Projektowanie Algorytmów i Metody Sztucznej Inteligencji
Projekt I - zadanie na ocenę **BDB**

Wiktor Kwiatkowski 275423
Wtorek 15:15 - 16:55 grupa nr.6

Data oddania sprawozdania: 15.04.2024 r.

1 Proponowane rozwiązanie problemu

Aby zasymulować przesył wiadomości przez internet, należy najpierw przygotować plik tekstowy pozbawiony polskich znaków specjalnych, gdyż może to powodować różnego rodzaju problemy podczas próby odczytu/zapisu. W drugiej kolejności należy sukcesywnie odczytywać zawartość pliku, dzieląc tekst na pakiety o zadanym rozmiarze i nadając każdemu pakietowi indywidualny numer odczyty, który później posłuży nam do „odszyfrowania" przesłanej wiadomości. Gdy suma wielkości pakietów będzie równa rozmiarowi wiadomości, którą chcemy przesłać, trzeba zasymulować losowość przesyłu, czyli mieszamy kolejność, w jakiej nasze pakiety zostały odczytane z pliku, następnie wysyłamy do naszego ADT. Zadaniem naszej struktury jest odczytanie pakietów i ułożenie ich w kolejności według priorytetu nadania, aby finalnie otrzymać spójną wiadomość, która w całości ma sens logiczny niezależnie od tego, w jaki sposób została „zamieszana" przed nadaniem.

2 Wybrana struktura

Do rozwiązania problemu postanowiłem zastosować kolejkę priorytetową opartą na liście jednokierunkowej. Kluczowe w tym zadaniu jest to, że nasza struktura musi się dynamicznie dostosowywać do wielkości danych na wejściu, dzięki wykorzystaniu listy jest to możliwe do realizacji. W przeciwieństwie do założeń w „milestone" nie użyłem listy dwukierunkowej ze względu na charakterystykę zadania, możliwość poruszania się w obie strony była zbędnym zabiegiem, co tylko komplikowało logikę samej struktury. Kolejność, w jakiej elementy zostaną dodane do ADT, determinuje wcześniej wspomniany priorytet, czyli wartość porządkująca nadawana podczas symulowania przesyłu danych.

2.1 Sposób implementacji

Kolejka priorytetowa bazująca na liście jednokierunkowej jest zbudowana z węzłów(Node), które zawierają wskaźnik na poprzedzający je element(Node *next), zmienną do przechowywania pakietu(std::string packet) oraz wartość klucza(unsigned int priority).
Dodatkowo w klasie PriorityQueue znajduję się wskaźnik(Node *head) zawierający informację o początku listy.

2.1.1 Metody kolejki priorytetowej

- **void push(unsigned int priority, const std::string &packet)**
  Tworzy nowy węzeł i inicjalizuje go priorytetem i danymi zgodnie z przekazanymi argumentami. Jeśli priorytet nowego elementu jest niższy niż priorytet obecnej głowy kolejki (head), nowy węzeł staje się nową głową kolejki, a jego wskaźnik next wskazuje na poprzednią głowę. W przeciwnym razie iteruje przez listę, dopóki nie znajdzie odpowiedniego miejsca dla nowego elementu. Nowy węzeł jest wstawiany przed węzłem o niższym priorytecie i po węźle o wyższym priorytecie.

- **bool isEmpty() const**
  Sprawdza, czy wskaźnik head równy jest nullptr. Jeśli tak, oznacza to, że kolejka jest pusta i zwraca true.

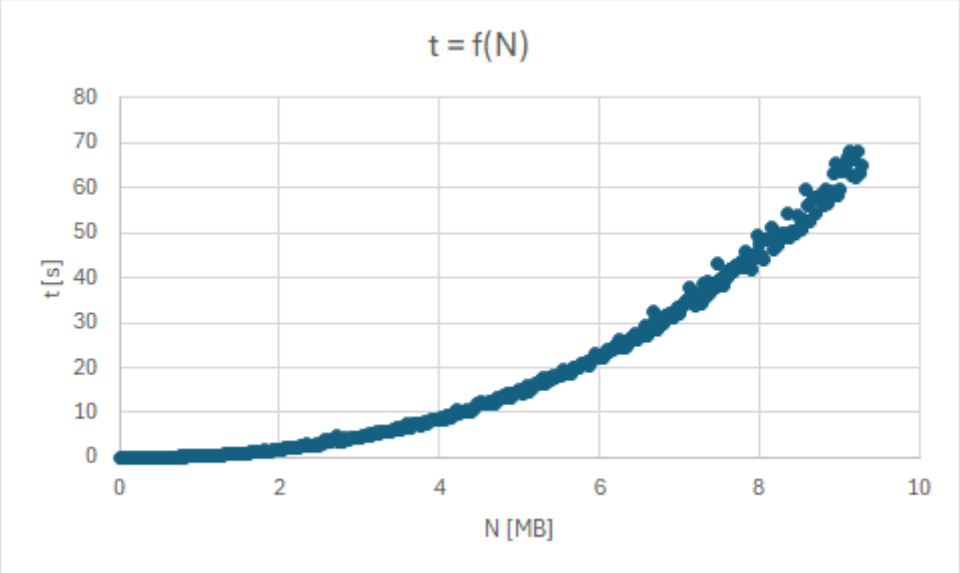
- **void pop()** Usuwa pierwszy element (o najwyższym priorytecie).
- **void clear();**
  Usuwa wszystkie elementy, powodując, że kolejka staje się pusta i zwalnia zaalokowaną pamięć.
- **void display()**
  Służy do wyświetlania zawartości kolejki priorytetowej na standardowym wyjściu.

3 Złożoności obliczeniowa

Analizie czasowej został poddany główny problem tego zadania, czyli odbiór i posortowanie pakietów aż do momentu wyświetlenia tekstu na standardowym wyjściu.

Metoda/Funkcja	Złożoność obliczeniowa	Wyjaśnienie
void push()	O(n)	W najgorszym przypadku musi przeszukać całą kolejkę.
void receivePacket()	$O(n^2)$	W najgorszym przypadku funkcja wykona n razy metodę push().
displayReceived()	O(n)	Funkcja musi przejść przez każdy węzeł w kolejce.

Zgodnie z tabelą złożoność obliczeniowa procesu wynosi: $n+n^2 + n = n^2 + 2n = \mathbf{O(n^2)}$.



Rysunek 1: Zależność czasu od rozmiaru wiadomości dla stałej wielkości pakietu 100 B.

Na wykresie widać krzywą, która swoim wyglądem przypomina zależność kwadratową. Jest to wynik, którego mogliśmy się spodziewać na podstawie złożoności obliczeniowej z tabeli powyżej. Nie jest to idealna zależność kwadratowa z powodu stałego rozmiaru pakietu, im mniejszy byłby rozmiar pakietu podczas testów, tym bardziej krzywa rosła by kwadratowo. Otrzymane wyniki zostały osiągniete na procesorze o taktowaniu od 1.3 GHz do 4.4 GHz.

4 Wnioski

Złożoność obliczeniowa programu wynosi O(n^2), nie jest to najgorszy wynik, lecz istnieją efektywniejsze metody na rozwiązanie tego zadania. Dokładniejsza analiza wykazuje, że największy ciężar obliczeniowy znajduje się w symulacji przesyłu, aniżeli samej strukturze danych. Najgorszy przypadek może wystąpić w momencie, gdy algorytm będzie musiał dodać duża ilość węzłów(mały rozmiar pakietów). Niemniej jednak program, jak i sama struktura działają poprawnie i spełniają założone im wymagania.

5 Informacje dodatkowe

- Zadanie zostało wykonane na Ubuntu w wersji 22.04.
- Wersja kompilatora GCC 11.40.
- Dane z wykresu zostały otrzymane przy pomocy skryptu napisanego w bashu.

6 Źródła

1. `https://tinyurl.com/ysac68s6`
2. `https://www.geeksforgeeks.org/priority-queue-using-linked-list/`
3. `hhttps://chat.openai.com/`