydatów, niwelując problem związany ze zduplikowanymi wzorcami oraz ich nieodpowiednią długością.



```

//rekurencyjne szukanie kandydatów
public static List<List<String>> generateCandidates(List<String>
↪ candidates, int index, int size, List<String> currCandidate) {
    List<List<String>> patterns = new ArrayList<>();
    //dodanie kandydata do zbioru
    if (currCandidate.size() == size)
    {
        patterns.add(new ArrayList<>(currCandidate));
        return patterns;
    }
    // Warunek zakończenia, przekroczony rozmiar listy
    if (index >= candidates.size())
        return patterns;
    // Krok rekurencyjny
    currCandidate.add(candidates.get(index));
    patterns.addAll(generateCandidates(candidates, index + 1, size,
↪ currCandidate));
    // Backtracking rekurencji
    currCandidate.remove(currCandidate.size() - 1);
    patterns.addAll(generateCandidates(candidates, index + 1, size,
↪ currCandidate));
    return patterns;
}

```

Listing 6: Rekurencyjne szukanie kandydatów

Wprowadzono również zmianę w procesie wyliczania Wsparcia kandydatów. Przed rozpoczęciem przeszukania obecności kandydatów w danym koszyku, wyliczano ile maksymalnie  $K$  elementowych wzorców może się znaleźć w  $N$  elementowym koszyku[32]. Na tej podstawie, jeżeli dla danego koszyka znaleziono wszystkie możliwe  $K$  elementowe podzbiory, to przerywano dalsze badanie obecności kandydatów w tym koszyku oraz przechodzono do kolejnego z nich. Usprawnienie to zależne jest od kolejności wygenerowanych kandydatów. Jeżeli kandydaci, którzy mogą stworzyć wzorec z danym koszykiem znajdują się na końcu zbioru kandydatów, to algorytm nie będzie w stanie obciąć zbędnych operacji porównań niekompatybilnych kandydatów z koszykiem.

## 2.4 Generowanie reguł

### 2.4.1 Wprowadzenie

Reguły asocjacyjne są to wyrażenia mające na celu znalezienie korelacji między wyznaczonymi za pomocą algorytmu Apriori wzorcami[23]. Mają postać logicznej implikacji[24]:  $(A \rightarrow B)$ , gdzie  $(A \cap B) = \emptyset$ , ponieważ mają za zadanie badanie wpływu jednego wzorca na drugi. Pomagają one zrozumieć zachowania klientów, dzięki czemu można świadomie kierować ich decyzjami zakupowymi. W celu ocenienia jakości reguł wprowadzony został szereg metryk[24], mających na celu opis poszczególnych elementów wyrażenia. W celu stworzenia autorskiego algorytmu rekomendacji wybrane zostały następujące metryki[23]:

1. Wsparcie (Support): Informuje w jakim % wszystkich transakcji znajduje się rozpatrywany wzorec. Metryka ta używana jest w trakcie działania wyżej opisanego algorytmu Apriori w celach przesiewu zbioru wzorców kandydujących.
2. Ufność (Confidence): Jest to podstawowa metryka służąca do badania jakości reguł asocjacyjnych. Informuje o % transakcji, w których podczas zakupu wzorca A został także zakupiony wzorec B. Metrykę tą wyznacza się ze wzoru:

$$\text{Ufność}(A \rightarrow B) = \text{Wsparcie}(A \rightarrow B) / \text{Wsparcie}(A)$$

3. Dźwignia (Lift): Jest to metryka wyliczana na podstawie dwóch poprzednich. Informuje ona jak zmieni się prawdopodobieństwo zakupu wzorca B, gdy klient kupuje także wzorec A. Wartości większe od 1 wskazują na pozytywny wpływ na prawdopodobieństwo, podczas gdy wartości mniejsze od 1 go obniżają. Metryka wyliczana jest ze wzoru:

$$\text{Dźwignia}(A \rightarrow B) = \text{Ufność}(A \rightarrow B) / \text{Wsparcie}(B)$$

### 2.4.2 Implementacja

Reguły asocjacyjne tworzone są na podstawie wzorców wyznaczonych przez algorytm Apriori. W ramach przetwarzania pojedynczego wzorca  $W$  generowane są wszystkie podwzorce możliwe do stworzenia na jego podstawie. Są one łączone w reguły  $A \rightarrow B$  takie że:  $(A \cap B = \emptyset) \wedge (A \cup B = W)$ . Listing 7 przedstawia proces wyznaczania omawianych reguł. Podwzorce, na podstawie których tworzone są reguły, generowane są za

pomocą opisanej w podrozdziale wyżej funkcji rekurencyjnej. Zapisywane są tylko te reguły, które spełniają wartości metryk podanych przez użytkownika podczas inicjalizacji algorytmu.

```
// ...
for(int i=1;i<size;i++)
    subsets.addAll(GeneratePattern.generateCandidates
        (pattern.getPattern(),0,i,new ArrayList<>()));
for (List<String> antecedent:subsets){
    //tworzenie prawej strony na zasadzie usunięcia lewej strony z
    ↪ głównego wzorca
    consequent=new ArrayList<>(pattern.getPattern());
    consequent.removeAll(antecedent);
    SimplePattern at = new
        ↪ SimplePattern(antecedent,patternMap.get(getString(antecedent)));
    SimplePattern ct= new
        ↪ SimplePattern(consequent,patternMap.get(getString(consequent)));
    confidence = pattern.getSupport() / at.getSupport();
    lift = confidence / ct.getSupport();
    if(conf<confidence&&lift>1)
        ruleList.add(new AssociationRule(at,ct, pattern.getSupport(),
            ↪ confidence,lift));}
```

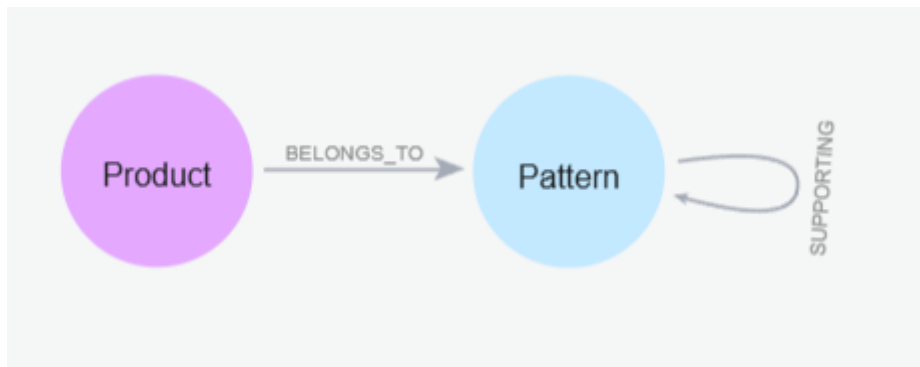
Listing 7: Generowanie reguł asocjacyjnych

Głównym problemem związanym z implementacją tej części algorytmu jest znajdowanie konkretnych podwzorców w wyznaczonym zbiorze wzorców, celem odczytania ich Wsparcia, potrzebnego do wyliczania wartości dalszych metryk. Generuje to potrzebę wielokrotnego wyszukiwania Wsparcia dla tego samego wzorca, co skutecznie zwiększa czas potrzebny na wykonanie algorytmu. Celem rozwiązania tego problemu, zdecydowano się, na potrzeby tej części algorytmu, zareprezentować zbiór wzorców za pomocą mapy. Struktura ta, w przeciwieństwie do listy, ma stały czas dostępu do danych wynoszący  $O(1)$ , co pozwoliło na skuteczne przyspieszenie działania całego algorytmu.

## 2.5 Przetwarzanie za pomocą Neo4j

Kolejnym krokiem algorytmu rekomendacji jest wczytanie wyżej pozyskanych reguł asocjacyjnych do bazy grafowej Neo4j. Jest to robione w celu wyznaczenia stopnia centralności wzorców w stworzonym przez bazę grafie. Pomaga to w określeniu najpopu-

larniejszych przedmiotów w badanym zbiorze. Powstały graf, którego wizualizację przedstawiono na Rysunku 3, składa się z dwóch wierzchołków oraz dwóch relacji. Wierzchołek Product składa się z pojedynczego atrybutu przechowującego nazwę produktu. Wierzchołek Pattern zawiera id wzorca oraz metrykę Wsparcia. Relacja BELONGS\_TO wskazuje, do których wzorców należy dany produkt. Relacja SUPPORTING bezpośrednio przenosi powiązania wzorców poprzez wyznaczone reguły. Zawiera w sobie informacje o metrykach Ufność oraz Dźwignia. Docelowa baza zawiera tylko wierzchołek Pattern oraz relację SUPPORTING. Jest to spowodowane dodaniem do wierzchołka nazwy wzorca, zawierającej wszystkie znajdujące się w nim produkty. Dzięki odpowiedniemu formatowaniu, relacja BELONGS\_TO oraz wierzchołek Product stają się zbędne, gdyż wszystkie informacje znajdują się bezpośrednio w wierzchołku Pattern.



Rysunek 3: Wizualizacja struktury utworzonej bazy grafowej

Dane do bazy grafowej są bezpośrednio wczytywane z aplikacji. Użyte polecenia zostały przedstawione na Listingu 8. Przed wczytaniem danych baza ta jest czyszczona w celu uniknięcia potencjalnych konfliktów z poprzednio zapisanymi w niej danymi. Następnie tworzone są wierzchołki dla wszystkich wyznaczonych wcześniej wzorców. Ostatnim etapem jest wygenerowanie relacji. Dla każdej stworzonej reguły baza wyszukuje odpowiadające jej wzorce i łączy je w pary. W ten sposób powstaje skierowany graf powiązań między danymi, którego przykładowy wycinek został zobrazowany na Rysunku 4.

Po wczytaniu danych następuje ich przetwarzanie. Dla każdego wierzchołka wyznaczany jest stopień centralności, oznaczony w zapytaniu znajdującym się na Listingu 8 jako *connections*, oraz sumaryczna wartość metryki Dźwignia dla wychodzących z wierzchołka krawędzi, oznaczona w poleceniu jako *strength*. Zwracane dane są segregowane malejąco w pierwszej kolejności po kolumnie *connections*, w drugiej zaś po *strength*.

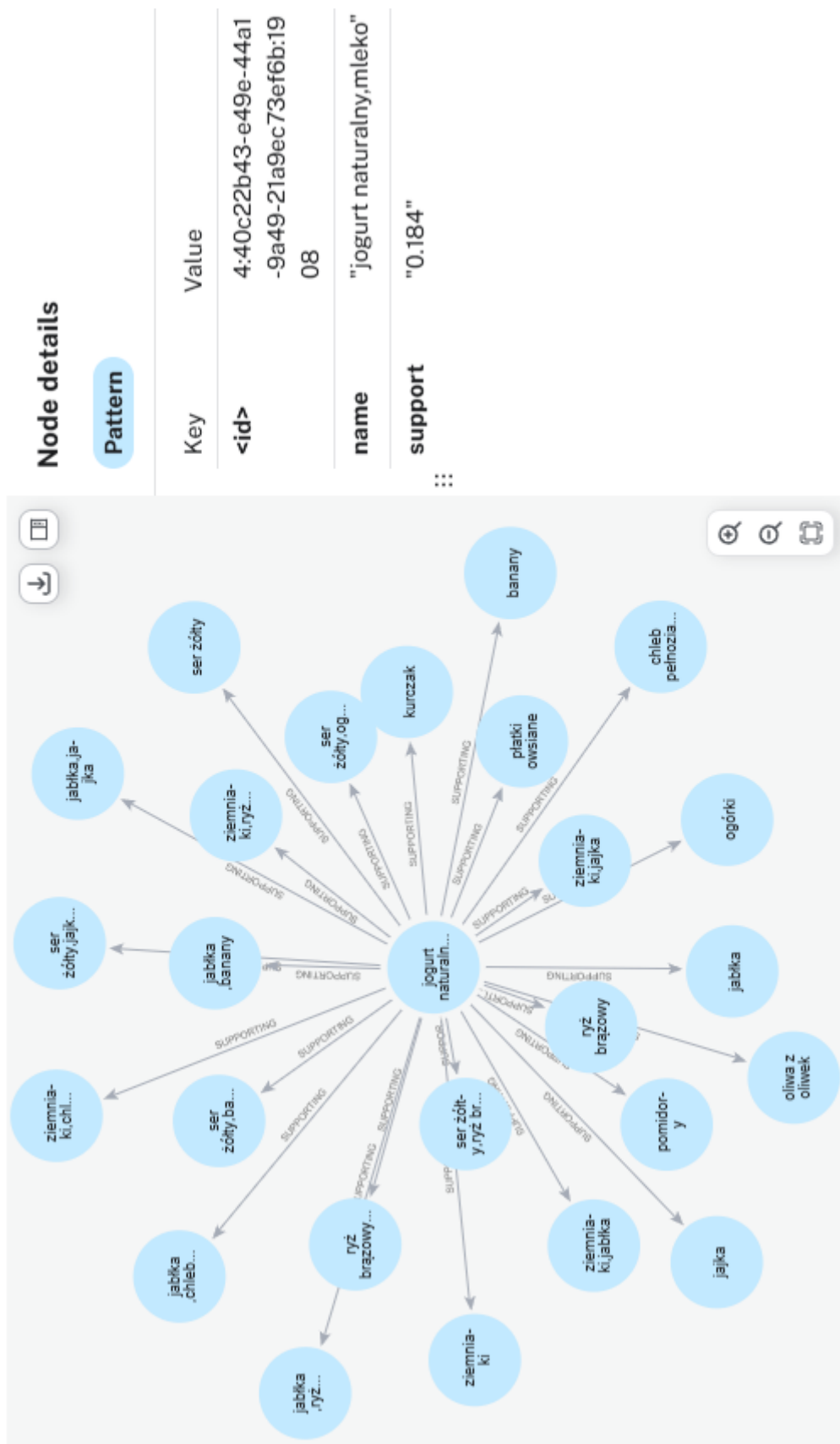
```

//wyczyszczenie bazy
var query = new Query("MATCH (n) DETACH DELETE n;");
//generowanie wierzchołków
var query = new Query("UNWIND $nodes AS node MERGE (p:Pattern {name:
    ↪ node.name, support:node.support})",parameters("nodes", nodes));
//generowanie relacji
var query = new Query("UNWIND $rules AS rule MATCH (left:Pattern {name:
    ↪ rule.antecedent}),(right:Pattern {name: rule.consequent}) MERGE
    ↪ (left)-[:SUPPORTING {confidence: rule.confidence,
    ↪ lift:rule.lift}]->(right)", parameters("rules", ruleData));
//wyznaczanie sąsiedztwa
var query = new Query("""MATCH (n:Pattern)-[s:SUPPORTING]->(m:Pattern)
WITH n, count(m) AS connections, sum(toFloat(s.lift)) AS strength
RETURN n.name AS pattern,n.support AS patternSupport, connections,
    ↪ strength ORDER BY connections DESC, strength DESC;""");

```

Listing 8: Wczytanie danych do bazy grafowej

Głównym celem tej części algorytmu rekomendacji jest wyznaczenie średniej wartości metryki Dźwignia powiązanej z każdym wzorcem. Pozwala to zobrazować globalny wpływ wzorców na siebie nawzajem. Dane z bazy Neo4j zwracane są jednak osobno, jako części do wyznaczenia średniej, a nie jako gotowa średnia. Działanie to ma za zadanie zniwelować wpływ wartości odstających od średniej na finalny wynik rekomendacji. Największe wartości metryki Dźwignia mają zazwyczaj długie wzorce, posiadające niskie Wsparcie, tworzące z powodu swojej długości niewiele reguł. Zwracanie gotowej średniej plasowałoby więc takie wzorce wysoko, co zaburzałoby działanie dalszej części algorytmu. Segregowanie wyznaczonych danych po ich stopniu centralności pozwala na odsianie wyżej wspomnianych, mało znaczących wzorców.



Rysunek 4: Przykład sąsiedztwa

## 2.6 Wyznaczenie Siły produktów

### 2.6.1 Zachowanie ogólne

Następnym krokiem algorytmu rekomendacji jest ocena popularności wszystkich produktów wchodzących w skład rozważanych transakcji. Podstawą tej oceny są dane pozyskane z analizy grafu znajdującego się w bazie Neo4j. Głównymi parametrami brany pod uwagę jest liczba wzorców z danym produktem oraz średnia wartość metryki Dźwignia dla relacji tworzonych przez wzorce, w których te produkty się znajdują.

```
for(Record r:records)
{
    double strength = r.getPatternSupport() *
        ↪ (r.getTotalLift()/r.getConnections())
    String[] patternProducts = r.getName().split(",");
    for(String s:patternProducts)
        productStrength.merge(s, strength, Double::sum);
}
```

Listing 9: Wyznaczanie popularności produktu

Listing 9 przedstawia opisany poniżej proces oceny popularności produktów. Struktura records przechowuje dane bezpośrednio zwrócone przez Neo4j. Załóżmy, że obecnie rozważanym rekordem *r* jest wzorec: {kawa, mleko, cukier}, którego Wsparcie wynosi 0,1 oraz tworzy on 20 relacji o łącznej wartości metryki Dźwignia 25. Prawdą jest więc, że każdy z wyżej wymienionych produktów na mocy relacji (:Product)-[:BELONGS\_TO]->(:Pattern) w 10% wszystkich przypadków wchodzi w skład wzorca, którego średni wpływ globalny na mocy metryki Dźwignia wynosi 1.25. Dlatego do Siły każdego z tych produktów dodawana jest wartość  $0.1 \cdot 1.25$ , czyli Wsparcie rozważanego wzorca \* jego średnia Dźwignia. Wyliczanie Siły w ten sposób w znacznym stopniu niweluje wpływ rzadkich wzorców na jej finalny wynik.

Przykładowe wyniki powyższej funkcji zostały przedstawione na Rysunku 5. Dane zostały posegregowane malejąco względem ustalonej Siły. Widoczna kolejność, będąca podstawą dalszej rekomendacji, w znaczącym stopniu różni się od listy otrzymanej po posegregowaniu produktów względem ich pierwotnego Wsparcia. Możliwa do zaobserwowania jest również wyraźna różnica pomiędzy przedmiotami popularnymi oraz ich mniej chętnie kupowanymi odpowiednikami. Różnica ta nie jest brana pod uwagę przez algo-

Message	Message
Produkt: Siła mleko: 4.904 chleb pełnoziarnisty: 4.745 ziemniaki: 4.682 jogurt naturalny: 4.638 jabłka: 4.609 ser żółty: 4.572 ryż brązowy: 4.518 jajka: 4.483 ogórki: 4.313 banany: 4.138 kurczak: 0.937 oliwa z oliwek: 0.921 płatki owsiane: 0.901 pomidory: 0.845 makaron pełnoziarnisty: 0.566	Produkt: Wsparcie jogurt naturalny: 0.434 mleko: 0.434 jajka: 0.432 ser żółty: 0.428 chleb pełnoziarnisty: 0.414 ryż brązowy: 0.412 jabłka: 0.407 ziemniaki: 0.401 ogórki: 0.396 banany: 0.389 płatki owsiane: 0.210 pomidory: 0.196 oliwa z oliwek: 0.196 kurczak: 0.192 makaron pełnoziarnisty: 0.183

Rysunek 5: Popularność produktów

rytm podczas finalnego rekomendowania. Może być ona za to dodatkową informacją dla użytkownika, pokazującą realną różnicę w zainteresowaniu produktami. Produkty rzadziej kupowane tworzą mniej wzorców inicjujących relację, dlatego też ich prezencja w czasie trwania opisywanej w tym podrozdziale funkcji jest mniejsza, co wpływa na widoczne rozwarstwienie.

### 2.6.2 Przypadki szczególne

Znaleziono zostały 2 przypadki szczególne dotyczące działania wyżej opisanego procesu wyliczania Siły produktów:

Pierwszy z nich dotyczy samych produktów. Jeżeli produkt jest na tyle rzadki, że jego Wsparcie nie jest wystarczające, aby na jego podstawie powstał wzorzec jednoelementowy to taki produkt zostanie zignorowany i nie będzie rekomendowany w dalszej części algorytmu. Jest to zachowanie pożądane, pozwalające na odsianie produktów, którymi zainteresowanie jest zerowe i się nie sprzedają.

Drugi przypadek dotyczy produktów, które nie tworzą wzorców dwuelementowych. Takie produkty są widziane przez algorytm rekomendacji, ale ich Siła jest zerowa, gdyż nie mają jak stworzyć jakiegokolwiek reguły. Z tego powodu ich kolejność na liście re-



komendacji jest losowa, w obrębie produktów zerowej Siły, co zaburza działanie całego algorytmu. Dlatego takim produktom przypisywane jest ich Wsparcie jako wyliczona Siła i produkty te są dopisywane na koniec listy rekomendacji w kolejności malejącej względem przypisanego Wsparcia.

## 2.7 Algorytm Dijkstry

### 2.7.1 Wprowadzenie

1. Zainicjuj listę ścieżek  $D[]$  wartościami  $INF$ , z wyjątkiem punktu  
→ startowego
2. Wybierz nieodwiedzony wierzchołek o najniższej drodze
3. Oznacz wierzchołek jako odwiedzony
4. Dla każdej krawędzi wychodzącej sprawdź czy  $D(a) + wag(ab) < D(b)$
5. Jeżeli tak, dodaj wierzchołek  $b$  wraz z wyznaczoną drogą jako do  
→ sprawdzenia
6. Jeżeli istnieje nieodwiedzony wierzchołek wróć do punktu 2.

Listing 10: Kroki Algorytmu Dijkstry

Algorytm Dijkstry jest algorytmem wyznaczania najkrótszej ścieżki w grafie[25]. Polega on na wyznaczeniu w nieujemnym, ważonym grafie najkrótszej ścieżki ze wskazanego wierzchołka do wszystkich pozostałych. Listing 10 przedstawia pseudokod algorytmu. Bazuje on za założeniu, że do wybranego wierzchołka  $A$  prowadzi zawsze dotychczasowo wyznaczona najkrótsza ścieżka. Główny wpływ na złożoność tego algorytmu ma sposób wyznaczania kolejnego wierzchołka do odwiedzenia. W przedstawionej na Listingu 10 wersji algorytmu, wierzchołki umieszczane są na liście, którą za każdym razem trzeba przejrzeć w całości w celu sprawdzenia najkrótszej ścieżki. Dlatego tak jak w przypadku Apriori powstały modyfikacje podstawowej wersji algorytmu Dijkstry, mające na celu optymalizację tego procesu.

Jednym z zastosowanych ulepszeń algorytmu jest użycie kopca Fibonacciego[27] zamiast zwykłej listy. Kopiec Fibonacciego[26] jest specjalnym rodzajem kopca. Składa się z listy, która przechowuje korzenie drzew, w których znajduje się minimalna wartość klucza. Główną cechą tych drzew jest to, że każde z nich ma inną głębokość. Struktura ma zdolności samo naprawiające. Dodawane oraz usuwane mogą być tylko korzenie. Zmiana struktury sprawia, że drzewa łączą się do momentu, aż każde z nich będzie miało inną

głębokość. Łączenie następuje poprzez porównanie wartości w korzeniach, a nowym korzeniem zostaje węzeł o mniejszej wartości klucza. Drugie drzewo zostaje zaś jego najbardziej prawym potomkiem. Użycie takiego kopca w algorytmie Dijkstry znacznie skraca czas potrzebny na wyznaczenie wierzchołka o obecnie najkrótszej ścieżce.

Inną modyfikacją algorytmu jest zastosowanie kolejek priorytetowych[28]. Bazują one na pojedynczym drzewie binarnym oraz założeniu, że wierzchołek z obecnie najkrótszą wyznaczoną drogą znajduje się w korzeniu. Wtedy główną operacją wykonywaną na kolejce jest naprawa jej struktury po pobraniu wierzchołka.

### 2.7.2 Implementacja

Algorytm rekomendacji ułożenia produktów sklepowych potrzebuje przestrzeni, dla której taka rekomendacja mogłaby zostać wykonana. Z tego powodu układ regałów przykładowego sklepu został zaprezentowany jako graf ważony.

Jest on tworzony przez użytkownika w przygotowanym do tego interfejsie graficznym. Dzięki temu może on być dowolnie modyfikowany i dostosowywany do konkretnych potrzeb, od symulowania pojedynczego regału, gdzie węzeł byłby pojedynczym miejscem na półce, po reprezentowanie całej planszy sklepu, gdzie pojedynczy węzeł stanowiłby cały regał. Wagi krawędzi symbolizują umowną odległość między wybranymi do zasymulowania punktami.

```
while (!pq.isEmpty()) {
    NodeDistance current = pq.poll();
    Node currentNode = current.getNode();
    if (visited.contains(currentNode)) continue;
    visited.add(currentNode);
    for (Edge edge : currentNode.getOutgoingEdges()) {// Aktualizuj
        ↪ sąsiadów
        Node neighbor = edge.getNode();
        if (visited.contains(neighbor)) continue;
        int newDist = distances.get(currentNode) + edge.getWeight();
        if (newDist < distances.get(neighbor)) {
            distances.put(neighbor, newDist);
            pq.add(new NodeDistance(neighbor, newDist));
        }
    }
}
```

Listing 11: Implementacja Algorytmu Dijkstry

W dalszej części algorytmu następuje wyznaczenie najkrótszych ścieżek w grafie, celem ustalenia ostatecznego wyglądu planszy. Implementacja pętli algorytmu Dijkstry została przedstawiona na Listingu 11. Punkt startowy podawany jest przez użytkownika. Do wyznaczania wierzchołków użyta została wbudowana kolejka priorytetowa znajdująca się w Java. Jest to samo organizująca się struktura względem podanego klucza. Dzięki temu, nieodwiedzony wierzchołek o najkrótszej ścieżce jest dostępny na samej górze takiej struktury.

## 2.8 Rekomendacja

Posiadając wyznaczoną wcześniej listę popularności produktów oraz listę ścieżek dostępnych w symulowanym sklepie, można przystąpić do finalnego procesu rekomendacji. Przed rozpoczęciem procesu generowania rekomendacji użytkownik może dodatkowo wczytać do programu plik z kategoriami produktów np. jabłko:owoc, w celu zwiększenia efektywności końcowego wyniku.

Algorytm przypisuje każdemu wierzchołkowi dokładnie jeden produkt. Podczas pojedynczego obrotu algorytmu funkcja w pierwszej kolejności znajduje najdalszy, względem punktu startowego, wierzchołek, do którego nie został przypisany jeszcze żaden produkt. Znalezionejmu punktowi przypisywany jest najpopularniejszy produkt, który nie został do tej pory uwzględniony w grafie.



Sąsiadom rozpatrywanego wierzchołka przypisywane są natomiast przedmioty o niższej popularności. Jeżeli kategorie produktów zostały wczytane, to sąsiedzi zostaną uzupełnieni najrzadziej kupowanymi produktami tej samej kategorii, co przedmiot powiązany z rozpatrywanym wierzchołkiem. Algorytm trwa, dopóki istnieją wierzchołki bez przypisanego produktu. W przypadku, gdy węzłów w grafie jest więcej niż produktów, to po wyczerpaniu listy produktów pobierana jest ona ponownie nie zmieniając przedstawionego działania algorytmu. W takim przypadku dany produkt może zostać przypisany do więcej niż jednego węzła grafu.

Rysunek 6 przedstawia przykładowy wynik algorytmu rekomendacji zawierający Rysunek analizowanego grafu wraz z listą przypisanych do jego wierzchołków produktów. Przedstawione rozwiązanie wykonane zostało dla punktu startowego w wierzchołku o numerze 10. Lista produktów jest wyświetlana względem kolejności uzupełniania grafu przez algorytm. Algorytm działa najlepiej w przypadku, gdy wierzchołki najbardziej oddalone od wybranego punktu startowego są jednocześnie najbardziej oddalone od siebie nawzajem. Sprawia to, że efektywność algorytmu jest zależna od podanego punktu startowego. Dlatego najoptymalniejszym wyborem struktury grafu dla algorytmu rekomendacji jest graf symetryczny, z punktem startowym znajdującym się na osi symetrii. W przypadku przedstawionego na Rysunku 6 grafu odpowiednim punktem startowym byłby wierzchołek 3. Wtedy punkty 11 oraz 15, które są po przeciwnych stronach grafu, byłyby bardzo oddalone od siebie, separując od siebie w znaczącym stopniu najpopularniejsze produkty.



## 3 Aplikacja

### 3.1 Okno główne

W celu przeprowadzenia analizy skuteczności działania algorytmu opisanego w rozdziale powyżej, stworzona została w pełni funkcjonalna aplikacja okienkowa. Pozwala ona wczytywać i modyfikować dane, na których algorytm wykonuje obliczenia.

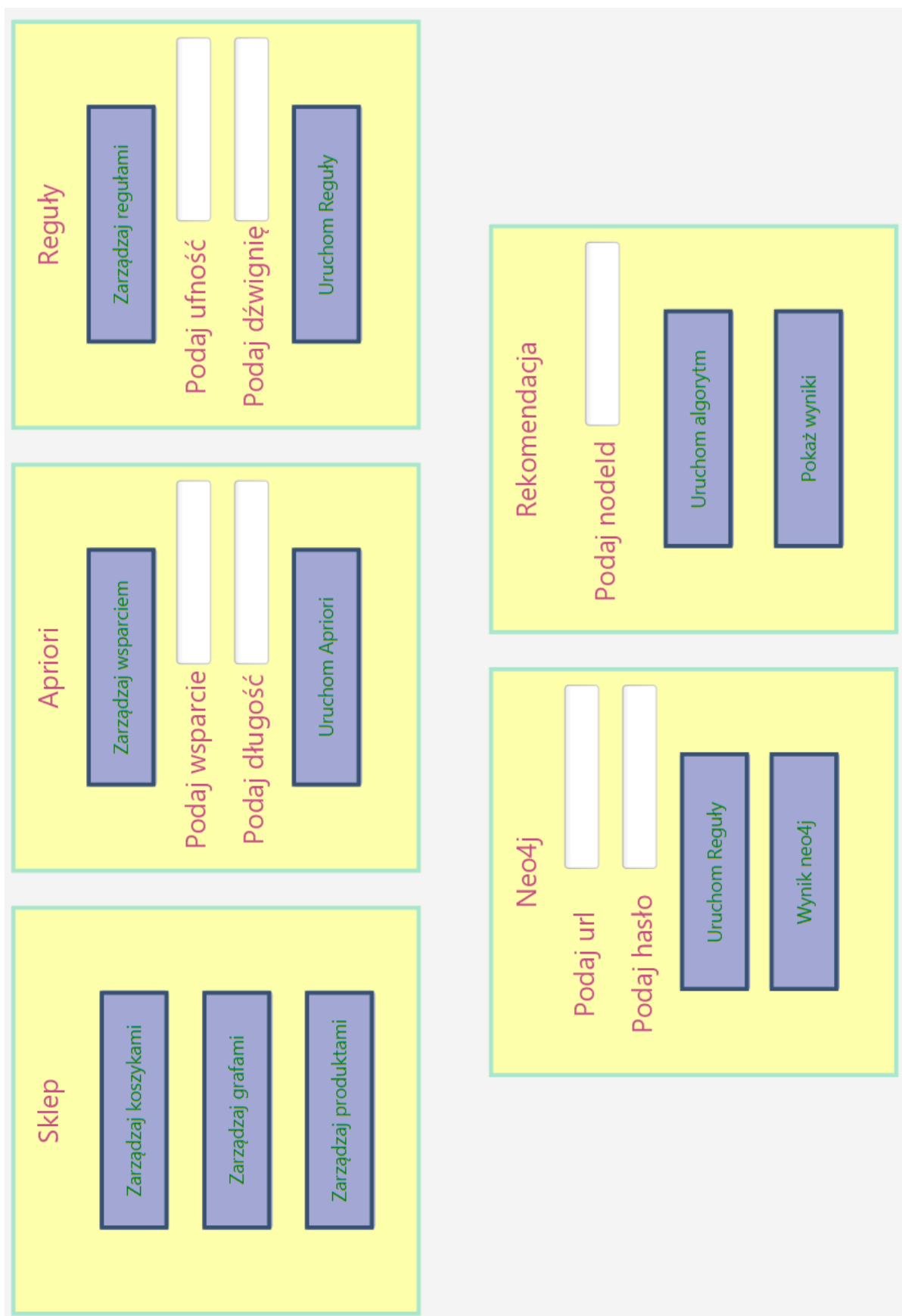
Rysunek 7 przedstawia główne okno aplikacji. Zostało ono podzielone na 5 tematycznych modułów. Każdy z nich odpowiada za konkretną część algorytmu rekomendacji.

Moduł SKLEP zajmuje się zarządzaniem produktami oraz powierzchnią sklepu. Udostępnia on w pełni funkcjonalny, graficzny interfejs do tworzenia grafu reprezentującego sklep, interfejs pozwalający zarządzać ilością i zawartością koszyków oraz miejsce do wczytania kategorii rekomendowanych produktów.

Moduły APRIORI i REGUŁY służą do obsługi wzorców oraz reguł powstałych na podstawie wczytanych koszyków. Udostępniają one pola, dzięki którym użytkownik może podać minimalne wartości metryk dla wyliczanych danych. Dla długości wzorca podawana jest liczba naturalna będąca minimalną jego długością. Dla pozostałych metryk podawana jest liczba wymierna. Dodatkowo udostępniany jest interfejs pozwalający na modyfikację wyznaczonych danych, tak samo jak w przypadku koszyków.

Moduł NEO4J służy do połączenia się z instancją bazy, na której będą wykonywane obliczenia. Do obliczeń wykorzystana została darmowa instancja Neo4j AuraDB[29], dostępna po założeniu konta na stronie internetowej. Pola tekstowe służą do podania danych dostępowych do bazy, które są otrzymywane wraz ze stworzeniem instancji. Pole URL służy do podania samego id bazy. Pole HASŁO służy do wpisania wygenerowanego przez system hasła do bazy. Brak podania danych sprawi, że wykorzystana zostanie baza, dla której wykonywane były testy aplikacji. Za pomocą przycisku *Wynik neo4j* tworzony jest nowy plik tekstowy, w którym zapisywane są wyliczone poziomy Siły rekomendowanych produktów.

Ostatnie pole REKOMENDACJA służy do uruchomienia ostatniej części algorytmu rekomendacji. Przed uruchomieniem procesu użytkownik podaje id wierzchołka w celu wyznaczenia ścieżek w grafie sklepowym. Przycisk *Pokaż wyniki* wyświetla okno z ukończoną rekomendacją, które zostało przedstawione we wcześniejszym podrozdziale.



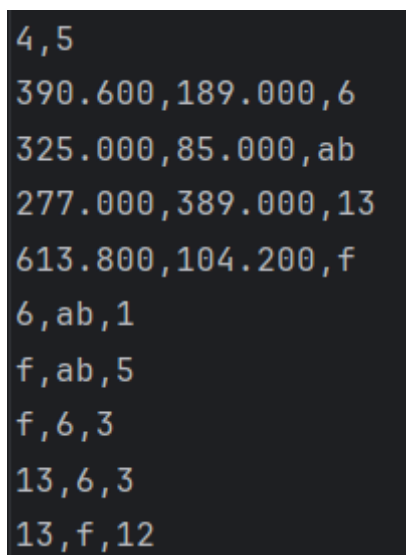
Rysunek 7: Okno główne Aplikacji



## 3.2 Interfejs grafowy

### 3.2.1 Funkcjonalności

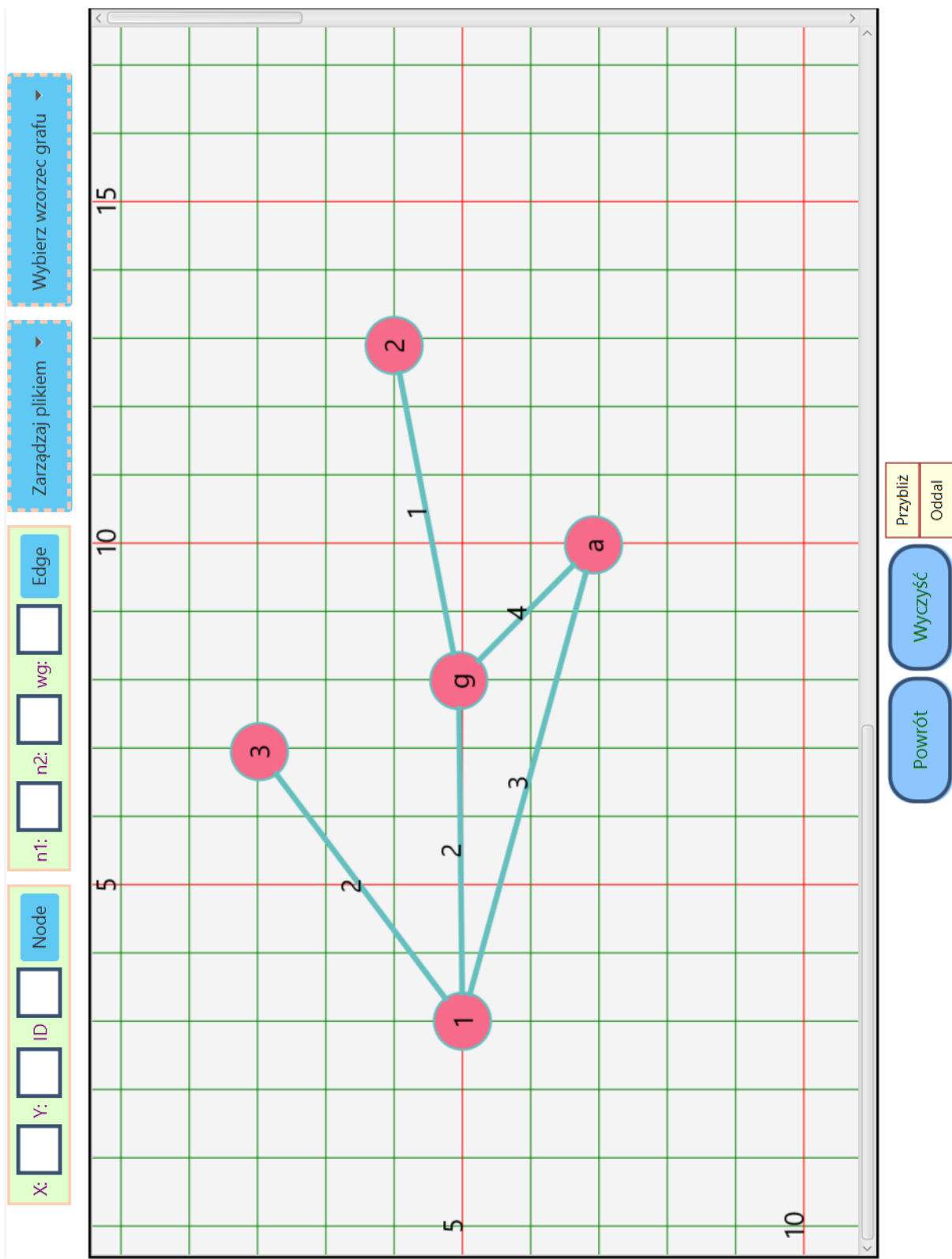
Rysunek 9 przedstawia wygląd interfejsu tworzenia grafu. Siatka, na której graf jest rysowany, została zbudowana na układzie współrzędnych. Górna belka interfejsu zawiera okna obsługujące dodawanie elementów do struktury grafu. Każdy nowy wierzchołek może być dodany na nieujemnych, całkowitych współrzędnych, na których lokalizacje wskazują punkty przecięcia linii na siatce. Jako ID może zostać podany dowolny ciąg znakowy. Zważywszy jednak na rozmiar rysowanych punktów zaleca się podawać ciąg o długości co najwyżej dwóch znaków. Krawędzie rysowane są po podaniu ID stworzonych już wierzchołków, które mają zostać połączone.



```
4,5
390.600,189.000,6
325.000,85.000,ab
277.000,389.000,13
613.800,104.200,f
6,ab,1
f,ab,5
f,6,3
13,6,3
13,f,12
```

Rysunek 8: Format pliku grafowego

Podstawową funkcjonalnością interfejsu jest możliwość zapisu i wczytania tworzonych grafów z pliku. Przy wyborze zapisu użytkownik zostaje poproszony o podanie nazwy zapisywanego pliku. Plik zostanie zapisany jako *nazwa\_graph.csv*. Przy wczytaniu pliku zaś użytkownik otrzyma okno dialogu celem wybrania odpowiedniego pliku. Format pliku został pokazany na Rysunku 8. Nagłówek pliku wskazuje odpowiednio na liczbę wierzchołków oraz krawędzi definiowanej struktury. Dla wierzchołków zapisane są współrzędne środka koła reprezentującego wierzchołek na planszy oraz jego indeks. Dla krawędzi zapisane są indeksy wierzchołków, które są połączone za pomocą krawędzi oraz jej waga.



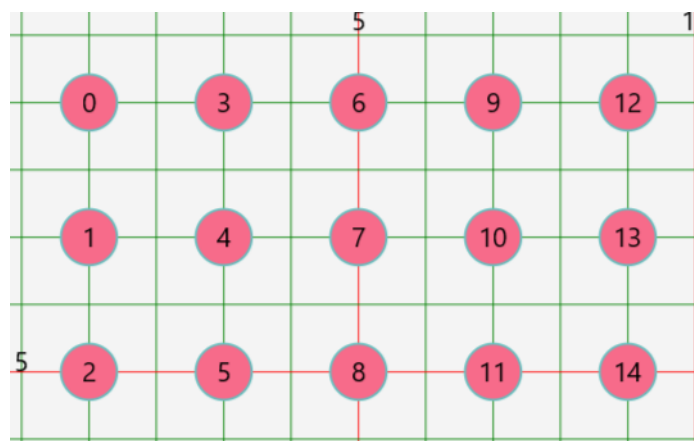
Rysunek 9: Interfejs Grafowy

Dolna belka interfejsu, widoczna na Rysunku 9, pozwala na wyczyszczenie całej planszy wraz z zapisanymi danymi oraz na powrót do okna głównego. Dodatkowo, pozwala ona na przybliżanie oraz oddalanie okna, w przypadku gdy podstawowo zdefiniowane miejsce nie wystarcza do wyświetlenia obecnie tworzonego grafu. Usuwanie elementów odbywa się za pomocą podwójnego kliknięcia na element. W przypadku uruchomienia tej opcji, użytkownik zostanie poproszony o potwierdzenie chęci usunięcia elementu. W przypadku usunięcia wierzchołka usunięte zostaną również wszystkie krawędzie powiązane z nim. Kliknięcie bezpośrednio na wagę krawędzi sprawi, że uruchomi się okno dialogowe zmiany wartości tejże. Indeks raz zdefiniowanego wierzchołka nie podlega zmianie. Ostatnią funkcjonalnością jest możliwość dowolnego przesuwania wierzchołków po planszy po złapaniu ich za pomocą kursora myszki.

### 3.2.2 Szablony grafów

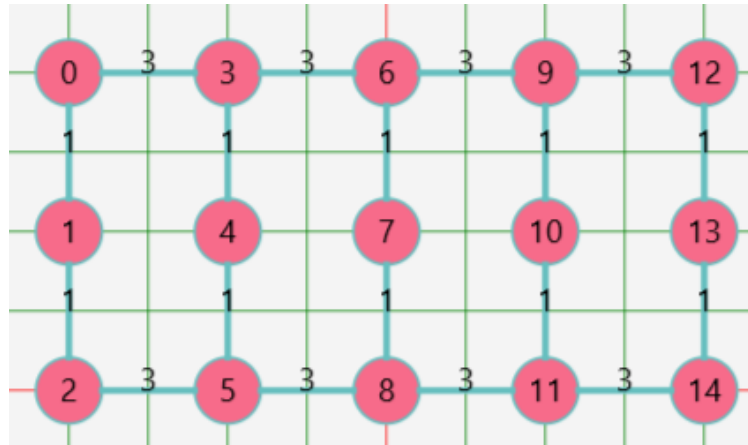
Zdefiniowany interfejs nie pozwala na wstawianie kilku elementów jednocześnie, co znacznie utrudnia oraz wydłuża czas ręcznego budowania dużych, użytecznych grafów. Dlatego powstała funkcja generująca graf z predefiniowanych szablonów. Szablony te zostały zdefiniowane poniżej:

1. Graf bez połączeń: Jest to najprostszy szablon dostępny w interfejsie. Generuje prostokątną siatkę wierzchołków, w której nie określono żadnej krawędzi pomiędzy nimi.. Tworzy się ją, podając w oknie dialogowym wartości  $A \times B$ , gdzie A B to odpowiednio liczbę kolumn oraz liczbę wierszy definiowanego pola. Rysunek 10 pokazuje strukturę, wygenerowaną za pomocą polecenia  $5 \times 3$ .



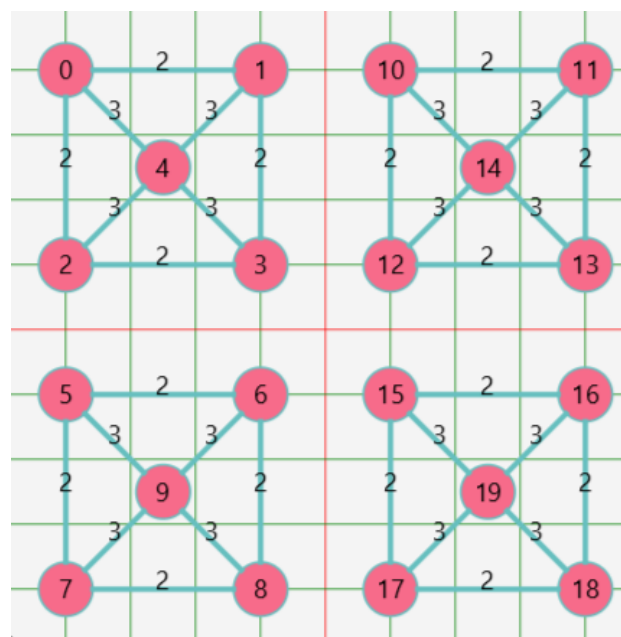
Rysunek 10: Graf bez połączeń

2. Rzędy regałów: Pokazany na Rysunku 11 szablon działa na tej samej zasadzie, co wzorzec poprzedni. Różnicą jest dodanie krawędzi łączących wierzchołki w kolumnach oraz na obwodzie pola.



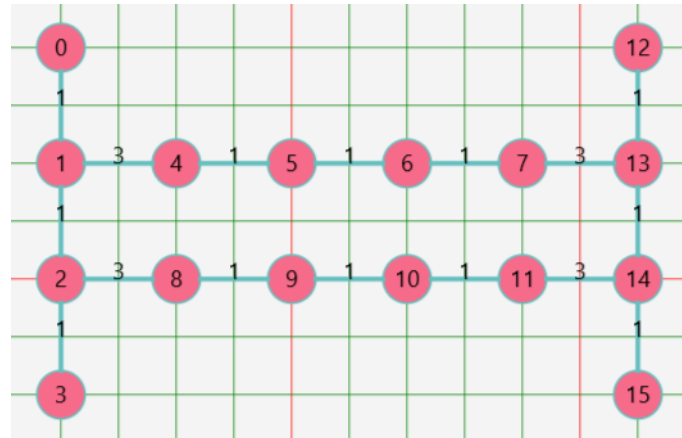
Rysunek 11: Rzędy regałów

3. Wyspy promocyjne: Jest to szablon mający reprezentować ekspozycje promocyjne na tymczasowych wystawach w sklepach. Składa się on z wielu niezależnych kwadratów z dodatkowym wierzchołkiem znajdującym się na przecięciu przekątnych. W celu stworzenia szablonu należy podać liczbę kwadratów. Rysunek 12 przedstawia strukturę zbudowaną z 4 kwadratów



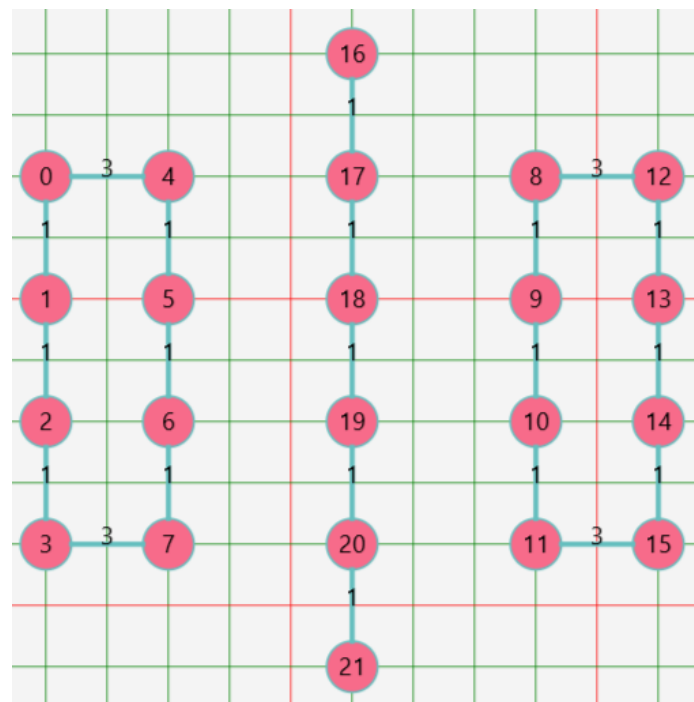
Rysunek 12: Wyspy promocyjne

4. Regały prostopadłe: Jest to szablon tworzący między dwoma półkami linie regałów do nich prostopadłych. Formatem tworzenia jest  $AxB$ , gdzie A B to odpowiednio liczbę regałów prostopadłych i ich długość. Rysunek 13 przedstawia strukturę 2x4.



Rysunek 13: Aleja prostopadła

5. Schowki: Ostatni szablon można interpretować jako jedną, długą, główną aleję z wystawionymi rzeczami promocyjnymi, obok której, na półkach znajdujących się na wejściach do poszczególnych alejek są rozmieszczone dodatkowe towary. Szablon tworzy się przez podanie liczby półek. Rysunek 14 przedstawia szablon stworzony z czterech półek.



Rysunek 14: Schowki

Powyższe szablony są tylko sugestią, w jaki sposób może wyglądać przykładowy graf. Są one komponentami, które mogą być łączone ze sobą w dowolny sposób, celem uzyskania zadowalającego układu sklepu. Przed uruchomieniem algorytmu należy dodatkowo sprawdzić, czy wszystkie komponenty zostały połączone w spójny graf, w przeciwnym razie istnieje ryzyko zignorowania części układu przez system rekomendacji. Okno łączenia komponentów zostało przedstawione na Rysunku 15. Współrzędne generowania komponentu podane są za pomocą wyrażenia  $X;Y$ .

Rysunek 15: Dodawanie komponentu do grafu

### 3.3 Interfejs koszykowy

#### 3.3.1 Kategorie Produktów

<input type="text"/>	Wybierz opcję ▼	Powrót	Wczytaj produkty	Wyczyść produkty
Produkt	Kategoria			
mleko	zwierzęce			
makaron pełnoziarnisty	zboża			
pomidory	owoce i warzywa			
chleb pełnoziarnisty	zboża			
płatki owsiane	zboża			
oliwa z oliwek	owoce i warzywa			

Rysunek 16: Kategorie produktów

Interfejs koszykowy składa się z kilku niezależnych okien. Na Rysunku 16 przedstawione zostało okno umożliwiające zarządzanie kategoriami produktów. Wprowadzone dane są wyświetlone za pomocą tabeli. Przyciski znajdujące się na górnej belce, umożliwiają użytkownikowi wyczyszczenie danych, wprowadzenie danych z pliku CSV oraz ręczną ich modyfikację. Format pliku przechowującego kategorie produktów został przedstawiony na Rysunku 17. Ręczne dodanie kategorii następuje poprzez wprowadzenie do pola tekstowego wyrażenia *produkt,kategoria*. Poskutkuje to dodaniem pojedynczego wiersza do wyświetlanej tabeli. Ręczne usunięcie pojedynczego elementu zaś możliwe jest poprzez wpisanie nazwy produktu do tego samego pola i wybranie opcji usuń.

```
id, product, category
1 , mleko, zwierzęce
2 , makaron pełnoziarnisty, zboża
3 , pomidory, owoce i warzywa
4 , chleb pełnoziarnisty, zboża
5 , płatki owsiane, zboża
6 , oliwa z oliwek, owoce i warzywa
7 , ziemniaki, owoce i warzywa
```

Rysunek 17: Format pliku z kategoriami

Wartym do zaznaczenia faktem jest to, że opisywane okno nie jest w żaden sposób powiązane z wczytanymi do programu koszykami. Sprawia to, iż możliwe jest dodanie w tym miejscu dowolnego produktu, wraz z powiązaną z nim kategorią, także takiego, który nie istnieje w rozważanych koszykach zakupowych. Dane takie zostają zignorowane w trakcie końcowego etapu rekomendacji, nie wpływając na jej przebieg.

### 3.3.2 Koszyki zakupowe

Rysunek 18 przedstawia wygląd interfejsu koszykowego. Pozwala on na wczytywanie, przeglądanie i modyfikacje danych. Dane wyświetlane są w liście przewijanej, po której użytkownik może się dowolnie poruszać za pomocą pola znajdującego się w prawej części dolnej belki interfejsu. Po lewej stronie belki natomiast użytkownik może zmienić liczbę jednorazowo wczytanych do listy elementów. Dostępne rozmiary wynoszą odpowiednio: 10, 25, 50, 100 oraz 250. Nagłówek tabeli umożliwia sortowanie koszyków po ich długości. Sortowanie możliwe jest zarówno malejąco jak i rosnąco.

Wybierz Filtr

Zarządzaj filtrami

Wczytaj koszyki

Wyczyść koszyki

Koszyk	↑
<input type="checkbox"/>	ziemniaki; mleko; ogórki; chleb pełnoziarnisty; jabłka; jogurt naturalny;
<input type="checkbox"/>	ryż brązowy; oliwa z oliwek; jogurt naturalny; banany;
<input type="checkbox"/>	ser żółty; oliwa z oliwek; pomidory; jajka; ryż brązowy; chleb pełnoziarnisty;
<input type="checkbox"/>	makaron pełnoziarnisty; jogurt naturalny; ogórki; ryż brązowy; jabłka; banany; oliwa z oliwek;
<input type="checkbox"/>	ryż brązowy; oliwa z oliwek; jajka; ser żółty; ogórki;
<input type="checkbox"/>	chleb pełnoziarnisty; jabłka; ryż brązowy; jogurt naturalny; ogórki; ziemniaki;
<input type="checkbox"/>	makaron pełnoziarnisty; ziemniaki; jajka; mleko; ser żółty; jabłka; płatki owsiane;
<input type="checkbox"/>	mleko; płatki owsiane; chleb pełnoziarnisty; kurczak; banany;
<input type="checkbox"/>	ser żółty; jogurt naturalny; banany; ziemniaki; ryż brązowy; chleb pełnoziarnisty; ogórki; jajka;
<input type="checkbox"/>	ryż brązowy; mleko; oliwa z oliwek; jabłka; kurczak; jogurt naturalny;
<input type="checkbox"/>	ziemniaki; kurczak; jajka; ogórki;
<input type="checkbox"/>	oliwa z oliwek; kurczak; mleko; ser żółty; jogurt naturalny; jajka;
<input type="checkbox"/>	chleb pełnoziarnisty; banany; jabłka; jajka; kurczak; ziemniaki;
<input type="checkbox"/>	ogórki; płatki owsiane; ziemniaki; jogurt naturalny;

Pokazuj

100

▼

elementów

Powrót

<

Pokazano 231-330 z 1000 elementów

>

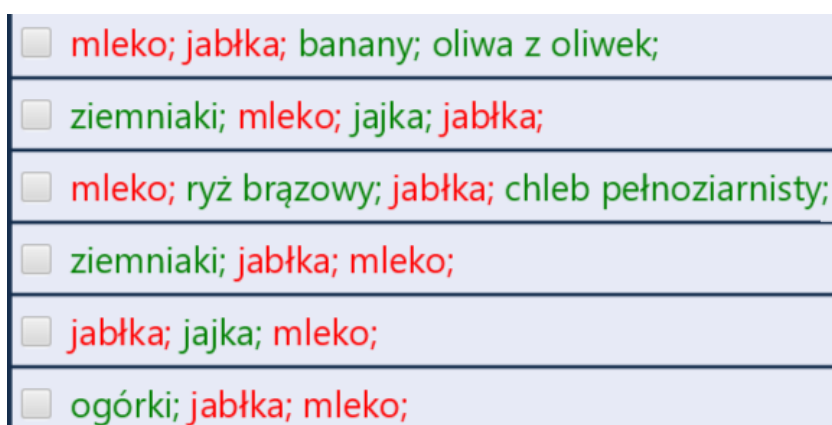
Rysunek 18: Interfejs Koszykowy



Przycisk *Zarządzaj filtrami* udostępnia dodatkowe możliwości związane z modyfikacją wczytanych danych. Są to następujące opcje:

1. Wyczyść filtry: Przywraca podstawowy widok danych, czyszcząc wszystkie dane filtrów zapisanych w tle.
2. Zaznacz wszystkie boxy: Zaznacza checkboxy wszystkich koszyków znajdujących się w obecnym widoku listy. Interfejs nie udostępnia jednak opcji odznaczenia tychże. Odznaczenie następuje w momencie zmiany wyświetlanego okna w liście.
3. Usuń zaznaczone wiersze: Usuwa na stałe z aplikacji oznaczone koszyki. W połączeniu z poprzednią opcją pozwala to na sprawne usuwanie całych okien danych.
4. Usuń wybrane elementy: Dostępna tylko w momencie aktywnego filtru nałożonego na dane. Usuwa ona z zaznaczonych koszyków, tylko te produkty, po których nastąpiło wyszukiwanie, czyli te zaznaczone w momencie usuwania na czerwono. Reszta koszyka pozostaje zapisana bez zmian.

Opcje te, w swoim założeniu, mają za zadanie pozwolić na prostą obróbkę danych przez rozpoczęciem właściwego procesu rekomendacji. Umożliwiają usunięcie wartości odstających, niepotrzebnych produktów, czy też koszyków konkretnej długości.



Rysunek 19: Wygląd filtrowanego koszyka

Górna belka interfejsu udostępnia narzędzia pozwalające na filtrowanie wczytanych danych. Pole tekstowe pozwala na wprowadzenie do programu filtru. Filtrowanie danych jest możliwe zarówno po długości koszyka, jak i jego zawartości. Aktywny może być jednocześnie tylko jeden filtr na raz. Zasada ta tyczy się wszystkich interfejsów znajdujących

się w aplikacji. Produkty, po których filtrowane są koszyki, powinny przy wprowadzaniu zostać oddzielone średnikiem. Rysunek 19 pokazuje koszyki po skończonym procesie wyszukiwania. Filtrowane dane zostały oznaczone na czerwono.

### 3.3.3 Interfejs reguł i Apriori

Rysunek 20 przedstawia interfejs reguł. Działa on na tej samej zasadzie co opisany podrozdział wyżej interfejs koszykowy. Główną różnicą jest wyświetlanie metryk związanych ze wzorcem/regułą i możliwość filtrowania po nich. Nagłówki tabeli również udostępniają możliwość sortowania po wartościach metryk. Interfejs reguł, w przeciwieństwie do koszykowego, pozwala na zapis wyznaczonych danych do pliku. Zabrana została dodatkowo opcja usuwania filtrowanych danych. Spowodowane jest to faktem, iż wzorce, a tym bardziej reguły są ściśle powiązane ze sobą, co sprawia że niewłaściwe usuwanie danych w tym miejscu może spowodować, że algorytm rekomendacji nie będzie się w stanie wykonać z powodu błędów w danych. W takim wypadku należy usunąć wszystkie dane związane z problematycznym produktem.

	Wybierz Filtr	Zarządzaj filtrami	Zarządzaj CSV	Wyczyść bazę
Reguła	Wsparcie	Ufność	Dźwignia	
<input type="checkbox"/> bottled water; -> brown bread;	0.038	0.178	1.306	
<input type="checkbox"/> brown bread; -> bottled water;	0.038	0.279	1.306	
<input type="checkbox"/> misc. beverages; -> pastry;	0.013	0.220	1.238	
<input type="checkbox"/> berries; -> pip fruit;	0.018	0.225	1.316	
<input type="checkbox"/> pip fruit; -> berries;	0.018	0.105	1.316	
<input type="checkbox"/> butter; -> coffee;	0.018	0.143	1.242	
<input type="checkbox"/> coffee; -> butter;	0.018	0.157	1.242	
<input type="checkbox"/> hard cheese; -> shopping bags;	0.012	0.226	1.348	
<input type="checkbox"/> pet care; -> other vegetables;	0.010	0.455	1.206	
<input type="checkbox"/> white wine; -> sausage;	0.013	0.295	1.434	
<input type="checkbox"/> meat; -> canned beer;	0.015	0.234	1.420	
<input type="checkbox"/> sugar; -> frankfurter;	0.011	0.167	1.208	
<input type="checkbox"/> beef; -> pip fruit;	0.025	0.208	1.218	
<input type="checkbox"/> pip fruit; -> beef;	0.025	0.146	1.218	
<input type="checkbox"/> canned beer; -> frozen vegetables;	0.024	0.145	1.412	

Pokazuj 100 elementów

Powrót

Pokazano 1-100 z 31189 elementów

Rysunek 20: Interfejs reguł



## 4 Analiza

W celu przetestowania działania algorytmu rekomendacji wykorzystano dwa różne zbiory danych dostępne na portalu Kaggle[30][31] oraz kilka zbiorów wygenerowanych losowo. Pozwoliło to na stworzenie różnych scenariuszy testowych, pozwalających na dokładne przebadanie zachowań algorytmu.

### 4.1 Przedstawienie zbiorów

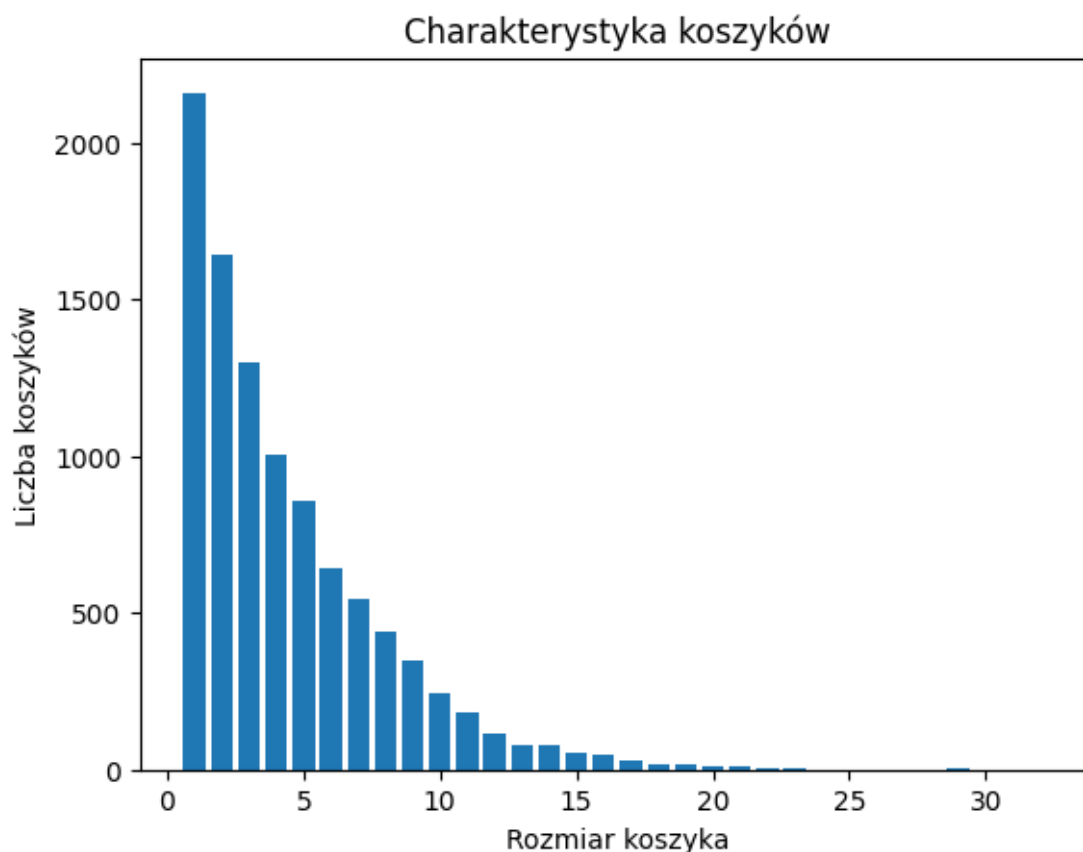
#### 4.1.1 Zbiór pierwszy

Tabela 1 przedstawia fragment ramki pierwszego z wybranych zbiorów[30]. Składa się on z 9835 wierszy oraz 33 kolumn. Każdy wiersz reprezentuje pojedynczy koszyk zakupowy. Każda kolumna zaś, oprócz kolumny pierwszej przechowującej rozmiar rozważanego koszyka, reprezentuje pojedynczy przedmiot znajdujący się w nim. Z tego powodu liczba kolumn w ramce musi być równa rozmiarowi największego koszyka znajdującego się w zbiorze. Taki sposób przechowywania danych, przy dużych wahaniami rozmiarów koszyków, jest bardzo nieefektywny, gdyż generuje sporą liczbę wartości pustych znajdujących się w tak zapisanej ramce danych.

Item(s)	Item 1	Item 2	...	Item 31	Item 32
4	citrus fruit	semi-finished bread	...	NaN	NaN
3	tropical fruit	yogurt	...	NaN	NaN
1	whole milk	NaN	...	NaN	NaN
4	pip fruit	yogurt cream	...	NaN	NaN
4	other vegetables	whole milk	...	NaN	NaN

Tabela 1: Ramka danych zbioru pierwszego

Rysunek 21 przedstawia wykres reprezentujący rozmiar i liczbę koszyków dla pierwszego zbioru testowego. Dominującą liczebnością produktów dla rozważanych koszyków zakupowych okazał się pojedynczy produkt, co jest interesującym wynikiem z perspektywy algorytmu rekomendacji. Koszyki jednoelementowe w procesie analizy koszykowej nie mogą stworzyć reguł, ponieważ na ich podstawie nie może powstać wzorec conajmniej dwuelementowy. Ich obecność w zbiorze powinna obniżyć Wsparcie wszystkim wyznaczanym wzorcom, oddziałując tym samym na wyznaczaną Siłę produktów. Wartym do odnotowania jest również fakt znajdowania się w zbiorze pojedynczych, dużych koszyków.



Rysunek 21: Charakterystyka koszyków zbioru pierwszego

Ich długość sprzyja powstawaniu dużej liczby krótkich wzorców, co powinno zostać zaobserwowane w wartości Wsparcia najpopularniejszych wzorców. Realny wpływ koszyków o skrajnych długościach, nazywanych dalej odstającymi, na algorytm został zbadany oraz opisany w dalszej części rozdziału.

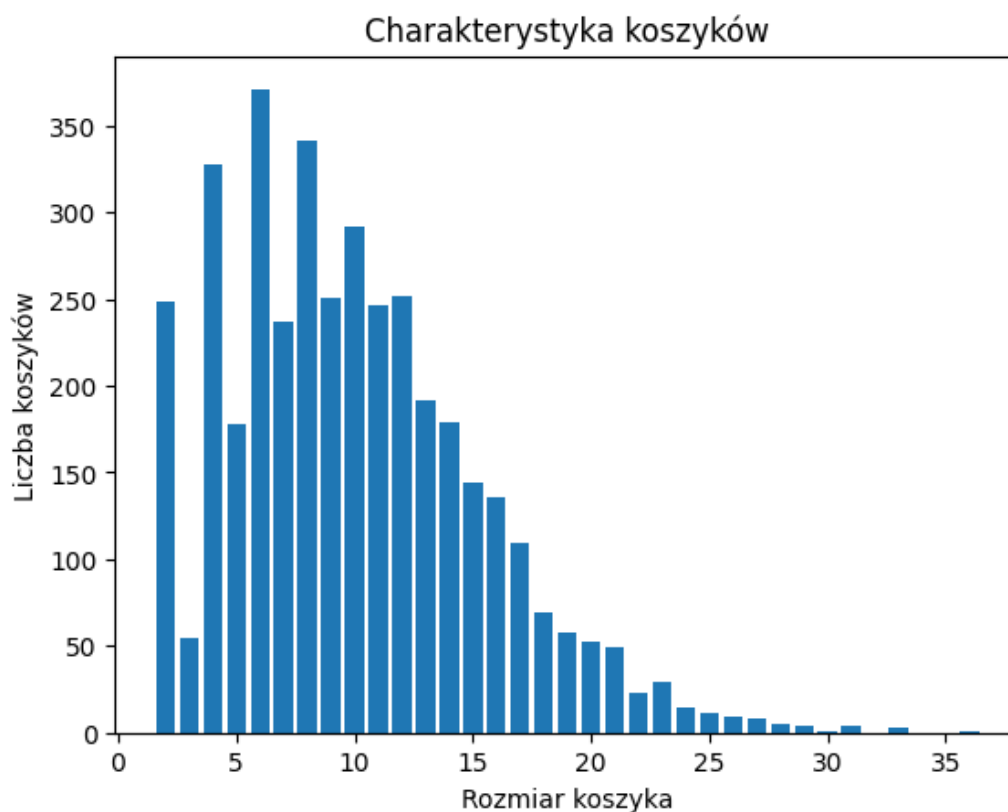
#### 4.1.2 Zbiór drugi

Member_number	Date	itemDescription
1808	21-07-2015	tropical fruit
2552	05-01-2015	whole milk
2300	19-09-2015	pip fruit
1187	12-12-2015	other vegetables
3037	01-02-2015	whole milk
4941	14-02-2015	rolls/buns
4501	08-05-2015	other vegetables

Tabela 2: Ramka danych zbioru drugiego

Tabela 2 przedstawia fragment ramki drugiego z wybranych zbiorów[31]. Składa się ona z 38765 wierszy oraz 3 kolumn, gdzie każdy wiersz reprezentuje pojedynczy produkt kupiony w ramach konkretnej transakcji. Jest to optymalniejszy sposób przechowywania danych, gdyż nie generuje wartości pustych, w przeciwieństwie do zbioru pierwszego.

Rysunek 22 przedstawia wykres reprezentujący rozmiar oraz liczbę koszyków zawartych w zbiorze drugim. Zbiór ten posiada łącznie 3898 unikalnych koszyków. Należy zwrócić uwagę na kształt rozkładu danych wyznaczonego wykresu. Jest on rozkładem prawoskośnym, który osiąga szczytową wartość dla koszyka równego 6. Jest to znacząca różnica w porównaniu ze zbiorem pierwszym, dla którego rozkład danych wygląda jak wykres funkcji homograficznej  $\frac{1}{x}$ .



Rysunek 22: Charakterystyka koszyków zbioru drugiego

Różnica w kształcie rozkładów danych wraz z ponad dwukrotnie mniejszą liczbą koszyków między zbiorami widoczna jest w średniej liczbie produktów zawartych w pojedynczym koszyku. Dla zbioru pierwszego wyniosła ona 4.41. Dla zbioru drugiego zaś liczba ta jest równa 9.97, co jest ponad dwukrotnie większym wynikiem. Wpływ średniej liczby przedmiotów w koszyku na liczbę wyznaczanych wzorców została zbadana i opisana w dalszej części rozdziału.

## 4.2 Zachowania Wsparcia

### 4.2.1 Koszyki odstające

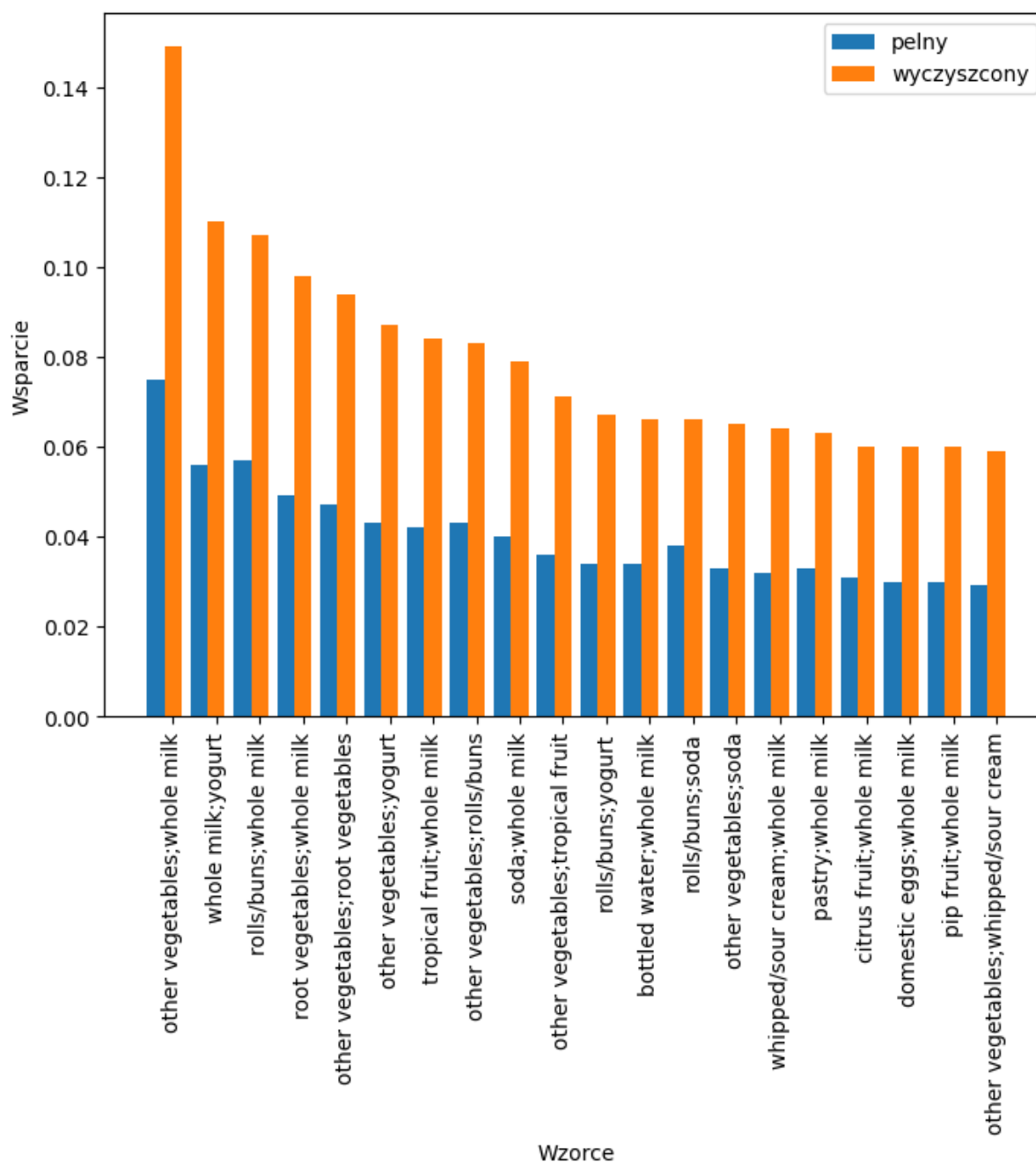
Zbadano wpływ koszyków odstających na średni poziom wyznaczanego Wsparcia wzorców. W tym celu użyty został zbiór pierwszy, z uwagi na liczną obecność w nim koszyków jednoelementowych. Rysunek 23 przedstawia analizę wartości odstających metodą IQR, czyli wyznaczeniem różnicy między trzecim a pierwszym kwantylem danych[34]. Koszykami odstającymi okazały się koszyki posiadające mniej niż 4 przedmioty, co stanowi aż 52% całości zbioru. Jest to znaczący ubytek danych mogący wpłynąć na otrzymane wyniki rekomendacji.



Rysunek 23: Analiza wartości odstających zbioru pierwszego

Na podstawie zaprezentowanego wykresu stworzono podzbiór, z którego usunięto wyznaczone koszyki odstające i porównano go ze zbiorem oryginalnym. Wynik tego porównania przedstawiony został na Rysunku 24. Przedstawia on porównanie Wsparcia dwudziestu najpopularniejszych wzorców dwuelementowych dla zbioru pierwszego oraz jego wyczyszczonego podzbioru. Zaobserwować można znaczącą przewagę w wartości wyznaczanego Wsparcia dla podzbioru wyczyszczonego. Wraz z rosnącym rozmiarem koszyka, zwiększa się też liczba wzorców, które mogą powstać na jego podstawie. Usunięte, małe koszyki posiadały niewielką liczbę wzorców, które mogły być do nich dopasowane, w rezultacie obniżając średnie Wsparcie dla całego zbioru. Należy również zwrócić uwagę na różnicę w kolejności wyznaczonych wzorców. Może mieć ona znaczenie dla finalnej rekomendacji, zmieniając kolejność przedmiotów otrzymaną po wyznaczeniu Siły, co może skutkować innym ułożeniem produktów. Dodatkowo zwiększone średnie Wsparcie zbioru zwiększa również liczbę wzorców spełniających próg minimalnego Wsparcia, co przekłada się na większą liczbę reguł, poprawiając tym samym dokładność rekomendacji.





Rysunek 24: Porównanie Wsparcia zbiorów

#### 4.2.2 Poziom minimalnego Wsparcia

W kolejnym etapie zbadano wpływ poziomu minimalnego Wsparcia na liczbę wyznaczanych wzorców. Dla każdej kolejnej próby zmniejszano wartość badanego progu. Badanie zostało przeprowadzone na oczyszczonym zbiorze pierwszym. Wyniki zostały zaprezentowane w Tabeli 3.

Należy zwrócić uwagę na kolumnę dotyczącą wzorców jednoelementowych. Konstrukcja algorytmu rekomendacji sprawia, iż rekomendowane są tylko te produkty, które zostaną zarejestrowane w ramach dowolnego wzorca. Z tego powodu algorytm ignoruje produkty skrajnie niepopularne. Obniżając próg minimalnego Wsparcia użytkownik zwiększa liczbę produktów dostępnych podczas finalnej rekomendacji

Wsparcie(%)	jeden	dwa	trzy	cztery	pięć	sześć	siedem
10	20	3	0	0	0	0	0
5	43	34	0	0	0	0	0
2.5	72	142	17	0	0	0	0
1	108	609	293	15	0	0	0
0.75	121	891	562	49	0	0	0
0.5	135	1356	1373	221	7	0	0
0.25	151	2511	4935	1922	168	3	0
0.1	163	4684	19160	18594	5975	769	28

Tabela 3: Liczba wyznaczonych wzorców dla oczyszczonego zbioru pierwszego

Odnotowano gwałtowny wzrost liczby wyznaczanych wzorców, widoczny szczególnie dla ostatniej przeprowadzonej próby oraz długości wzorców wynoszących conajmniej 4. Wzrost ten mógł być spowodowany kształtem rozkładu danych, który mimo usunięcia wartości odstających, nadal był rozkładem funkcji homograficznej. Postanowiono więc powtórzyć badanie dla zbioru drugiego, z zachowaniem ustalonych wcześniej progów. Wyniki badania zostały przedstawione w Tabeli 4

Wsparcie(%)	jeden	dwa	trzy	cztery	pięć	sześć	siedem	osiem
10	29	13	0	0	0	0	0	0
5	37	97	11	0	0	0	0	0
2.5	78	346	143	4	0	0	0	0
1	116	1126	1459	311	4	0	0	0
0.75	126	1476	2441	753	29	0	0	0
0.5	141	2099	4975	2432	228	1	0	0
0.25	154	3464	13942	13428	3437	195	7	0
0.1	164	5872	41941	87756	17043	1740	28	49

Tabela 4: Liczba wyznaczonych wzorców dla zbioru drugiego

Również i w tym przypadku odnotowano gwałtowny wzrost liczby wyznaczanych wzorców dla zmniejszającego się progu Wsparcia. Należy także zwrócić uwagę na łączną

liczbę wyznaczonych wzorców w ramach omawianego obecnie badania. Jest ona znacząco większa w porównaniu z badaniem dla zbioru pierwszego. Różnica ta może mieć związek z średnią liczbą produktów znajdujących się w pojedynczym koszyku. Postanowiono więc, z pomocą zbiorów losowych, dokładniej zbadać to założenie.

### 4.2.3 Średnia liczba produktów w koszyku

W Tabeli 5 przedstawiono specyfikację koszyków, wygenerowanych losowo za pomocą języka Python, na potrzeby przeprowadzanego badania.

Nazwa	Liczba koszyków	Liczba produktów	Średnia produktów na koszyk
losowy1	5000	50	7.22
losowy2	2000	50	7.49
losowy3	2000	100	7.53
losowy4	2500	50	4.95
losowy5	2000	50	11.58
losowy6	2000	100	11.34
losowy7	2500	100	8.49

Tabela 5: Opis wygenerowanych zbiorów

Średnia liczba produktów dla zbiorów losowych została ustalona na podstawie średniej liczby produktów zbiorów Kaggle. Liczba koszyków oraz produktów została dopasowana do użytego rozkładu danych. Działania te miały na celu przybliżenie wyglądu danych losowych do danych oryginalnych.

Do badania wykorzystano dodatkowo zbiór pierwszy o średniej 4.41, zbiór pierwszy oczyszczony o średniej 7.18, oraz zbiór drugi o średniej 9.94. Analiza rozkładu zbioru drugiego nie wykazała obecności koszyków odstających. Ustalono próg minimalnego Wsparcia dla próby na poziomie 0.2%. Próg ten został on dobrany na podstawie wyników badań opisanych w podrozdziale powyżej i pozwala on na wyznaczenie reprezentatywnej grupy wzorców dla każdego zbioru.

Wynik badania został przedstawiony na Rysunku 25. Pominęto na nim wynik zbioru losowego 5, gdyż wyniósł on 535388, co zaburzyło by wygląd całego wykresu. Wynik przeprowadzonego eksperymentu wskazuje na fakt, iż średnia liczba przedmiotów na koszyk nie jest głównym czynnikiem, od którego zależy liczba wyznaczanych wzorców. Zwiększanie liczby koszyków lub liczby produktów, przy jednoczesnym utrzymywaniu średniej liczby przedmiotów w koszyku na stałym poziomie obniżało liczbę znalezionych wzorców.



Rysunek 25: Porównanie liczby wyznaczonych wzorców dla poszczególnych zbiorów

Zwiększenie liczby koszyków sprawia, że konkretny produkt musi się pojawić więcej razy w transakcjach celem zachowania wcześniejszego poziomu Wsparcia. Zwiększenie liczby produktów zaś generuje konkurencję między przedmiotami, co przy zachowaniu stałej średniej liczby produktów na koszyk w zbiorze, zmniejsza liczebność poszczególnych przedmiotów w transakcjach.

Dodatkowo pozytywny wpływ na liczebność znajdujących wzorców miała obecność dużych koszyków w zbiorze, co widać podczas porównania nieoczyszczonego zbioru pierwszego z losowym zbiorem czwartym. Dla zbioru losowego największym rozmiarem koszyka jest 12, gdzie dla zbioru pierwszego jest to już 32. Jest to znacząca rozbieżność, widoczna w liczbie znalezionych wzorców.

Przeprowadzane badania pozwalają na określenie optymalnego poziomu minimalnego Wsparcia dla zbioru. Na podstawie danych zawartych w rozdziale 4.2.2 poziom ten powinien zostać ustalony w przedziale 0.1-0.25%. Jego dokładna wartość powinna być dostosowywana na podstawie rozkładu danych wejściowych, pozwalając na bardziej rygorystyczne podejście podczas ustalania progu Wsparcia dla danych, posiadających wyższą średnią liczbę produktów na koszyk.

#### 4.2.4 Wydajność algorytmu

Podczas badania związanego z poziomem minimalnego Wsparcia natknięto się na problem związany z czasem wykonywania zaimplementowanego algorytmu Apriori. Posta-

nowiono więc przyjrzeć się jego wydajności. Okazało się, że głównym problemem związanym z rozważanym algorytmem była nadprogramowa liczba generowanych kandydatów, którzy nie mogli zostać wzorcami częstymi, doprowadzając tym samym do sporej ilości nadprogramowych porównań spowalniających ogólny proces generowania kandydatów. Wprowadzono więc szereg usprawnień mających na celu poprawę wydajności Apriori.

Pierwszym wprowadzonym usprawnieniem było usuwanie koszyków, których rozmiar był mniejszy od  $K$ , czyli od rozmiaru obecnie rozpatrywanych kandydatów. Zniwelowało to operacje, podczas których sprawdzana była obecność kandydata dla koszyka mniejszego niż on sam.

K	Przed zmianami	Po zmianach	Znalezione wzorce
2	11026	11026	3464
3	349504	196458	13942
4	4421275	532919	13428
5	17259390	269347	3437
6	3838380	19225	195
Czas(s):	2241	136	—

Tabela 6: Porównanie liczby kandydatów

Po drugie, wprowadzono dodatkowy próg używany podczas klasyfikacji znalezionych wzorców częstych jako bazę  $K+1$  kandydatów. Skorzystano z faktu, iż wszystkie podwzorce wzorca częstego również są wzorcami częstymi. Dlatego, jeżeli wzorec kandydujący został uznany za wzorec częsty, to aby zostać dopuszczonym do bazy  $K+1$  kandydatów, jego Wsparcie musiało być większe od ustalonego progu minimalnego Wsparcia o współczynnik wynoszący 1.5. Współczynnik ten został dobrany na bazie eksperymentu. Dla każdej kolejnej próby zwiększano współczynnik o 0.1 zaczynając od wartości 1.1. Dla współczynnika na poziomie 1.6 zaczęto odnotowywać spadek wyznaczonej liczby wzorców. Dlatego wartość współczynnika została ustalona na poziomie 1.5. Działanie to redukuje zbiór kandydatów o wzorce, które zostały sklasyfikowane na granicy progu, z bardzo małą szansą na stworzenie wzorca częstego o długości  $K+1$ . Czyli, jeżeli próg minimalnego Wsparcia został ustalony na poziomie 1%, to minimalny próg Wsparcia dla kolejnych kandydatów musiał wynieść odpowiednio 1.5%.

Zmianie uległ również sam proces generowania kandydatów. Krok algorytmu dotyczący wyznaczania unikalnej listy przedmiotów, które weszły w skład poprzedniej epoki

został niezmienny. Całkowicie zaś zrezygnowano z rekurencji. Napisano nową funkcję, pobierającą wyznaczoną unikalną listę przedmiotów, oraz listę kandydatów, którzy zostali dopuszczeni po uwzględnieniu nowego progu. Listy te były łączone w nowy zbiór kandydatów  $K+1$  elementowych poprzez dodawanie do obecnego zbioru kandydatów pojedynczych przedmiotów z unikalnej listy.

Tabela 6 pokazuje skuteczność i czas działania algorytmu Apriori przed wprowadzeniem zmian oraz po ich zaimplementowaniu. Dotyczy ona próby przeprowadzonej na zbiorze drugim dla poziomu Wsparcia wynoszącego 0.25%, podczas której zauważono oraz zdiagnozowano problem. Zauważyć można znaczącą redukcję liczby generowanych kandydatów, co skutecznie obniża liczbę porównań wykonywanych w ramach wyliczania Wsparcia, co jest widoczne w czasie wykonania usprawnionej wersji algorytmu.

## 4.3 Zachowania reguł

### 4.3.1 Ufność

Postanowiono zbadać rozkład przedziałów Ufności na dotychczas wygenerowanych zbiorach wzorców. Zrezygnowano ze zbiorów losowych, z uwagi na silne powiązanie Ufności z bezpośrednią zawartością wzorców. Takie dane, z powodu tego że są losowe, mogłyby zaburzyć otrzymane wnioski. Dla oczyszczonego zbioru pierwszego oraz zbioru drugiego opisanych w rozdziale 4.2.2 zbadano zmiany w rozkładzie Ufności względem zmieniającego się poziomu minimalnego Wsparcia oraz sprawdzono wpływ pierwotnego rozkładu danych koszykowych na otrzymywaną Ufność. Dźwignia reguł została ustalona na poziomie 0, celem wygenerowania wszystkich możliwych reguł. Wyniki badania zostały przedstawione w Tabelach 7 oraz 8.

Ufność(%) \ Wsparcie(%)	>40	30-40	20-30	10-20	0-10
1	10.83	8.73	14.85	31.04	34.55
0.75	10.58	7.72	12.93	27.81	40.96
0.5	10.04	6.75	11.32	24.65	47.22
0.25	11.12	6.48	10.44	19.52	53.32
0.1	9.53	5.61	9.81	15.07	59.96

Tabela 7: Rozkład Ufności dla zbioru pierwszego

Ufność(%) \ Wsparcie(%)	>40	30-40	20-30	10-20	0-10
1	14.38	7.61	15.44	25.19	37.38
0.75	12.88	7.26	14.02	23.96	41.58
0.5	11.17	6.56	12.52	21.76	47.99
0.25	10.55	5.64	10.19	18.85	54.77
0.1	9.13	5.94	6.29	15.93	62.71

Tabela 8: Rozkład Ufności dla zbioru drugiego

Wyniki, otrzymane na bazie użytych zbiorów Kaggle, nie różnią się znacząco względem siebie. Wraz ze zmniejszaniem poziomu minimalnego Wsparcia zbioru wzorców, malała również różnica w rozkładzie Ufności między zbiorami. Z tego powodu nie można jednoznacznie potwierdzić wpływu pierwotnego rozkładu danych koszykowych na otrzymywane wartości Ufności reguł.

Zauważyć można natomiast wzrost udziału najmniejszego przedziału Ufności wraz ze zmniejszaniem poziomu minimalnego Wsparcia użytego do wyznaczenia wzorców. Zmniejszając próg Wsparcia, zwiększa się szansa wygenerowania krótkiej reguły łączącej najpopularniejsze produkty z tymi najrzadszymi. Reguły powstałe na podstawie takich wzorców, będą miały niską Ufność z powodu różnicy w częstotliwości zakupu poszczególnych produktów wewnątrz niej. Nie zostanie zaś to uchwycone przy wyliczaniu Dźwigni z uwagi na niskie Wsparcie zarówno całości rozważanego wzorca jak i rzadkiego produktu. Przepuszczenie tego typu danych ma ogromne znaczenie dla całej rekomendacji, wypaczając finalny wynik.

Ufność jest podstawową metryką służącą do oceny jakości wzorca. Wartość tej metryki bliska 0 informuje, iż produkty znajdujące się wewnątrz rozważanego wzorca nie mają ze sobą nic wspólnego. Dlatego jeżeli takie paczki zostaną dopuszczone do inicjacji reguł, to wyznaczona na tej podstawie Siła nie będzie miała pokrycia w rzeczywistości. Dlatego tak ważnym jest dobór odpowiedniego współczynnika Ufności, celem odsiania skrajnych przypadków.

Na podstawie sformułowanych powyżej wniosków ustalono optymalny poziom Ufności reguł w przedziale 10-20% celem pogodzenia liczby wyznaczanych reguł z ich przekrojową jakością.

### 4.3.2 Dźwignia

Zbadano rozkład przedziałów Dźwigni w wyznaczanych zbiorach reguł. Skorzystano z wyznaczonych już danych na rzecz poprzedniego podrozdziału. Na podstawie wyżej wyciągniętych wniosków, ustalono próg Ufności reguł na poziomie 10%, celem odsiania najmniej jakościowych wzorców. Wyniki badania zostały przedstawione w Tabelach 9 oraz 10. W przeciwieństwie do rozkładu Ufności, dla rozkładu Dźwigni można wskazać realny wpływ kształtu danych koszykowych na wyliczaną metrykę. Zbiór pierwszy o rozkładzie wykresu funkcji homograficznej posiada zdecydowanie większą liczbę reguł o Dźwigni mniejszej niż jeden. Wartość ta informuje o tym, że powiązane w ten sposób wzorce zazwyczaj nie są kupowane razem i tego typu reguły zaburzałyby realny wynik rekomendacji.

Wraz ze zmniejszaniem minimalnego Wsparcia wzorców w zbiorze, znacznie zwiększa się wyznaczana wartość Dźwigni reguł. Jeżeli dany wzorzec posiada niskie Wsparcie i jest on wzorcem stosunkowo długim, to posiadając w koszyku jego podwzorzec gwałtownie zwiększa się szansa na zakup reszty wzorca, aby ten mógł zaistnieć.

Dźwignia Wsparcie(%)	<1	1-1.5	1.5-2	2-5	>5
1	15.77	60.95	18.56	4.70	0
0.75	14.50	56.64	22.07	6.78	0
0.5	11.97	50.28	25.63	11.97	0.13
0.25	7.12	31.28	26.44	33.09	2.05
0.1	5.00	16.22	15.91	43.27	19.57

Tabela 9: Rozkład Dźwigni dla zbioru pierwszego

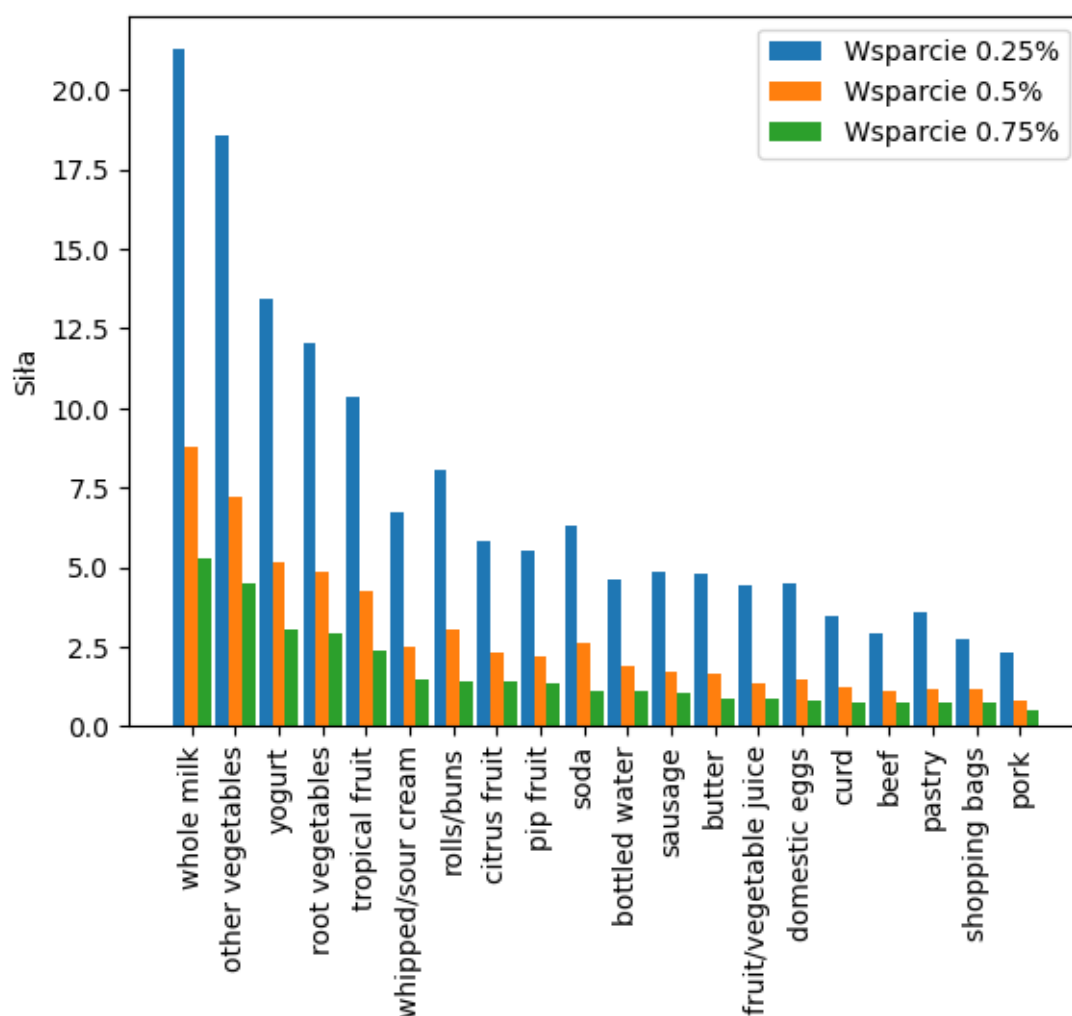
Dźwignia Wsparcie(%)	<1	1-1.5	1.5-2	2-5	>5
1	1.46	81.36	16.40	0.76	0
0.75	1.79	72.95	23.27	1.98	0
0.5	2.37	61.20	30.32	6.10	0
0.25	2.21	34.03	33.24	29.84	0.66
0.1	2.63	10.43	20.91	46.87	19.1

Tabela 10: Rozkład Dźwigni dla zbioru drugiego



Na podstawie przeprowadzonych badań nie można jednoznacznie wskazać jednego, optymalnego poziomu metryki Dźwigni dla zbioru reguł. Poziom ten powinien być zależny od minimalnej wartości Wsparcia. Zaleca się jednak, pamiętając o tym, że wartości Dźwigni bliskie 1 informują o braku korelacji między produktami, aby wartość Dźwigni nie była mniejsza niż 1.2 celem poprawnego tworzenia powiązań między wzorcami, które mają realny wpływ na siebie nawzajem.

### 4.3.3 Siła



Rysunek 26: Rozkład Siły najpopularniejszych produktów

Przeprowadzono zbiorcze badanie mające na celu sprawdzenie wpływu minimalnego Wsparcia wzorców oraz ustalonego progu Dźwigni na wyznaczaną wartość Siły produktów, co za tym idzie na kolejność produktów w ramach rekomendacji. Ustalono Ufność reguł we wszystkich próbach na poziomie 10%. Pominęto próbę dla Wsparcia wynoszącego 1%

z powodu niewystarczającej liczby reguł, oraz próbę dla Wsparcia wynoszącego 0.1% z uwagi na limit pamięci wykorzystywanej w ramach tejże pracy darmowej instancji Neo4j. Rysunek 26 przedstawia wyniki dotyczące badania wpływu Wsparcia na Siłę. Z uwagi na rozmiary badania postanowiło przedstawić reprezentatywny wycinek danych dotyczący zbioru pierwszego oraz prób dla Dźwigni wynoszącej 1.5.

Wsparcie% Dźwignia	whole milk	yogurt	rolls/buns	citrus fruit	chicken
0.75 1.2	6.088	3.374	2.321	1.419	0.380
0.75 1.5	5.267	3.019	1.412	1.410	0.415
0.75 1.8	4.300	2.149	0.815	1.041	0.306
0.5 1.2	9.135	5.17	3.749	2.250	0.698
0.5 1.5	8.763	5.159	3.026	2.290	0.693
0.5 1.8	7.750	4.152	2.001	2.062	0.607
0.25 1.2	20.693	12.880	8.002	5.505	1.807
0.25 1.5	21.268	13.458	8.067	5.824	1.917
0.25 1.8	22.204	13.699	8.064	6.169	1.962

Tabela 11: Zmiana Siły w stosunku do poziomu Dźwigni dla zbioru pierwszego

Wsparcie% Dźwignia	whole milk	yogurt	rolls/buns	citrus fruit	chicken
0.75 1.2	17.970	10.382	13.132	3.694	0.976
0.75 1.5	15.801	8.916	11.286	2.511	0.504
0.75 1.8	10.745	6.109	7.558	1.374	0.418
0.5 1.2	27.880	16.657	20.557	6.194	1.832
0.5 1.5	27.331	15.765	18.442	5.530	1.682
0.5 1.8	21.983	12.491	15.161	3.989	1.329
0.25 1.2	71.219	43.065	52.066	17.420	5.666
0.25 1.5	75.632	45.651	54.540	18.234	6.054
0.25 1.8	76.813	44.343	53.506	17.686	5.736

Tabela 12: Zmiana Siły w stosunku do poziomu Dźwigni dla zbioru drugiego

Zauważyć można wzrost wyznaczanej Siły dla zmniejszającego się progu Wsparcia. Wraz ze zwiększaniem dopuszczonej liczby wzorców do procesu generowania reguł, zwiększa się też potencjalna liczba operacji, dzięki którym konkretny produkt może zyskać na sile. Należy również zwrócić uwagę na zmianę kolejności najpopularniejszych produktów dla kolejnych progów Wsparcia. Dopuszczenie coraz to dłuższych oraz rzadszych wzorców zwiększa dokładność rekomendacji.

Tabele 11 oraz 12 pokazują realny wpływ wybranej wartości Dźwigni na Siłę produktów. Produkty zostały wybrane na podstawie ich popularności, celem sprawdzenia zachowania

wań siły na jej różne przedziały. Zauważyć można punkt krytyczny dla wartości metryki Dźwigni. Dążenie do tego punktu zwiększa wyznaczaną Siłę produktów. Po jego przekroczeniu zaś wyznaczana Siła ulega obniżeniu. Graniczna wartość Dźwigni jest zależna od poziomu Wsparcia oraz jest skorelowana z rozkładem przedziałów Dźwigni pokazanym w rozdziale 4.3.2.

Opisane zachowanie ma ścisły związek z budową algorytmu rekomendacji. Siła produktów jest dodawana na podstawie średniej wartości Dźwigni każdego inicjującego regułę wzorca. Przed przekroczeniem punktu krytycznego, zwiększając wartość Dźwigni zwiększa się również średnia wartość metryki dla inicjującego regułę wzorca, co za tym idzie zwiększa się jednorazowa wartość dodawanej Siły dla produktów. Po przekroczeniu progu, wartość Dźwigni staje się na tyle duża, że zmniejsza się liczba wzorców, które inicjują reguły, przez co zmniejsza się ilość danych mogących zwiększyć wartość Siły produktów.

## 4.4 Układ sklepu

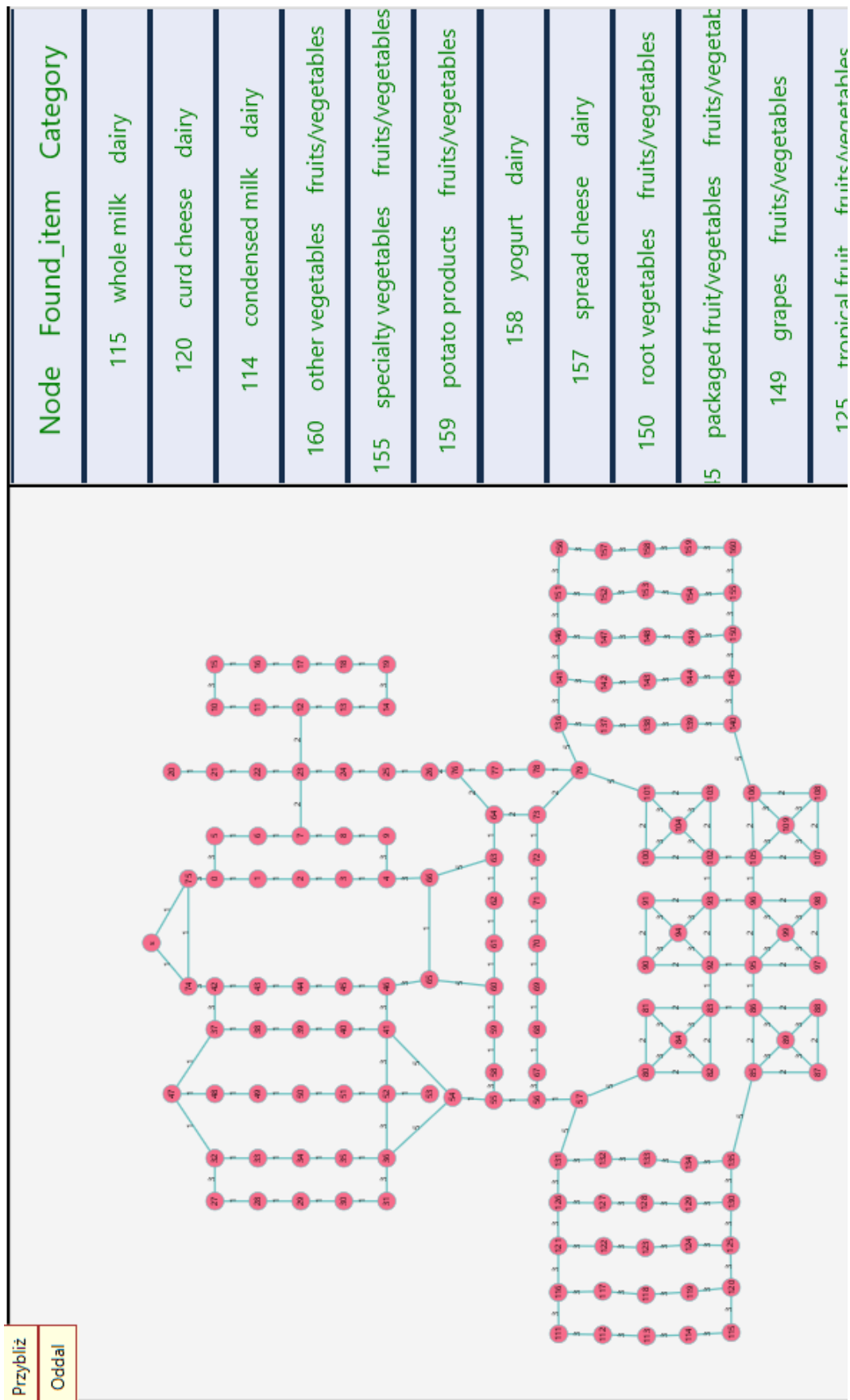
Układ sklepu jest nie mniej ważny od wyboru odpowiednich wartości wyżej badanych metryk. Dla zbiorów omawianych na rzecz tego rozdziału, tworzony graf powinien zawierać cały układ sklepu. W takim przypadku pojedynczy wierzchołek grafu nie powinien odnosić się do konkretnego miejsca na półce. Powinien on reprezentować umowny segment, będący na przykład konkretną częścią regału znajdującego się w alejce sklepowej. Ich zagęszczenie powinno być zależne od wielkości sklepu oraz liczby produktów. Graf sklepowy powinien być tworzony po poznaniu listy produktów wynikającej z poziomu Siły. Pozwala to dopasować liczbę wierzchołków do liczby rekomendowanych produktów. Za mała liczba wierzchołków sprawi, że pominięte zostaną produkty znajdujące się mniej więcej w połowie listy popularności. Zbyt duża, że najpopularniejsze produkty zostaną sklonowe i znajdą się blisko wejścia do sklepu.

Kolejnym ważnym czynnikiem jest sposób deklaratowania ścieżek w grafie. Algorytm ustawiając popularny produkt w konkretnym wierzchołku przypisuje wszystkim jego sąsiadom niepopularny produkt tej samej kategorii. Poprzez umiejętne tworzenie grafu, znając liczebność konkretnych kategorii produktów, można wymusić powstanie skupisk gromadzących przedmioty danych kategorii blisko siebie.

Dodatkowo ważnym czynnikiem jest wybór punktu startowego w grafie. Wygenerowanie symetrycznego grafu wraz z wierzchołkiem startowym na linii symetrii, sprawia że

wierzchołki najbardziej oddalone od startu są jednocześnie stosunkowo mocno oddalone od siebie nawzajem. Pozwala to spełnić założenie postawione na początku pracy, mówiące o maksymalizacji powierzchni, po której musi poruszać się klient celem skompletowania jego listy zakupowej.

Rysunek 27 przedstawia przykładowy układ sklepu. Wykorzystano wyczyszczony zbiór pierwszy, Wsparcia=0.25%, Ufności=10% oraz Dźwigni=1.2. Graf sklepowy został w całości zbudowany z dostępnych w interfejsie szablonów. Punkt startowy znalazł się w wierzchołku S. Najpopularniejsze produkty, czyli *whole milk* oraz *other vegetables* znalazły się po przeciwległy stronach sklepu odpowiednio w wierzchołkach 115 i 160, co było efektem zamierzonym



Rysunek 27: Rekomendacja



## Podsumowanie

Tematem niniejszej pracy dyplomowej była *Analiza danych zakupowych i budowa systemu rekomendacji ułożenia produktów sklepowych z wykorzystaniem bazy grafowej*. Udało się spełnić wszystkie wyznaczone założenia pracy. Przybliżono problem manipulacji klientem w sklepie. Opisano szczegółowo budowę, zasady działania oraz implementację algorytmu rekomendacji. Porównano zaproponowaną implementację algorytmów Apriori i Dijkstry do istniejących rozwiązań. Przybliżono strukturę aplikacji stworzonej na potrzeby obsługi algorytmu. Opisano budowę interfejsu grafowego wraz z zawartymi w nim szablonami układu sklepów.

Przeprowadzono szczegółowe badania, na podstawie zbiorów danych dostępnych w bazie Kaggle, mające na celu zbadanie zachowań napisanego algorytmu. Pochylono się nad tematem metryk służących do opisu jakości reguł asocjacyjnych i zbadano zależności między nimi. Opisano m.in. wpływ metryki Wsparcia na wyznaczaną liczbę wzorców, znaczenie doboru odpowiedniej wartości metryki Ufności na jakość rekomendacji, czy wpływ metryki Dźwigni na Siłę produktów. Ustalono optymalne poziomy metryk na poziomach: Wsparcie=0.2%, Ufność=10% oraz Dźwignia=1.2. Określono wpływ skrajnych długości koszyków na liczbę otrzymywanych wzorców jako pozytywny dla koszyków długich oraz negatywny dla koszyków krótkich. Poruszono również temat układu sklepu i jego połączeniu z rekomendowaną listą produktów.





## Literatura

- [1] Chapman, A. (1980). Barter as a universal mode of exchange. *L’homme*, 33-83.
- [2] Leninkumar, V. (2017). The relationship between customer satisfaction and customer trust on customer loyalty. *International Journal of Academic Research in Business and Social Sciences*, 7(4), 450-465.
- [3] Becher, S. I., Feldman, Y. (2016). Manipulating, fast and slow: The law of non-verbal market manipulations. *Cardozo L. Rev.*, 38, 476-486.
- [4] Biesaga-Słomczewska-dr, E. J.(2010) Zachowania klientów hipermarketów a techniki manipulacji. *Ekonomiczne Problemy Usług* nr 54, 57-66
- [5] <https://www.leafio.ai/pl/> dostęp: (07.02.2025)
- [6] <https://Wsparcie-sprzedazy.assecobs.pl/produkty/mobile-touch/> dostęp: (07.02.2025)
- [7] <https://www.leafio.ai/pl/blog/> dostęp: (07.02.2025)
- [8] <https://docs.oracle.com/javase/8/docs/api/> dostęp: (21.02.2025)
- [9] <https://docs.oracle.com/javafx/2/api/> dostęp: (21.02.2025)
- [10] <https://developer.mozilla.org/en-US/docs/Web/CSS> dostęp: (21.02.2025)
- [11] <https://neo4j.com/> dostęp: (24.02.2025)
- [12] Zhang, P., Chartrand, G. (2006). Introduction to graph theory (Vol. 2, No. 2.1). New York: Tata McGraw-Hill.
- [13] Fan, W., Ma, Y., Li, Q., He, Y., Zhao, E., Tang, J., Yin, D. (2019, May). Graph neural networks for social recommendation. In *The world wide web conference* (pp. 417-426).
- [14] Darban, Z. Z., Valipour, M. H. (2022). GHRS: Graph-based hybrid recommendation system with application to movie recommendation. *Expert Systems with Applications*, 200, 116850.
- [15] <https://neo4j.com/docs/cypher-manual/current/clauses/> dostęp: (24.02.2025)

- [16] A. Falkowski, T. Tyszka (2006). Psychologia zachowań konsumenckich. Gdańskie Wydawnictwo Psychologiczne (GWP) 203.
- [17] [https://pandas.pydata.org/docs/user\\_guide/index.html#user-guide](https://pandas.pydata.org/docs/user_guide/index.html#user-guide)  
dostęp: (20.03.2025)
- [18] <https://numpy.org/doc/stable/user/basics.html> dostęp: (20.03.2025)
- [19] [https://matplotlib.org/stable/users/explain/quick\\_start.html](https://matplotlib.org/stable/users/explain/quick_start.html) dostęp: (20.03.2025)
- [20] Efrat, A. R., Gernowo, R. (2020, April). Consumer purchase patterns based on market basket analysis using apriori algorithms. In Journal of Physics: Conference Series (Vol. 1524, No. 1, p. 012109). IOP Publishing.
- [21] Al-Maolegi, M., Arkok, B. (2014). An improved Apriori algorithm for association rules. arXiv preprint arXiv:1403.3948.
- [22] Yuan, X. (2017, March). An improved Apriori algorithm for mining association rules. In AIP conference proceedings (Vol. 1820, No. 1). AIP Publishing.
- [23] Karthiyayini, R., Balasubramanian, R. (2016). Affinity analysis and association rule mining using apriori algorithm in market basket analysis. International Journal, 6(10).
- [24] Alawadh, M. M., Barnawi, A. M. (2022). A survey on methods and applications of intelligent market basket analysis based on association rule. Journal on Big Data, 4(1).
- [25] E. W. Dijkstra (1959). A Note on Two Problems in Connexion with Graphs. Numerische Mathematik 1, 269-271.
- [26] Fredman, M. L., Tarjan, R. E. (1987). Fibonacci heaps and their uses in improved network optimization algorithms. Journal of the ACM (JACM), 34(3), 596-615.
- [27] Driscoll, J. R., Gabow, H. N., Shairman, R., Tarjan, R. E. (1988). Relaxed heaps: An alternative to Fibonacci heaps with applications to parallel computation. Communications of the ACM, 31(11), 1343-1354.

- [28] Chen, M., Chowdhury, R. A., Ramachandran, V., Roche, D. L., Tong, L. (2007). Priority queues and Dijkstra's algorithm.
- [29] <https://neo4j.com/product/auradb/> (dostęp 27.02.2025)
- [30] <https://www.kaggle.com/datasets/irfanasrullah/groceries> (dostęp 18.03.2025)
- [31] <https://www.kaggle.com/datasets/heeraldedhia/groceries-dataset> (dostęp 01.04.2025)
- [32] Kenneth A. Ross, Charles R. B. Wright 2001, Matematyka dyskretna. Wydawnictwo Naukowe PWN, s. 276
- [33] <https://neo4j.com/cloud/platform/aura-graph-database/faq/> (dostęp 08.04.2025)
- [34] Rousseeuw, P. J., Hubert, M. (2011). Robust statistics for outlier detection. Wiley interdisciplinary reviews: Data mining and knowledge discovery, 1(1), 73-79.