Projektowanie algorytmów i metody sztucznej inteligencji - Raport 1

Damian Ryś

26 kwietnia 2021

Spis treści

1 Wstęp

Grupa lab: E12-93c zmien
Termin zajęć: PN 18:45-20:35
Numer indeksu: 252936
Prowadzący: Dr inż. Piotr Ciskowski

2 Zadanie

Postawionym pzed nami problemem jest wysłanie przez użytkownika A do użytkownika B wiadomości przez Internet. wiadomość ta powinna zostać wysłana w serii pakietów na komputer użytkownika B. Problem polega w tym, że pakiety mogą przychodzić z różnych przycznym w losowej kolejności do użytkownika B. Nasz program powinien zatem:

1. Podzielić naszą wiadomość na szereg pakietów
2. Przelosować pakiety w celu zasymulowania sytuacji
3. Wczytać do struktury danych nasze pakiety
4. Przesyłać pakiety w foramcie [klucz,wartość]
   (a) klucz powininen mieć unikalną wartość
   (b) wartość powinna być dowolnego typu (użytkownik powinine mieć możliwość przesłania dowolnej wiadomości)
5. Złączyć wiadomość w jedną,prawidłową całość i wyświetlić użytkownikowi

3 Kolejka Priorytetowa

Wobec postawionego problemu strukturą ADT, którą wybrałem jest kolejka priorytetowa bazowana na liście.

3.1 Czym jest kolejka priorytetowa?

Kolejka priorytetowa (ang. priority queue) jest kolejką, w której elementy są ułożone nie w kolejności wprowadzania, lecz w kolejności priorytetu. Każdy element kolejki posiada dodatkowe pole priority, w którym przechowuje swój priorytet – czyli ważność. Gwarantuje to pobieranie z kolejki jako pierwszych elementów o najwyższym priorytecie. Elementy o priorytetach niższych zostaną pobrane dopiero wtedy, gdy zostaną usunięte wszystkie elementy o priorytetach wyższych.

3.2 Zalety i wady

Strukturę tą wybrałem ze względu na:

- Łatwość implementacji
- Złożoność czasową lepszą niż O(n) dla uzyskania min oraz max
- Dynamiczną alokację danych

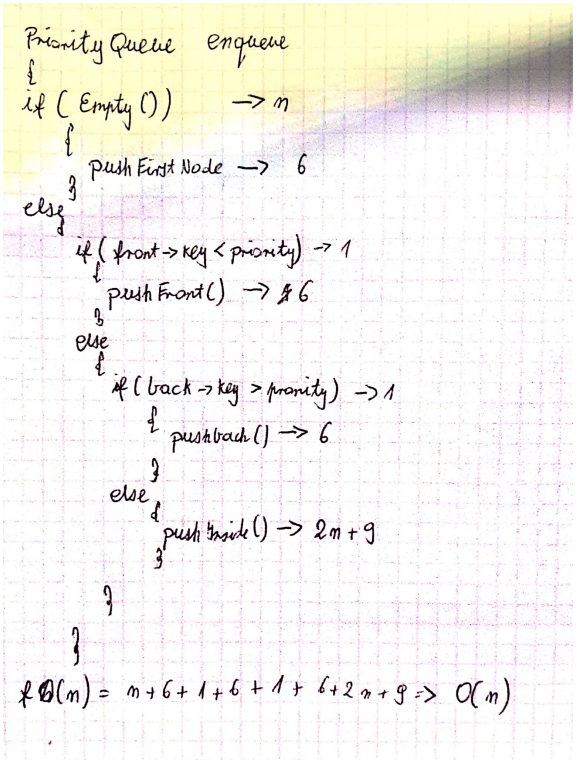
3.3 Implementacja

```
template<typename T>
class PriorityQueue
{
    typedef struct Node
    {
        T field; //wartosc
        int key; // klucz
        Node *nextNode;
    } *nodePointer;

    nodePointer front; //wskaznik na nastepny element
    nodePointer back; // wskaznik na poprzedni element
public:
    PriorityQueue(); //konstruktor
    ¬PriorityQueue(); //destruktor
    // dodanie elementu
    void enqueue(const T& newElement, int priority);
    // usuniecie elementu
    void dequeue();
    // wyswietlenie kolejki w prawidlowym porzadku
    void Print();
    // sprawdzenie czy kolejka nie jest pusta
    bool Empty();
private:
//metody do obslugi enqueue i dequeue
    void pushBack(const T& newElement, int priority);
    void pushFront(const T& newElement, int priority);
    void pushFirstNode(const T& newElement, int priority);
    void pushInside(const T& newElement, int priority);

};
```

Powyżej znajduję się klasa kolejki priorytowej, reszta kodu dostępna jest na rezpozytorium na Githubie lub dołączonym skompresowanym pliku.

3.4 Złożoności obliczeniowe

Dla naszego programu zgodnie z załozeniami projektowymi zostały obliczene złożoności czasowe poszczególnych funkcji naszej struktury:



Rysunek 1: Złożoność obliczeniowa dla metody enqueue

Tak przedstawia się reszta złożoności oblicziowych naszej struktury: //tabelka

Nasze wyniki pokrywają się z danymi dostępnymi w internecie,jak na przykład TUTAJ. Uznaje zatem, że kolejka została zaimplementowana poprawnie.

3.5 Testy

Za bazę naszych testów posłużyły nam "polskie copy pasty". Teksty zostały podzielone,następnie wymieszane, załadowane do strktury, by finalnie zostały posortowane i złączone w jedną całość w osobnym pliku wynikowym. Na poniższym przykładzie został zaprezentowany cykl życia naszego pliku.

Program finalnie przeszedł testy na bazie większej ilości plików i był w stanie prawidłowo odtwrzyć plik, zatem uznaje, że program działa prawidłowo.

4 Wnioski

5 Bibliografia

Schemat implementacji wraz z schematami blokowymi
Inne rozwiązanie problemu na drzewach binarnych
Złożoności czasowe różnych algorytmów