Dokumentacja Projektu

Zespół projektowy nr. 5

27 maja 2025

Spis treści

1 Skład zespołu i role poszczególnych osób

Imię i nazwisko	Rola w projekcie	Zakres obowiązków
Kacper Pawlicki	Koordynator projektu i Programista Frontend	Planowanie pracy zespołu, podział zadań, implementacja interfejsu użytkownika, integracja z backendem
Paweł Witkowski	Programista backend	Implementacja algorytmów oraz struktur danych
Marcin Adamczyk	Programista backend	Implementacja algorytmów oraz struktur danych
Łukasz Szmołda	Dokumentacja, Testy	Przygotowanie dokumentacji technicznej, testowanie funkcjonalności, zgłaszanie błędów

2 Tytuł projektu

BeerLand – aplikacja do rozwiązania problemu projektowego

Celem projektu jest stworzenie aplikacji, która w sposób funkcjonalny i nowoczesny rozwiązuje postawiony problem, głównie przy użyciu odpowiednich i dobrze zaimplementowanych algorytmów. Aplikacja została zaprojektowana na potrzeby realizacji zadania akademickiego i ma na celu symulację rzeczywistego procesu tworzenia oprogramowania – od analizy wymagań, przez projektowanie, aż po implementację i testowanie.

3 Specyfikacja problemu

Celem projektu jest stworzenie rozwiązania informatycznego, które umożliwi efektywne zarządzanie transportem jęczmienia oraz produkcją i dystrybucją piwa w fikcyjnym kraju Shire. Problem został przedstawiony jako symulacja rzeczywistego zagadnienia logistycznego, w którym konieczne jest zapewnienie maksymalnej ilości piwa dostarczanej do lokalnych karczm przy uwzględnieniu ograniczeń związanych z infrastrukturą transportową oraz wydajnością pól i browarów.

Dane wejściowe

- Położenie i wydajność pól jęczmienia (ilość zboża dostępna na każdym poletku).
- Lokalizacje browarów wraz z ich maksymalną przepustowością produkcyjną.
- Lokalizacje karczm będących punktami docelowymi dla transportu piwa.
- Sieć dróg reprezentująca możliwości transportu zboża i piwa pomiędzy skrzyżowaniami (wraz z informacją o przepustowości i ewentualnym koszcie naprawy).

Opis problemu

Problem można sformalizować jako przepływ w sieci, w której:

- Pola są źródłami surowca (jęczmień),

- Browary są przetwórniami o ograniczonej przepustowości (produkują piwo z jęczmienia),
- Karczmy są punktami końcowymi (odbiorcami piwa),
- Drogi reprezentują możliwe przepływy (z ograniczoną przepustowością i opcjonalnym kosztem naprawy).

Zadania

1. Wyznaczenie maksymalnego przepływu piwa z pól do karczm przy zachowaniu wszystkich ograniczeń.
2. Minimalizacja kosztu naprawy dróg przy zachowaniu maksymalnego przepływu (problem minimum-cost maximum-flow).
3. Uwzględnienie regionalnych różnic w wydajności upraw – każda ćwiartka Shire ma inną efektywność.
4. Implementacja efektywnego mechanizmu wyszukiwania słów kluczowych (np. *„piwo”, „jęczmień”, „browar”*).

Wymagania wobec rozwiązania

Ostateczne rozwiązanie powinno uwzględniać:

- Dobór odpowiednich struktur danych.
- Zastosowanie klasycznych algorytmów grafowych.
- Teoretyczny opis rozwiązania wraz z analizą złożoności obliczeniowej i pamięciowej.
- Implementację w wybranym języku programowania.
- Testy działania aplikacji na różnych zestawach danych (również losowych i skrajnych przypadkach).

Projekt pełni rolę ćwiczenia inżynierskiego, łącząc elementy algorytmiki, modelowania danych, optymalizacji oraz przetwarzania tekstu, umożliwiając zespołowi studentów kompleksową realizację procesu tworzenia rozwiązania programistycznego.

4 Podział problemu na podproblemy

Projekt BeerLand został podzielony na szereg mniejszych zadań (podproblemów), które razem umożliwiają kompleksowe rozwiązanie postawionego problemu. Każdy z podproblemów wymaga odpowiednich kompetencji w zakresie algorytmiki, programowania oraz analizy danych.

1. **Modelowanie danych wejściowych**
    - Reprezentacja pól, browarów, karczm i dróg jako elementów grafu.
    - Wczytywanie i walidacja danych wejściowych.

2. **Wyznaczenie maksymalnego przepływu z pól do karczm**
    - Implementacja algorytmu maksymalnego przepływu.
    - Uwzględnienie etapów: pole → browar → karczma.
3. **Minimalizacja kosztu naprawy dróg przy zachowaniu maksymalnego przepływu**
    - Zastosowanie algorytmu min-cost max-flow.
    - Dodanie kosztów do krawędzi grafu.
4. **Uwzględnienie regionalnych różnic w wydajności upraw**
    - Przypisanie pól do odpowiednich ćwiartek mapy.
    - Przeliczanie produkcji jęczmienia na podstawie regionu (wielokąty wypukłe).
5. **Wyszukiwanie tekstowe w dokumentacji**
    - Implementacja struktur danych do wyszukiwania słów.
    - Porównanie efektywności metod (czas i pamięć).
6. **Frontend aplikacji**
    - ???
    - ???
7. **Testowanie i walidacja rozwiązania**
    - Tworzenie danych testowych, w tym przypadków brzegowych.
    - Pomiar czasu działania i zużycia pamięci.
8. **Dokumentacja i raport końcowy**
    - Opis teoretyczny algorytmów oraz analiza złożoności.
    - Instrukcja obsługi aplikacji oraz omówienie wyników.

5 Harmonogram realizacji projektu

Etap	**Data rozpoczęcia**	**Data zakończenia**
Reprezentacja i modelowanie danych wejściowych	07 kwietnia	20 kwietnia
Wczytywanie i walidacja danych	07 kwietnia	04 maja
Implementacja algorytmu maksymalnego przepływu	07 kwietnia	13 kwietnia
Implementacja algorytmu przepływu z kosztami	13 kwietnia	27 kwietnia
Uwzględnienie kosztów naprawy dróg	13 kwietnia	27 kwietnia

Podział mapy na ćwiartki (regiony)	12 maja	18 maja
Przeliczanie wydajności pól na podstawie regionów	12 maja	18 maja
Implementacja algorytmu Boyera–Moore'a	05 maja	11 maja
Implementacja algorytmu KMP i Huffmana	12 maja	18 maja
Implementacja algorytmu Rabina–Karpa	12 maja	18 maja
Tworzenie menu głównego aplikacji (GUI)	07 kwietnia	13 kwietnia
Rozbudowa GUI o okna wczytywania danych	13 kwietnia	27 kwietnia
Wizualizacja wyników algorytmów	05 maja	18 maja
Integracja GUI z logiką aplikacji	19 maja	25 maja
Tworzenie testów jednostkowych i szkiców testów	13 kwietnia	04 maja
Rozbudowa testów i testowanie klas algorytmicznych	05 maja	25 maja
Pomiar wydajności i przypadki brzegowe	13 kwietnia	25 maja
Wstępna dokumentacja i opis algorytmów	07 kwietnia	04 maja
Końcowy raport i instrukcja obsługi	19 maja	25 maja
Wstępny projekt prezentacji	19 maja	25 maja

6 Ogólny opis przebiegu prac nad projektem

Prace nad projektem rozpoczęły się w dniu 7 kwietnia i trwały do 25 maja. Projekt został podzielony na szereg zadań, które realizowane były etapami przez członków zespołu. Przebieg prac przedstawia się następująco:

- W początkowej fazie (07.04–13.04) skupiono się na przygotowaniu podstawowych komponentów programu, takich jak reprezentacja danych (Paweł), algorytm przepływu bez kosztu napraw (Marcin), graficzne menu główne aplikacji (Kacper) oraz wstępny zarys dokumentacji (Łukasz).
- W kolejnym etapie (13.04–27.04) kontynuowano rozwój funkcjonalności – wprowadzono możliwość parsowania plików tekstowych (Paweł), rozbudowano menu o możliwość wczytywania danych (Kacper) oraz zaimplementowano algorytm przepływu z kosztami napraw (Marcin). Dodatkowo, rozpoczęto prace nad testami (Łukasz).

- W trzecim etapie (20.04–04.05) rozwijano możliwość zapisywania danych do pliku (Paweł), uporządkowano kod (Marcin), poprawiono logikę wczytywania danych (Kacper) oraz przygotowano algorytmy geometryczne (Paweł). Równolegle kontynuowano prace nad dokumentacją oraz testami (Łukasz).
- W dniach 05.05–18.05 wdrażano bardziej zaawansowane funkcje, w tym wizualizację wyników algorytmów (Kacper), implementację algorytmów przeszukiwania tekstu: Boyera-Moore'a, Huffmana, KMP oraz Rabina-Karpa (Marcin, Kacper), a także algorytm dzielenia płaszczyzny na ćwiartki (Paweł). Kod był również optymalizowany i porządkowany.
- W ostatnim etapie projektu (19.05–25.05) przeprowadzono refaktoryzację pod kątem wielowątkowości (Paweł) oraz integracji logiki i interfejsu graficznego (Kacper). Powstał również wstępny projekt prezentacji (Marcin), a także finalna wersja dokumentacji wraz z dodatkowymi testami klas implementujących algorytmy (Łukasz).

Każdy z członków zespołu miał przypisane konkretne zadania, a prace były realizowane równolegle w sposób zorganizowany. Dzięki wyraźnemu podziałowi obowiązków i dobrej współpracy projekt został zrealizowany w zaplanowanym terminie.

7 Zastosowane algorytmy i ich złożoność obliczeniowa

W ramach projektu zaimplementowano i wykorzystano kilka klasycznych algorytmów oraz specjalistyczne procedury do analizy danych i przetwarzania tekstu. Poniżej przedstawiono zestawienie użytych algorytmów wraz z ich złożonościami obliczeniowymi:

- Algorytm przepływu w sieci (bez kosztów) – złożoność: $\mathbf{O(VE^2)}$ (dla algorytmu Edmondsa-Karpa)
- Algorytm przepływu w sieci (z kosztami napraw) – złożoność: $\mathbf{O(VE \log V)}$ (dla algorytmu min-cost max-flow z Dijkstrą i kopcem)
- Algorytm Boyera–Moore'a – złożoność: $\mathbf{O(n/m)}$ w praktyce, w pesymistycznym przypadku $\mathbf{O(nm)}$
- Algorytm Rabina–Karpa – złożoność: $\mathbf{O(n + m)}$ średnio, $\mathbf{O(nm)}$ w pesymistycznym przypadku
- Algorytm Knutha–Morrisa–Pratta (KMP) – złożoność: $\mathbf{O(n + m)}$
- Algorytm Huffmana – złożoność: $\mathbf{O(n \log n)}$ (dla n różnych znaków)
- Algorytmy geometryczne (np. podział płaszczyzny na ćwiartki) – złożoność: $\mathbf{O(n)}$ (dla iteracyjnego podejścia do klasyfikacji punktów)

Gdzie:

- n – długość przeszukiwanego tekstu lub liczba danych wejściowych
- m – długość wzorca
- V – liczba wierzchołków w grafie
- E – liczba krawędzi w grafie

8 Poprawność rozwiązania

???

9 Rodzaj wykorzystanej technologii, języków programowania

- **Backend:** Java
- **Frontend:** JavaFX (desktopowa aplikacja graficzna)
- **System kontroli wersji:** Git (repozytorium hostowane na GitHub)
- **Zarządzanie zależnościami i budowaniem projektu:** Maven
- **Ułatwienia w kodzie:** Lombok (automatyzacja generowania kodu — np. getterów, setterów)

10 Informacja dodatkowa

???