Opis projektu

Zadanie 1

Podproblemy:
- Wyznaczenie granic krainy.
- Selekcja elitarnych tragarzy.
- Strategiczny rozkład dostarczania desek.
- Montaż kamer nadzorczych.

Wyznaczenie granic krainy
Opis:
Zadanie polega na wyznaczeniu granic krainy. Płaszczaki żyją w świecie dwuwymiarowym, więc wejściem będą punkty x, y. Odpowiednie punkty zostaną uwzględnione w wyjściu, czyli liście która będzie przedstawiać otoczkę wypukłą.

Wyjście:
- Punkty x, y.

Wyjście:
- Lista punktów x, y będąca otoczką wypukłą.

Użyty algorytm:
- Algorytm Grahama.

Selekcja elitarnych tragarzy
Opis:
Zadanie polega na dobraniu tragarzy w pary tak aby wspólnie mogli przenosić jak największa ilość desek. Tragarze zostają postawieni w dwóch rzędach, ci z rękoma z przodu postawieni na miejscu x = 1, a y dowolnym, natomiast ci z rękoma z tyłu postawieni zostają x=5 oraz y dowolnym. Ci tragarze, którzy mają ręce po przeciwnej stronie oraz lubią się, zostają połączeni krawędzią o przepływie 1. Z utworzonego grafu dwudzielnego tworzona jest sieć przepływu oraz algorytm Forda Fulkersona, który znajduje maksymalne skojarzenie w grafie dwudzielnym.

Wejście:
- Punkty x, y.
- Połączenia między punktami wraz z przepływem.

Wyjście:
- Liczba będąca maksymalnym skojarzeniem w grafie dwudzielnym.

Użyty algorytm:
- Algorytm Forda Fulkersona.

Strategiczny rozkład dostarczania desek
Opis:
Zadanie polega na zaplanowaniu w jaki sposób deski będą dostarczane z fabryki do punktów odbioru. Plan budowy to sieć przepływu z punktami x, y oraz połączeniami i przepływem między nimi. Fabryka to dowolny punkt nie znajdujący się w otoczce wypukłej jednocześnie będący startem sieci przepływu. Punkty odbioru to punkty z otoczki wypukłej. Ilość desek potrzebnych do wybudowania płotu to łączna długość otoczki wypukłej. Każdy punkt odbioru potrzebuje M desek. To, ile desek jest potrzebne oznacza długość między jednym punktem a drugim. Wynik (czyli maksymalny przepływ) daje informację tragarzom, czy potrzebny jest dalszy transport desek i ile ich trzeba donieść.

Wejście:
- Punkty x, y.
- Połączenia między punktami wraz z przepływem.

Wyjście:
- Liczba będąca maksymalnym przepływem w grafie.

Użyty algorytm:
- Algorytm Forda Fulkersona.

Montaż kamer nadzorczych
Opis:
Zadanie polega na wyznaczeniu miejsc do montażu kamer. Wejściem jest graf wraz z połączeniami. Wyjściem natomiast minimalne pokrycie wierzchołkowe. Do wyznaczenia miejsc kamer został zastosowany zachłanny algorytm minimalnego pokrycia wierzchołkowego. Algorytm wybiera zawsze wierzchołek który ma najwięcej połączeń.

Wejście:
- Punkty x, y.
- Połączenia między punktami.

Wyjście:
- Lista punktów będąca minimalnym pokryciem wierzchołkowym.

Użyty algorytm:
- Zachłanny algorytm minimalnego pokrycia wierzchołkowego.

Zadanie 2

Podproblemy:
- wyszukanie wzorców w tekście
- zamiana błędnych wyrazów na poprawne
- kompresja bezstratna poprawionego tekstu

Wejście:
- tekst z błędnie zapisanym słowem/słowami
- błędne i odpowiadające im poprawne słowa

Wyjście:
- naprawiony i skompresowany tekst

Zastosowane algorytmy:
- wyszukanie wzorców w tekście - algorytm Rabina-Karpa
- kompresja bezstratna poprawionego tekstu - algorytm kompresji Huffmana

Wyszukanie wzorców w tekście
Opis:
Klasa Rabin_Karp posiada metodę find_pattern(), która zwraca słownik {"wzorzec":[i1, i2, ...,in]}, gdzie in są to indeksy wystąpień wzorca w tekście

Wejście:
- tekst
- wzorzec do wyszukania

Wyjście:
- słownik indeksów wystąpień podanego wzorca

Zamiana błędnych wyrazów na poprawne
Opis:
Zakładam, że błędne wyrazy różnią się od poprawnych tylko znakami, a nie długością. W związku z tym odbywa się przekonwertowanie ciągu znaków z tekstem na listę i podmiana po indeksach.

Wejście:
- Tekst
- Poprawne słowo

Wyjście:
- Tekst z poprawionym słowem

Kompresja bezstratna poprawionego tekstu
Opis:
W klasie Huffman wywoływana jest metoda huffman(), która kolejno wywołuje metodę build_huffman_tree() odpowiedzialną za zliczenie wystąpień każdego unikalnego znaku w tekście i stworzenie kopca na podstawie słownika "frequency" - {"znak":ilość wystąpień}. Na końcu metoda zwraca "roota" do naszego kopca. Następnie huffman() wywołuje metodę huffman_coding() i jako argument przekazuje otrzymanego wcześniej "roota". huffman_coding() jest rekurencyjną metodą, która tworzy słownik "codes" - {"znak":"kod huffmana"}. Tworzy go za pomocą rekurencyjnego przeszukiwania wcześniej utworzonego drzewa Huffmana. Ostatnim etapem działania metody huffman() jest przeiterowanie po każdym znaku otrzymanego w konstruktorze klasy tekstu i dodawanie na jego podstawie odpowiadający mu kod huffmana (za pomocą wcześniej utworzonego słownika) do stringa "encoded_text". Dodatkowo klasa Huffman posiada metodę dehuffman() dzięki której możemy zdekompresować skompresowany wcześniej tekst i sprawdzić czy zgadza się w 100% z tym podanym w argumencie klasy.

Wejście:
- tekst

Wyjście:
- skompresowany ciąg składający się z "0" i "1"
- zdekompresowany tekst (opcjonalnie)

Jak wykonywane jest zadanie krok po kroku?
1. za pomocą klasy File_reader wczytujemy wejściowy tekst z błędnymi i poprawnymi wyrazami
2. File_reader tworzy nam słownik wordsToReplace - {"błędne słowo":"poprawne słowo"} (możemy mieć w tekście więcej niż 1 błędne słowo)
3. za pomocą klasy Rabin_Karp szukamy wszystkich błędnych słów iterując po wordsToReplace.keys()
4. po kolei podmieniamy wszystkie słowa za pomocą zwróconych indeksów przez Rabin_Karp.find_pattern()
5. kompresujemy naprawiony tekst za pomocą Huffman.huffman() i wypisujemy go na wyjściu

Zadanie 3

Podproblemy:
- Wyznaczenie płotu

- wyznaczenie strażnika z największą energią danego dnia
- patrol strażnika

Wygenerowanie płotu
Opis:
Generuje płot na podstawie liczby punktów oraz jasności punktów. Na początku sprawdzam czy liczba wartości jasności odpowiada liczbie punktów. Jeśli nie to drukujemy odpowiedni komunikat, że nie możemy stworzyć płotu.
Funkcja tworzy trasę przechodząc przez każdy punkt od 1 do ilości punktów. Dla każdego punktu tworzy krotkę zawierający numer punktu i odpowiadająca mu wartość jasności, a następnie dodaje ją do listy trasa.

Wejście:
- liczba_punktow: liczba punktów które są w płocie
- jasnosci: lista wartości reprezentujących jasności punktów w trasie

Wyjście:
- Zwraca listę ‘trasa’ która zawiera punkt trasy oraz ich odpowiadające jasności Każdy punkt jest przedstawiony jako krotka (numer punktu, jasność)

Znajdowanie strażnika
Opis:
Funkcja przeszukuje listę płaszczaków i zwraca tego który ma najwyższy poziom energii. Najpierw ustala, że maksymalny poziom energii wynosi 0 i nie ma jeszcze żadnego płaszczaka z tym poziomem. Następnie iteruje przez listę płaszczaków i dla każdego sprawdza czy jego poziom energii jest wyższy niż aktualnie najwyższy. Jeśli tak to aktualizuje maksymalny poziom energii i zapisuje i zwraca tego płaszczaka jako płaszczaka z największą energią.
Wejście:
- plaszczaki: lista obiektów która przechowuje obiekty klasy Płaszczak. Atrybuty - (id, energia)

Wyjście:
- Obiekt płaszczak z największą wartością atrybutu ‘energia’

Patrol
Opis:
Wyświetla informację o aktualnym dniu oraz aktualnej trasie patrolu strażnika danego dnia. Wywołuje w tej funkcji inne funkcje, aby znaleźć strażnika oraz wyznaczyć trasę patrolu.

Wejście:
- plaszczaki: lista obiektów która przechowuje obiekty klasy Płaszczak. Atrybuty(id, energia)
- zasieg: zasięg ruch strażnika. Jeśli zasięg jest równy 0, funkcja wyświetli komunikat o niemożności przemieszczania się strażnika i zakończy działanie

Wyjście:
- Funkcja nie zwraca żadnej wartości, jej działanie polega na wyświetlaniu kolejnych dni patrolowania, informacji o płaszczakach i trasie strażnika

Wyznaczanie trasy strażnika
Opis:
Dopóki nie zostaną odwiedzone wszystkie punkty na trasie wybierane są dostępne punkty na trasie, które znajdują się w zasięgu strażnika. Wybieramy punkt o mniejszej jasności od obecnego punktu w którym znajduje się strażnik, aby uniknąć odpoczynku strażnika. Gdy przechodzimy do punktu o większej jasności, dodajemy do trasy strażnika napis 'melodia'

który oznacza, że strażnik musiał odpocząć. Proces wybierania punktów kontynuuje się dopóki możliwe są dalsze ruchy strażnika.

Wejście:
- straznik: Obiekt klasy ‘Plaszczak’ reprezentujący strażnika (Płaszczak z największą energią)
- zasieg_straznika: zasięg ruchu strażnika, określający maksymalną odległość, którą może przebyć strażnik w jednym kroku

Wyjście:
- Lista zawierająca punkty trasy, którą ma pokonać strażnik