Politechnika Wrocławska

Wydział Elektroniki, Fotoniki i Mikrosystemów

Projektowanie algorytmów i metody sztucznej inteligencji

Projekt 1

Zadanie na ocenę bdb

Prowadzący:
Dr inż. Łukasz Jeleń

Wykonała:
Zuzanna Mejer, 259382

Wrocław, 31 marca 2022r.

1. Wstępne założenia

Zadanie polegało na zaprojektowaniu i zaimplementowaniu algorytmu, który radziłby sobie z sortowaniem otrzymanych pakietów składających się z numeru i tekstu wiadomości. Do wykonania zadania przyjęto pewne założenia. Pierwszym z nich jest przyjęcie perspektywy Anny i jej komputera. Oznacza to tyle, że napisany program nie zajmuje się dzieleniem wiadomości na n pakietów oraz nie nadaje im numerów, gdyż wymienione czynności wykonywał, zgodnie z poleceniem, Jan na swoim komputerze. Zatem program został przygotowany tak, żeby pracować już z losowo ułożonymi i ponumerowanymi pakietami. Drugim przyjętym założeniem jest przesyłanie tych pakietów w jednym pliku tekstowym jako drugi argument wywołania podczas uruchamiania pliku wykonywalnego. Kolejnym założeniem jest forma zakończenia programu i wyświetlenie posortowanej wiadomości. Zdecydowano, że posortowana wiadomość nie tylko będzie wyświetlana na standardowym wyjściu, ale również będzie zapisywana do nowo tworzonego pliku nazwanego „uporządkowany_list.txt".

2. Struktura danych

W celu zrealizowania zadania wybrano strukturę danych do przechowywania informacji (numeru i tekstu wiadomości), jaką jest kolejka priorytetowa na bazie listy dwukierunkowej. Poniżej przedstawiono uzasadnienie wyboru. Kolejka priorytetowa jest abstrakcyjnym typem danych, służącym do przechowywania zbioru elementów, przy czym każdy element posiada dodatkowe pole do przechowywania priorytetu lub inaczej klucza. Zatem, z założenia, do kolejki priorytetowej można wprowadzać takie pakiety, składające się z numeru (priorytetu) i tekstu. Kolejnym argumentem przemawiającym za użyciem kolejki priorytetowej jest układanie elementów w kolejce nie w kolejności wprowadzania, a w kolejności priorytetu rosnąco lub malejąco. Jako implementację kolejki priorytetowej wybrano listę dwukierunkową. Umożliwia ona wstawianie elementu w dowolnym miejscu, nie tylko na początku lub na końcu listy, co pozwala na wydajne sortowanie elementów przy ich wstawianiu do kolejki, a więc tworzenie listy uporządkowanej. Analogicznie, lista dwukierunkowa pozwala także na usuwanie elementów z dowolnego miejsca.
Operacje na kolejce priorytetowej uwzględniają takie metody jak:

- umieszczanie nowego elementu w kolejce,
- usuwanie elementu z kolejki,
- zwracanie informacji o tym, czy kolejka jest pusta czy nie,
- informowanie o liczbie elementów umieszczonych w kolejce,
- zwracanie wartości pierwszego elementu w kolejce.

Zaimplementowanie powyższych metod będzie omówione poniżej.

3. Omówienie programu

3.1. Obsługa plików

Do obsługi plików została stworzona klasa o nazwie *file* z dwoma prywatnymi polami określającymi plik wejściowy *in_file* oraz plik wyjściowy *out_file*. Stworzone zostały metody służące do otwierania i zamykania plików wejściowych i wyjściowych (*open_in_file, open_out_file, close_in_file,*

close_ out_ file), czytania (*read_ file*) i sprawdzania końca (*end_ of_ file*) pliku wejściowego oraz zapisywania do pliku wyjściowego (*write_ out_ file*).
+ listing kodu
Plik wejściowy podawany przy uruchomieniu programu ma domyślnie formę przedstawioną na poniższym zdjęciu. Na początku linii jest numer pakietu, a po spacji wiadomość.
+ zdjęcie formy pliku

3.2. Kolejka priorytetowa jako lista dwukierunkowa

Do zaimplementowania kolejki na liście dwukierunkowej, została stworzona struktura *node*, która ma dwa pola do przechowywania pakietu: *key* oraz *text*, oraz dwa wskaźniki na następny i poprzedni element: **next* i **prev*. Została stworzona klasa *priority_ queue* z pierwszym wskaźnikiem na początek kolejki: **header* typu struktury *node*. Zostały zaimplementowane metody:

- *comparison* - służy do porównywania wartości dwóch kluczy i jest wykorzystywana przy sortowaniu elementów podczas wprowadzania do kolejki.
- *insert* - odpowiednik metody *push*; służy do wstawiania elementów do kolejki. Metoda *insert* różni się tym, że umożliwia umieszczenie elementu w dowolnym miejscu w kolejce, tak, aby tworzyć kolejkę uporządkowaną rosnąco. Przykładowo, jeżeli do metody *insert* zostanie przekazany pakiet z pliku o wartości klucza 4, a w kolejce będą już elementy o wartościach kluczy 1,2,6,8, to metoda ta, wykorzystując metodę *comparison* znajdzie pierwszą większą wartość klucza od pobranej (4) i wstawi ją przed ten element (6).
- *display_ text* - odpowiednik metody *top*; odpowiada za wyświetlanie samego tekstu pierwszego elementu w kolejce.
- *remove_ minimum* - odpowiednik *pop*; służy do usuwania elementu z początku kolejki.
- *empty* - zwraca informację o tym, czy kolejka jest pusta.
- *return_ minimum* - jest funkcją pomocniczą do zwracania tekstu pierwszego elementu w kolejce. Wykorzystuje ją metoda zapisywania do pliku.
- *size* - zwraca ilość elementów umieszczonych w kolejce priorytetowej.

+listing kodu

3.3. Działanie programu

Program wykonuje następujące działania:

1. Otwiera zarówno plik wejściowy, podany jako argument wywołania pliku wykonywalnego, jak i wyjściowy, do którego będą zapisywane dane.
2. Aż do natrafienia na koniec danych w pliku, program sczytuje kolejne sekwencje, tworzy nowy element typu *node*, przypisuje do niego wartości klucza i wiadomości, a następnie wstawia w odpowiednie miejsce w kolejce, sortując rosnąco względem wartości klucza.
3. Kiedy nie ma więcej danych w pliku i kolejka jest już utworzona, następuje wyświetlenie tekstu elementu o najmniejszej wartości klucza na standardowe wyjście, zapisanie go do pliku wyjściowego oraz usunięcie go z kolejki priorytetowej. Ten etap powtarza się aż do wyczyszczenia całej kolejki.

+listing kodu

4. Analiza złożoności obliczeniowej

5. Testy

6. Podsumowanie i wnioski

7. Literatura

1. Wykład „Projektowanie Algorytmów i Metody Sztucznej Inteligencji”
2. M.T.Goodrich, R. Tamassia, D.Mount „Data Structures & Algorithms in C++”
3. `https://cpp0x.pl/kursy/Kurs-STL-C++/Adapter-kolejki-priorytetowej-std-priority_queue/118` (dostęp: 31.03.2022)