Imię Nazwisko: Kacper Brodziak

Nr albumu: 258978

Prowadzący: Dr inż. Andrzej Rusiecki

Grupa: Y03-51f

Termin: 13:15 – 15:00

Projektowanie Algorytmów i Metody Sztucznej Inteligencji

Projekt nr 1

1. Opis projektu

Założeniem projektu jest zaprojektowanie i zaimplementowanie rozwiązania radzącego sobie z ustalaniem priorytetu wysyłanych wiadomości oraz późniejsze ich odczytywanie w prawidłowej kolejności – pozwalającej poprawnie odczytać treść wysłanej wiadomości. Całość projektu opierać się będzie na strukturze ADT (ang. Abstract Data Type). Zakładamy przy tym, że z różnych powodów wiadomość musi być podzielona na pakiety. Każdemu pakietowi nadany jest numer przed wysłaniem, a następnie po odebraniu pakiety są składane i odczytywane w poprawnej kolejności.

2. Opis zastosowanej struktury danych

Kolejka jest kolekcją obiektów bazującą na zasadzie FIFO (First In First Out) dla operacji dodawania, bądź usuwania elementów. Operacje te bardzo często nazywa się enqueue oraz dequeue. Różnicą, która przemawia na korzyść kolejki w zestawieniu z listami oraz tablicami polega na braku możliwości losowego dostępu znajdującego w strukturze obiektu. Często kolejki możemy porównywać do stosów, ale jest jednak spora różnica. W przypadku kolejki stosujemy zasadę pierwszy wszedł, pierwszy wyszedł (FIFO), z kolei za pomocą stosu usuwamy ostatnio dodany element (LIFO). Przejdźmy teraz do wydajności. W stosunku do kolejki oczekuje się, że implementacja zajmie O(1) czasu na wstawianie i usuwanie. Te dwie operacje wykonywane w są w kolejce i zawsze są szybkie w przypadku poprawnej implementacji. Kolejki bardzo dobrze odnajdują się w algorytmach szeregowania oraz w programowaniu równoległym. Przykładem algorytmu może być BFS, czyli przeszukiwanie wszerz w strukturze danych drzew lub grafu.

Szczególnym przykładem kolejki, jest kolejka priorytetowa. Różnica w tym przypadku polega na dodatkowej możliwości nadawania priorytetu, co jest ograniczeniem w standardowej kolejce. Udogodnienie to pozwala zachować porządek, mimo operacji z kolejką. Problem z zmianami, mieszaniem, przesuwaniem przestają być już uciążliwe. Zastosowaniem kolejek priorytetowych są algorytmy planowania. Przykładowo zamiast pobierać następny element, kolejka pobiera od razu element o najwyższym priorytecie. O priorytecie elementów zawsze decyduje kolejka na podstawie zastosowanych w niej kluczach. Do ograniczeń kolejki możemy na pewno zaliczyć brak

możliwości zmiany kolejności obiektów w strukturze danych, wynika z tego, że wkładając elementy w odpowiedniej kolejności otrzymamy je w niezmienionym stanie. Jako strukturę kolejki możemy potraktować zwykłą listę, ale nie jest to najlepsze rozwiązanie pod względem wydajności. W takich przypadkach listy są dużo wolniejsze. Powód jest prosty: w przypadku wstawienia lub usunięcia elementu konieczne staje się przesunięcie wszystkich pozostałych elementów listy o jeden. Jest to bardzo czasochłonne i zajmuje O(n) czasu. Z tego powodu nie zaleca się używania listy jako kolejki w przypadku większych projektów, natomiast w przypadku małych list jest to praktycznie niezauważalne. Ogromną zaletą kolejki priorytetowej ~~jest,~~ jest fakt, że może ona sortować i usuwać elementy z kolejki na podstawie priorytetów, zamiast usuwać najstarszy element. Daje to możliwość sortowania zadań ze względu na większy priorytet wykonania.

3. Implementacja kolejki priorytetowej oraz istotne metody

Gdy już zdecydujemy się na użycie kolejki priorytetowej w naszym kodzie, trzeba ją jakoś zaimplementować. Musimy pamiętać, że na początek musimy zadeklarować pustą listę, do której możemy stawiać elementy za pomocą metody append() klasy list. Lista jest potem sortowana w porządku rosnącym. Elementy możemy zdejmować przy użyciu metody pop(). Pobieramy i usuwamy wówczas ostatni dodany element listy (szczyt listy). Bardzo przydatny jest maksymalny priorytet kolejki - po zastosowaniu lista zostanie ułożona w kolejności malejącej, według priorytetów. Pętla while służy wówczas do pobierania elementów za pomocą metody pop(0).

Python zapewnia również wbudowaną implementację kolejki priorytetowej. Importujemy moduł kolejki - queue.PriorityQueue, a elementy wstawiamy wówczas za pomocą metody put(). Do usuwania elementów stosuje się metodę get(). Warto wspomnieć również o złożoności obliczeniowej kolejki priorytetowej w pythonie, która wynosi O(logn). Biblioteka queue jest sporym ułatwieniem, natomiast implementacja metody kolejki priorytetowej wygląda dokładnie tak, jak przedstawiono wyżej.

Python posiada również moduł do obsługi kopców - moduł heapq, który również posłużyć do zaimplementowania kolejki priorytetowej. Importujemy heapq z biblioteki, a potem tworzymy pustą listę – kopiec, gdzie dodajemy kolejne elementy, będące krotką w postaci – (priorytet, wartość). Zastosowanie kopców, jako kolejki priorytetowej, ogranicza nas do pobierania elementów tylko z najwyższym priorytetem.

4. Korzyści zastosowania kolejki priorytetowej

a) Utrzymywanie danych jako posortowaną listę względem priorytetów. Wtedy element o największym priorytecie jest pierwszy lub ostatni, zależy to od użytego sortowania. Operacja wstawiania elementu jest bardzo kosztowna, zmiana priorytetu wymaga ponownego sortowania, ominąć to można poprzez wyjęcie elementu, zmianie jego priorytetu i ponowne wstawienie do kolejki.
b) Istnieje możliwość dodania elementów do kolejki bez sortowania, a dopiero przy usuwaniu elementu wybierać największego. Wówczas operacja usuwania elementu będzie kosztowna, ale bez problemu zmieniamy priorytety w kolejce.
c) W kolejce możemy utrzymywać częściowe uporządkowanie za pomocą sterty. Operacja wstawiania i usuwania wymaga wówczas przebudowy sterty.

5. Proponowane rozwiązanie problemu

a) W zmiennej zapisujemy tekst, który chcemy przesłać (w rozbudowanym programie byłyby to pliku do przesłania)

b) Sprawdzamy długość zmiennej i ustalamy z ilu znaków się składa, pozwoli nam to na równy podział wiadomości.

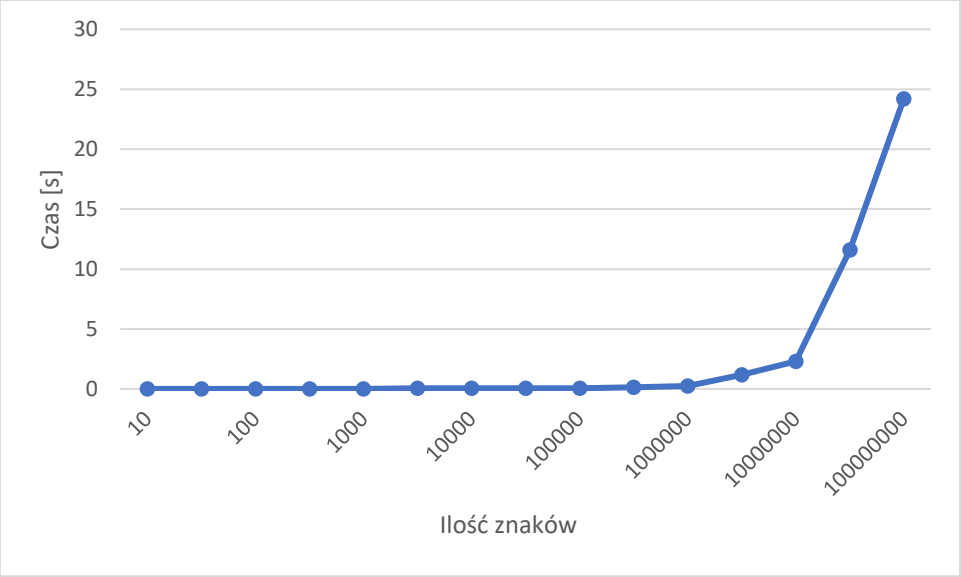
c) Dzielimy ciąg znaków na równe części (jeżeli w wyniku dzielenia zostaje reszta – zostaje ona dołączona do ostatniej części podzielonego tekstu).

d) Każdej części podzielonego pliku przypisujemy jego priorytet (dla początkowego elementu jest to indeks 0) , robimy to do momentu przejścia po wszystkich częściach

e) Dla zwizualizowania działania wyświetlamy 3 formy wiadomości:

- Przesłana wiadomość - ustawiona w kolejce priorytetowej
- Pomieszana lista (symulacja przesyłania pliku przez sieć)
- Poskładana wiadomość (wiadomość złożona z przesłanych części przez odbiorcę)

6. Czas wykonywania programu w zależności od wielkości - wizualizacja



Rysunek 1.Wykres zależności czasu wykonywania programu od ilości znaków

7. Złożoność obliczeniowa podstawowych operacji:

Operacja	**Złożoność obliczeniowa**
Wstawienie elementu	O(log n)
Dostęp do elementu minimalnego	O(1)
Usunięcie elementu minimalnego	O(log n)
Dostęp lub usunięcie dowolnego elementu	O(n)
Zmniejszenie priorytetu	O(log n)
Stworzenie kolejki składającej się z n elementów	O(n)
Stworzenie pustej kolejki	O(1)

Rysunek 1 Tabel złożoności obliczeniowej podstawowych operacji