Politechnika Wrocławska

Projektowanie Algorytmów i Metody Sztucznej Inteligencji

Raport z projektu I

Data:	05.04.2022
Wykonał:	Mateusz Malik, 249425
Prowadzący:	dr hab. inż. Andrzej Rusiecki
Termin zajęć:	Poniedziałek, 9:15 – 11:00

Spis treści

Wprowadzenie

Definicją programów wg. Niklausa Wirth'a jest suma algorytmów i struktur danych, a pierwszy projekt kursu Projektowania Algorytmów i Metod Sztucznej Inteligencji jest jej bardzo dobrym odzwierciedleniem. Jest to spowodowane naciskiem narzucanym przez zakładaną funkcjonalność projektu na implementację zarówno abstrakcyjnej struktury danych, jak i algorytmu stwarzającego morzliwość jej posortowania.

Opis wymaganych funkcjonalności

Założeniem projektu było stworzenie programu, który podzieli wiadomość tekstową na ponumerowane, lecz zapisane w losowej kolejności pakiety danych a także dokona ich scalenia w oryginalny tekst, co ma symulować odbieranie i obsługę pakietów danych przesyłanych w sieci. Ilość paczek powinna być dowolna, a program ma być przygotowany na dynamiczne ich alokowanie. Struktura danych przechowująca paczki powinna również dawać możliwość odpowiedniej modyfikacji w celu posortowania – niedozwolone było przepisanie danych do tablicy w celu przetworzenia. Dodatkowymi wymaganiami był brak użycia bibliotek STL zawierających gotowe potrzebne struktury danych, które należało zaimplementować samodzielnie.

Podział na podproblemy

Pierwszym najważniejszym zadaniem projektu było przekucie założeń dotyczących funkcjonalności na plan programu. Sporządzono zatem następującą listę podproblemów projektu:

- Stworzenie struktury danych umożliwiającej przechowywanie pary zmiennych składającej się z ciągu znaków oraz liczby, operacje na elementach o dowolnych indeksach oraz zapewniającej dynamiczne alokowanie pamięci i względnie dowolny rozmiar.
- Zaimplementowanie dowolnego algorytmu sortowania, będącego w stanie operować na stworzonej strukturze.
- Stworzenie metody pozwalającej na podzielenie tekstu z pliku na fragmenty, ponumerowanie oraz zasymulowanie losowości kolejności.
- Stworzenie metody pozwalającej na scalenie posegregowanych już fragmentów.

Opis struktur i algorytmów

Strukturą, na którą zaimplementowano jest wektor, działający w analogiczny sposób jak w przypadku klasy vector dostępnej w STL, czyli będący dynamicznie alokowaną tablicą zwiększającą rozmiar w zależności od zapotrzebowania. Ze względu na możliwość łatwego dostępu do dowolnego pola jest to struktura optymalnie spełniająca założone wymagania. Nie wymaga usuwania, lub przechowywania w innej strukturze elementów dodanych jako ostatnie, tak jak w przypadku np. stosu, ale również nie ma z góry narzuconego rozmiaru, jak to jest w przypadku tablicy.

Zastosowanym algorytmem sortowania jest Quick Sort z losowaną pozycją pivota. Został on wybrany ze względu na wypośrodkowanie między skomplikowaniem implementacji (łatwiejszy do zaimplementowania niż np. intro sort), złożonością pamięciową (nie potrzebuje tablicy pomocniczej jak Merge Sort) oraz złożonością obliczeniową, która w przypadku Quick Sorta wynosi według notacji dużego O: $O(n\,logn)$ dla wariantu korzystnego. Ryzyko wystąpienia wariantu niekorzystnego, przy którym złożoność Quick Sorta sięga $O(n^2)$ zostało zminimalizowane poprzez losowanie elementu osiowego z wykorzystaniem funkcji rand().

Odczyt tekstu wiadomości został zrealizowany jako odczyt pliku tekstowego, który następnie jest dzielony na fragmenty za pomocą funkcji getline() z wykorzystaniem argumentu znaku rozdzielającego kolejne ciągi znakowe, na jaki została ustawiona kropka.

Wymieszanie kolejności pakietów odbywa się poprzez wylosowanie dwóch indeksów z zakresu zapisanych elementów w wektorze za pomocą funkcji rand() a następnie zamianę ich miejscami. Czynność zostaje powtórzona zadaną w funkcji main ilość razy.

Analiza złożoności obliczeniowej programu

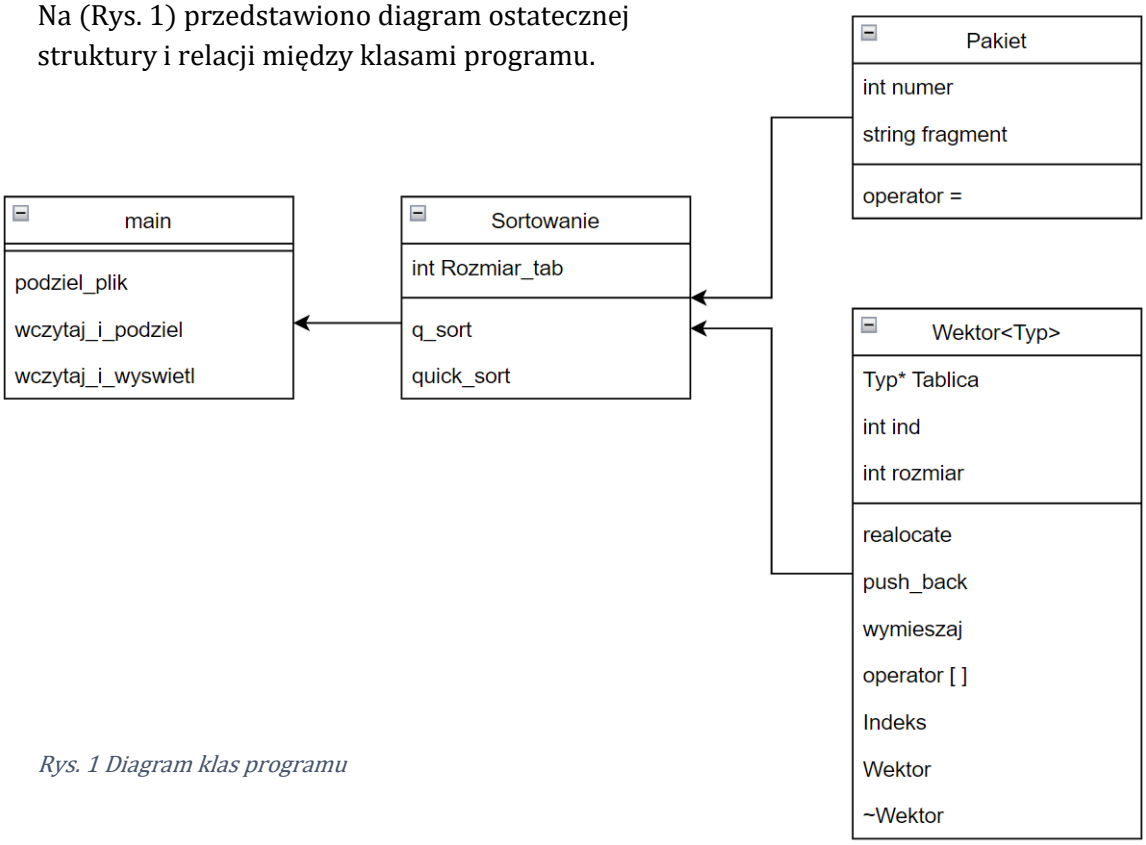
W zależności od danej ilości n pakietów, program musi wykonać asymptotycznie:

- $x \cdot n$ operacji przy odczycie, podzialne, mieszaniu, scalaniu tekstu
- $x \cdot nlog(n)$ operacji przy sortowaniu oraz dynamicznym alokowaniu pamięci

co daje w notacji dużego O sumaryczną złożoność $O(n\ log\ n)$. Złożoność pamięciowa jest w przybliżeniu liniowa i w notacji dużego O wynosi odpowiednio $O(n)$.

Struktura programu

Na (Rys. 1) przedstawiono diagram ostatecznej struktury i relacji między klasami programu.



Rys. 1 Diagram klas programu

Działanie programu

Program po uruchomieniu wypisuje na standardowe wyjście trzy zasadnicze bloki tekstu:

1. Tekst oryginalny w całości, bez przetworzenia.
2. Nieuporządkowaną listę składającą się z numerów oraz ciągów znakowych sporządzonych z tekstu oryginalnego.
3. Analogiczą listę, lecz tym razem już uporządkowaną.
4. Tekst ciągły po odtworzeniu.

Nazwa pliku, z którego ma nastąpić odczyt jest wpisana w kodzie programu, natomiast jeśli nie uda się go poprawnie otworzyć program będzie wymagał podania nowej nazwy, do skutku. Poza tym nie ma więcej interakcji użytkownika z programem.

Środowisko uruchomieniowe oraz repozytorium

Program został uruchomiony w systemie operacyjnym Windows 10, a do kompilacji posłużył kompilator g++ z użytymi flagami -Iinc -Wall oraz -pedantic. Plik Makefile użyty do kompilacji, oraz wszystkie pliki zawierające nagłówki i kod źródłowy są zamieszczone w repozytorium znajdującym się pod poniższym hiperłączem:

https://github.com/malikerro/PAMSI_projekt1.git

Literatura

- http://lukasz.jelen.staff.iiar.pwr.edu.pl/styled-2/page-2/index.php
- http://www.algorytm.edu.pl/algorytmy-maturalne/quick-sort.html
- https://pl.wikipedia.org/wiki/Sortowanie
- https://pl.wikipedia.org/wiki/Sortowanie_szybkie
- https://www.youtube.com/watch?v=2s717IFZLuU&t=3s
- https://www.youtube.com/watch?v=82XxdhRCMbI&t=88s