Teoretyczny opis rozwiązania problemów

Problem 1.

Interpretacja Problemu:

Po Krainie rozrzucone są punkty orientacyjne dla płaszczaków. Budujemy płot (otoczkę wypukłą), aby ochronić płaszczaków przed istotami z trzeciego wymiaru.

Między punktami znajdują się drogi o różnej grubości, którymi poruszają się płaszczaki, aby przetransportować kawałki płotu z fabryki (dowolny punkt) do punktu odbioru (inny dowolny punkt). Musimy znaleźć największy możliwy przepływ dla tych dróg, aby jak najszybciej dostarczyć kawałki płotu.

Płaszczaki pracują ze sobą tylko wtedy kiedy się lubią oraz mają odpowiednio ułożone ręce (jeden płaszczak z pary ma ręce z przodu, drugi musi mieć z tyłu).

Teoretyczny opis rozwiązania

Budowa płotu:

Do budowy używamy algorytmu Grahama. Proces rozpoczyna się od wyboru punktu początkowego, który ma najmniejszą współrzędną y (a w przypadku remisu, najmniejszą x). Ten punkt to punkt startowy. Następnie, wszystkie pozostałe punkty są sortowane według kąta, jaki tworzą z punktem początkowym względem osi x. Po sortowaniu inicjalizujemy stos i umieszczamy na nim punkt początkowy oraz dwa pierwsze punkty z posortowanej listy. Przechodzimy przez pozostałe punkty i dla każdego z nich sprawdzamy, czy dodanie go do stosu nie powoduje, że utworzona krawędź staje się wklęsła. Jeśli tak, usuwamy ostatni punkt ze stosu. Proces powtarzamy aż krawędź będzie wypukła, a następnie dodajemy nowy punkt do stosu. Po przetworzeniu wszystkich punktów stos zawiera wierzchołki otoczki wypukłej. Algorytm Grahama zapewnia, że utworzona otoczka jest najmniejszym wielokątem wypukłym, obejmującym wszystkie dane punkty.

Drogi różnej grubości:

Aby stworzyć drogi o różnej przepustowości, tworzymy symetryczną macierz sąsiedztwa. Macierz sąsiedztwa jest macierzą $n \times n$ (n jest ilością wszystkich punktów w Krainie), gdzie każdy element a_{ij} reprezentuje krawędź między wierzchołkiem i i

wierzchołkiem j. Do wyznaczenia maksymalnego przepływu używamy algorytmu Edmondsa-Karpa zaczynając od przepływu zerowego. Algorytm BFS przeszukuje sieć, szukając ścieżki, w której można zwiększyć przepływ, biorąc pod uwagę aktualne przepustowości podane w macierzy. Jeśli taka ścieżka zostanie znaleziona, metoda oblicza maksymalny możliwy przepływ przez tę ścieżkę (minimalną pozostałą przepustowość na ścieżce). Następnie aktualizuje przepływy wzdłuż tej ścieżki i odejmując go w przeciwnym kierunku (w sieci rezydualnej). Proces ten jest powtarzany aż do momentu, gdy nie można znaleźć żadnej nowej ścieżki powiększającej. Gdy BFS nie jest w stanie znaleźć nowej ścieżki, osiągnięty przepływ jest maksymalnym przepływem w sieci.

Praca płaszczaków:

Do określenia ilości par płaszczaków, również używamy algorytmu Edmondsa-Karpa. Na początku tworzymy graf dwudzielny, który reprezentuje ułożenie rąk płaszczaków, po lewej stronie znajdują się płaszczaki z rękami do przodu, a po prawej z tyłu. Następnie określamy relacje między płaszczakami (wpisujemy relacje, który płaszczak lubi którego). Łączymy ze sobą punkty, między którymi jest relacja. Po lewej stronie tworzymy źródło, do którego podpinamy wszystkie punkty z lewej strony, po prawej tworzymy ujście i podpinamy do niego wszystkie punkty z prawej strony, dzięki czemu mamy sieć. Każda krawędź ma przepustowość równą 1, a maksymalny przepływ w tym grafie odpowiada liczbie maksymalnej ilości par, jakie mogą powstać.

Problem 2.

Zapis melodii:

Interpretacja Problemu:

Problem polega na przetworzeniu ciągu tekstu w sposób odwracalny tak, aby zmniejszyć jego rozmiar (zredukować ilość bajtów wymaganych do jego przechowania) wykorzystując jedynie znaki 0 oraz 1. Zdaniem Informatyka miejsca brakuje niewiele, zatem kompresja rzędu kilkudziesięciu procent powinna być wystarczająca.

Rozwiązanie problemu:

Kompresja odbywa się przy pomocy algorytmu Huffmana. Najpierw program zamienia wszystkie wystąpienia znaku nowej linii na spacje. Następnie zlicza wystąpienia poszczególnych znaków w tekście i zapisuje je w słowniku oraz sortuje je w porządku malejącym. Ten słownik jest następnie wykorzystywany do utworzenia drzewa binarnego, które przypisuje kombinacje znaków 0 oraz 1 odpowiednim znakom z

tekstu wejściowego. Im “popularniejszy” jest znak, tym krótszy jego odpowiednik w zapisie bitowym, co czyni algorytm wydajniejszym niż ten, który zastosował Informatyk. Zakodowane dane są następnie zapisywane do pliku oraz drukowane na ekran (jeśli uruchomiono program z flagą -v). Dodatkowo, program zapisuje reprezentację drzewa binarnego w postaci pliku .json celem odszyfrowania pliku wejściowego.
Dekompresja odbywa się poprzez czytanie pliku wyjściowego i zamianę kombinacji binarnych na odpowiednie znaki alfabetu według słownika zawartego w pliku .json i wykorzystuje algorytm zachłanny.
Budowanie drzewa odbywa się w czasie O(nlogn), zaś kodowanie w czasie O(n).

Podmiana dźwięków:

Interpretacja Problemu:

Przed dokonaniem kompresji należy zamienić wszystkie wystąpienia słów “poli” na “boli”.

Rozwiązanie problemu:

Podmiana wystąpień “poli” na “boli” odbywa się poprzez wyszukanie wzorca algorytmem Knutha-Morrisa-Pratta, a następnie podmianę znaków na te docelowe. Złożoność czasowa to O(n + m).

Problem 3.

Zarządzanie Strażnikami:

Interpretacja Problemu:

Każdego dnia wybieramy strażnika o najwyższej energii z zadanego zakresu strażników numerowanych sekwencyjnie.

Rozwiązanie problemu:

Do zarządzania strażnikami użyto drzewa przedziałowego. Drzewo przedziałowe jest efektywną strukturą umożliwiającą szybkie zapytania o maksymalne wartości w zadanych przedziałach, a także pozwala na szybkie dynamiczne aktualizacje wartości.

Zapytania o strażnika z maksymalną energią w danym przedziale:

Algorytm poprawnie znajduje strażnika z maksymalną energią ponieważ używa drzewa przedziałowego, struktury danych poznanej na wykładzie.
Użyto drzewa do znajdowania strażnika o maksymalnej energii w danym przedziale, jest operacją O(logn).
Aktualizacje:
Aktualizowanie energii strażników jest operacją O(logn), co pozwala na dynamiczne zmiany w czasie rzeczywistym(na przykład po zmianie płaszczaka, można zaktualizować mu energię)

Minimalizacja ilości odsłuchań melodii na trasie:

Interpretacja problemu

Strażnik powinien patrolować trasę wokół płotu, zatrzymując się co pewną liczbę punktów orientacyjnych, by rozejrzeć się.
Energia strażnika jest utrzymywana tylko wtedy, gdy przemieszcza się od jaśniejszego punktu do ciemniejszego; w przeciwnym razie musi zatrzymać się i odsłuchać melodię, aby odzyskać energię.

Rozwiązanie problemu

Strażnik ma pewną(N > 0) ograniczoną liczbę punktów orientacyjnych, które może przebyć jednorazowo, zanim musi się zatrzymać.
Trzeba zaplanować punkty zatrzymania tak, aby minimalizować liczbę wymaganych odsłuchań melodii.
Do optymalizacji trasy strażnika zastosowano naiwny algorytm, który zawsze minimalizuje ilość odsłuchań melodii, ale nie minimalizuje jednocześnie całkowitej ilości przystanków.
Algorytm inkrementuje licznik punktów, przez które przeszedł strażnik.
Jeżeli strażnik przechodzi od jaśniejszego do ciemniejszego punktu, energia strażnika jest zachowana, bez odsłuchiwania melodii, więc algorytm zawsze nakazuje strażnikowi się zatrzymać, licznik punktów jest resetowany.
Gdy licznik osiągnie maksymalną liczbę punktów po których wymagane jest odłuchanie melodii, algorytm nakazuje strażnikowi się zatrzymać i odłuchać melodię, licznik punktów jest resetowany. Algorytm poprawnie minimalizuje ilość odłuchań melodii ponieważ, jeżeli strażnik odsłuchuję melodię, to oznacza, że zatrzymanie się w którymkolwiek z poprzednich N punktów zmuszało by go do odsłuchania melodii.

Złożność obliczniowa tego algorytmu to O(m) gdzie m jest liczbą punktów orientacyjnych na płocie.

Dokumentacja techniczna znajduje się w repozytorium