W12n, AIR Projektowanie Algorytmów i Metody Sztucznej Inteligencji	Sprawozdanie nr 1 Temat: Kolejka Priorytetowa
Jakub Cebula	Wt 15:15, 28 marca 2023 Dr inż. Marek Bazan

Spis treści

1 Treść Zadania:

Załóżmy, że Jan chce wysłać przez Internet wiadomość W do Anny. Z różnych powodów musi podzielić ją na n pakietów. Każdemu pakietowi nadaje kolejne numery i wysyła przez sieć. Komputer Anny po otrzymaniu przesłanych pakietów musi poskładać je w całą wiadomość, ponieważ mogą one przychodzić w losowej kolejności. Państwa zadaniem jest zaprojektowanie i zaimplementowanie odpowiedniego rozwiązania radzącego sobie z tym problemem. Należy wybrać i zaimplementować zgodnie z danym dla wybranej struktury ADT oraz przeanalizować czas działania - złożoność obliczeniową proponowanego rozwiązania. W sprawozdaniu (3-5 strony + strona tytułowa) należy opisać rodzaj wybranej struktury danych wraz z uzasadnieniem wyboru oraz opisać proponowane rozwiązanie problemu. Należy także opisać sposób analizy złożoności obliczeniowej i podać jej wynik w notacji dużego O.

2 Rodzaj wybranej struktury danych:

Do wykonania powyższego zadania wybrałem strukturę danych taką jak kolejka priorytetowa. Struktura charakteryzuje się tym, że każdemu obiektowi jest przyporządkowana wartość- "priorytet". Kolejka priorytetowa może być typu min lub max. Wykorzystałem kolejkę posortowaną typu min co oznacza, że kolejka zwraca jako pierwszy element o najniższym kluczu(priorytecie).Elementy są sortowane podczas dodawania ich do kolejki.
Przesyłana wiadomość jest dzielona na pakiety ze względu na ustalony rozmiar. Każdemu pakietowi zostaje przypisany priorytet, ponieważ pakiety mogą przychodzić w losowej kolejności. Kolejka priorytetowa typu min w tym przypadku według mnie jest najlepszym rozwiązaniem, dlatego że pakiety będą numerowane w kolejności -1,2,3,4,...,n, więc zwracanie pakietów będzie w kolejności od początku do końca. Taka implementacja pozwoli odebrać wiadomość w poprawnej kolejności.

2.1 Sposób implementacji:

Template klasy kolejki priorytetowej typu min (Queue) składa się z:

- Konstruktora Queue, który przypisuje rozmiar kolejki jako 0, a węzły head i tail jako pusty wskaźnik na nullptr,
- metody insert, która zawiera obiekt oraz jego priorytet, dzięki któremu obiekt jest wstawiany w odpowiednim miejscu,
- metody min, zwraca element o najniższym kluczu,
- metody removeMin, która usuwa i zwraca element o najniższym kluczu,
- metoda isEmpty typu bool, która informuje gdy kolejka jest pusta (true), w przciwnym wypadku (false),
- metdy size, która podaje rozmiar kolejki,
- metody print, która na ekranie wyświetla w jaki sposónb zmienia się kolejka,
- struktury node, która definiuje cały węzeł czyli priorytet, obiekt i dwa wskaźniki (*next i *prev) oraz konstruktora,
- wskaźnika *head, który wskazuje na początek kolejki,
- wskaźnika *tail, który wskazuje koneic kolejki,
- pola int _size, które zawiera informacje ile obiektów jest w kolejce.

```
template<class T>
class Queue
{
public:
Queue(): head(nullptr), tail(nullptr), _size(0) {};
bool isEmpty() const {return head==nullptr;}
void insert(unsigned int priorytet ,const T& element);
const T min() const;
const T removeMin();
unsigned int size() const { return _size; };
void print() const;


private:

struct node
{

T element;
unsigned int priorytet;
node* prev;
node* next;
node(unsigned int p,const T& ele, node*next, node* prev):priorytet(p), element(ele), next(nullptr), prev(nullptr){}
};

node* head;
node* tail;
int _size;



};
```

Rysunek 1: Szablon klasy kolejki priorytetowej.

3 Proponowane rozwiązanie

Tworzymy plik z wiadomością, którą chcemy przesłać. Ze względu na podział pakietów o rozmiarze np. 10 bajtów to każdy plik otwieramy w trybie binarnym.
Podczas podziału wiadomości na pakiety od razu każdemu pakietowi jest nadany priorytet.Następnie funkcje swap i srand losowo mieszają pakiety i zapiusją do osobnego pliku w celu sprawdzenia poprwności problemu.
Kolejno z pomieszanego pliku wczytujemy w pętli paczki o określonym rozmiarze wraz z ich priorytetem i umieszczamy je w kolejce priorytetowej za pomocą metody insert. Gdy wszystkie paczki zostaną umieszczone w kolejce to za pomocą metody removeMin (również w pętli) usuwamy paczke o najniższym priorytecie i zapisujemy ją do zmienne std::string wiadomosc. Na sam koniec zostało tylko wiadomosc za pomocą operatora strumienia wyjsciowego skierować do pliku z odebraną wiadomością.

4 Złożoność obliczeniowa

Złożoność obliczeniowa określa, ile głównych operacji musi wykonać algorytm, aby rozwiązać problem dla n elementów będących danymi wejściowymi. Elementami tymi mogą być liczby, znaki itd. Główną operacją może być porównywanie elementów i ich przestawienie (np. w algorytmach sortujących), dodawanie, mnożenie itp. Złożoność obliczeniowa kodu w notacji dużego O jest taka jak największa złożoność jaka znajduje się w danym kodzie.

4.1 Złożoność metod klasy:

- metody isEmpty,min,size oraz removeMin mają złożoność obliczeniową O(1)-ilość operacji podstawowych jest stała,
- metody insert i print mają złożoność obliczeniową O(n) - są zależne od ilości.

4.2 Złożoność funkcji main

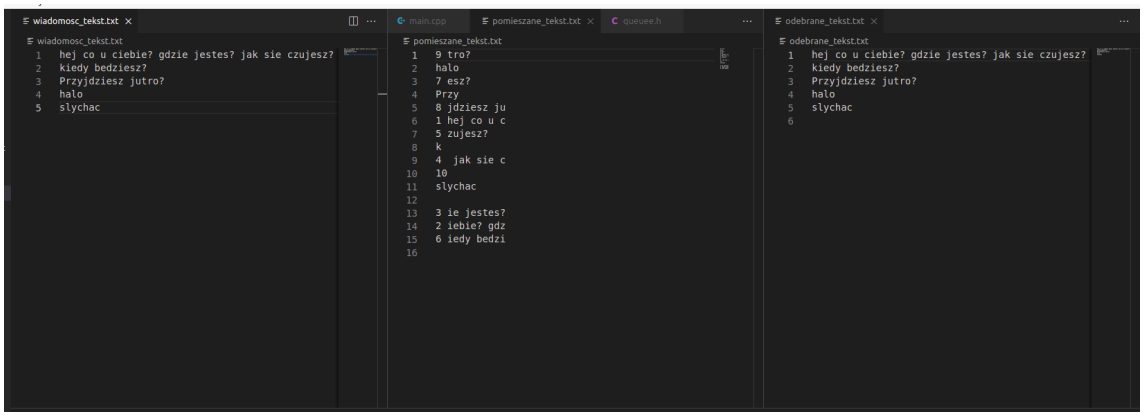
W funkcji main wczytywanie w pętli pakietów z pliku jest zależne od ilości pakietów. To znaczy, że gdyby ten fragment kodu tylko wczytywał pakiety miałby złożoność obliczeniową O(n). Ten fragment kodu w pętli również korzysta z funkcji insert, która także ma złożonosc obliczeniową równą O(n). Z tego wynika, że złożoność obliczeniowa całego algorytmu wynosi O(n^2).

```
plik_wys.open(nazwa_pliku_do_wsylania); //otwieranie pliku
if(plik_wys.good()) //sprawdzenie
    {

        if (plik_wys.is_open())
        {

        std::string slowo;
        while ((plik_wys>>a))
        {
        slowo=""; //zmienna do której zapisujemy pakiet
        plik_wys.get(c); //bierzemy jeden znak spacji
        for(int i=0;i<10;i++) //pętla wczytuje 10bajtów
        {
            plik_wys.get(c); //bierzemy jeden zank i go sprawdzamy
            if(c!='\n') //jeśli to nie znak nowej linii
            slowo+=c; //to dodajemy go do słowa
            else
            {
                slowo+=c; //jeśli to znak nowej linii to ją dodajemy
                plik_wys.get(c); //znowu sprawdzamy
                if(c!='\n')
                plik_wys.unget(); //umieszcza ostatni odczytany znak spowrotem w strumieni wejściowym
                else
                break; //przerwanie
            }

        }

        kolejka_pakietowa.insert(a, slowo); //dodawanie do kolejki
        kolejka_pakietowa.print(); // pokazanie jak wygląda kolejka
        }
        }

            plik_wys.close();


        }
```

Rysunek 2: fragment kodu decydujący o złożoności obliczeniowej-O(n^2)

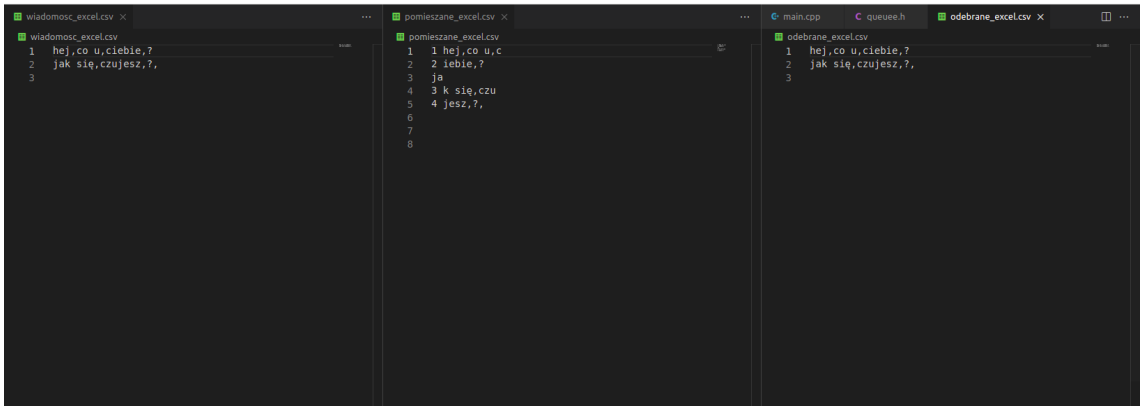
5 Testy

5.1 Test pliku typu tekstowego:



Rysunek 3: Plik typu txt

5.2 Test pliku typu excel:



Rysunek 4: Plik typu excel