1. Opis aktualnego stanu prac

W ramach realizacji projektu zaimplementowaliśmy funkcje kodera i dekodera zgodne z algorytmem LZSS.

- **Koder**:
  - Funkcja compress umożliwia kompresję danych tekstowych oraz obrazów.
  - W procesie kompresji zastosowaliśmy wyszukiwanie najdłuższych dopasowań w buforze wyszukiwania, a następnie zastępowaliśmy je tokenami w formacie <przesunięcie, długość>.
  - Użyliśmy stałych kontrolujących rozmiary bufora i długości dopasowań, co pozwoliło nam na elastyczne dostosowanie parametrów do specyfiki danych wejściowych.
- **Dekoder**:
  - Funkcja decompress została zaprojektowana tak, aby odtwarzać oryginalne dane na podstawie zakodowanych tokenów oraz surowych bajtów zapisanych przez koder.
  - Obecnie przetestowaliśmy dekoder tylko na danych tekstowych.
- **Testy wstępne**:
  - Przeprowadziliśmy testy kompresji i dekompresji dla krótkich ciągów znaków, które potwierdziły poprawność działania podstawowych funkcji.
  - Zauważyliśmy, że algorytm dobrze radzi sobie z plikami tekstowymi o powtarzalnej strukturze, ale wymaga dalszej optymalizacji dla dużych plików.
- **Użyte technologie**:
  - **Python**: wybraliśmy język programowania Python do implementacji algorytmu.
  - **Biblioteka bitarray**: wykorzystaliśmy ją do efektywnej manipulacji bitami, co pozwoliło na precyzyjną obsługę tokenów i danych wejściowych.
  - **Moduł argparse**: za jego pomocą stworzyliśmy interfejs linii komend, umożliwiający przekazywanie argumentów do skryptu.
  - **Moduł lzss.utils**: opracowaliśmy niestandardowe funkcje do odczytu i zapisu danych.
  - **Docker**: wdrożyliśmy Dockera, aby zarządzać środowiskiem uruchomieniowym i zapewnić spójność działania kodu na różnych maszynach.
    - Stworzyliśmy plik Dockerfile oparty na obrazie Pythona
    - W procesie budowania zainstalowaliśmy wszystkie zależności w izolowanym środowisku
  - **Venv**: zastosowaliśmy wirtualne środowisko Pythona do zarządzania zależnościami projektu.

2. Wnioski z dotychczasowych prac

Na obecnym etapie projektu osiągnęliśmy pierwsze cele, które pozwoliły nam na stworzenie i weryfikację podstawowej funkcjonalności algorytmu LZSS.

- **Co udało nam się osiągnąć:**

    - Opracowaliśmy i wdrożyliśmy podstawowe wersje kodera i dekodera zgodne z założeniami teoretycznymi LZSS.
    - Upewniliśmy się, że koder działa poprawnie dla plików tekstowych i obrazów, a dekoder jest w stanie odtworzyć dane wejściowe dla danych tekstowych.
    - Przeprowadziliśmy pierwsze testy, które potwierdziły poprawność działania algorytmu na krótkich ciągach znaków.
- **Obszary wymagające dalszej pracy:**
    - Optymalizacja wydajności dekodera: Dekoder wymaga usprawnienia, aby lepiej radził sobie z dużymi plikami oraz z bardziej złożonymi przypadkami, takimi jak obrazy czy dane binarne. Szczególną uwagę musimy poświęcić zarządzaniu buforem wyszukiwania oraz przyspieszeniu procesu odczytu danych zakodowanych w formacie <przesunięcie, długość>.
    - Rozszerzenie testów dekodera: Planujemy przeprowadzić bardziej zaawansowane testy dekodera, które obejmą różne typy danych, w tym pliki graficzne i binarne, co pozwoli nam ocenić jego uniwersalność i skuteczność.
    - Poprawa obsługi potencjalnych błędów: Chcemy usprawnić mechanizmy obsługi błędów w całym algorytmie, tak aby dekoder mógł skutecznie reagować na niezgodne formaty danych wejściowych oraz inne niespodziewane sytuacje.

3. Plan dalszych działań

W najbliższym czasie zamierzamy:

1. **Optymalizować algorytm kodera:**
    - Skupimy się na przyspieszeniu wyszukiwania dopasowań w buforze oraz na zmniejszeniu zużycia pamięci przez algorytm.
    - Przeprowadzimy analizy, aby wybrać najlepsze parametry dla rozmiaru bufora i maksymalnej długości dopasowania, w zależności od charakterystyki danych wejściowych.
2. **Rozbudować testy:**
    - Przygotujemy testy dla bardziej złożonych scenariuszy, takich jak obrazy, dane binarne oraz pliki o różnym stopniu powtarzalności.
    - Porównamy stopień kompresji i wydajność w różnych przypadkach testowych.
3. **Oceniać efektywność:**
    - Skoncentrujemy się na dwóch głównych metrykach: stopniu kompresji oraz przepływności bitowej (bitrate).
    - Przygotujemy raport z analiz, który pomoże nam zrozumieć, jak algorytm radzi sobie w różnych warunkach i jak można go dalej usprawniać.
4. **Dopracować dekoder:**
    - Zaktualizujemy dekoder, aby radził sobie z bardziej złożonymi przypadkami testowymi, w szczególności z obrazami i dużymi plikami.

6. Podsumowanie

Dotychczasowa realizacja projektu pozwoliła nam na stworzenie podstawowych funkcji algorytmu LZSS i przeprowadzenie wstępnych testów. Nasz koder działa dla plików

tekstowych i obrazów, a dekoder został przetestowany dla prostych plików tekstowych. W kolejnych etapach skupimy się na optymalizacji algorytmu, rozszerzeniu testów oraz analizie efektywności. Realizacja zaplanowanych działań pozwoli nam na ukończenie projektu zgodnie z założeniami i w pełni zweryfikuje działanie algorytmu w różnych przypadkach użycia.