Spis treści

1 Struktury Danych

Wyróżniamy kilka rodzajów abstrakcyjnych struktur danych, które pozwalają nam na przechowywanie danych.

1.1 Rodzaje

Wyróżniamy klasycznie dwa rodzaje fizycznie przechowywania sekwencyjnie danych. Wszystkie inne struktury można sprowadzić lub zaimplementować przy pomocy tych dwóch.

1.1.1 Tablica

Ciąg elementów w ciągłym bloku pamięci o stałym rozmiarze. Po ang. jest to array i w językach niskopoziomowych jest to po prostu surowy blok pamięci o rozmiarze n bajtów. Operacja powiększania tablicy jest niezwykle kosztowna, albowiem wymaga stworzenia nowej tablicy, i przekopiowania wszystkich elementów.

1.1.2 Lista

Abstrakcyjny zbiór elementów, gdzie każdy element zna tylko swojego następnika. W zależności od wariantu, element może też znać głowę listy (arbitralnie wybrany pierwszy element) lub/i swojego poprzednika. Bardzo dobra struktura jeśli tylko jest realizowany sekwencyjny dostęp.

1.2 Interfejsy

Wyróżniamy kilka rodzajów interakcji z strukturami danych.

1.2.1 Stos

Zmienna kolekcja elementów, modyfikowana poprzez dodawanie lub usuwanie elementów z jednego końca. W związku z tym dostęp do elementów jest realizowany w stylu LIFO.

1.2.2 Kolejka

Zmienna kolekcja elementów, modyfikowana przez dodawanie elementów z jednego końca i usuwanie z drugiego. W związku z tym dostęp do elementów jest realizowany w stylu FIFO.

1.2.3 Sterta

Po ang. heap, to sposób realizacji drzewa binarnego na (tradycyjnie) tablicy. Dla wierzchołka w pozycji i lewy element znajduje się na $i * 2$ pozycji, a prawy na $i * 2 + 1$ pozycji.

2 Przeszukiwanie grafu

- Dany jest graf
- Rozpoczynamy od pewnego wierzchołka. Dodajemy go do listy niezbadanych wierzchołków.
- Następnie tak długo jak są niezbadane wierzchołki, usuwamy z listy niezbadanych wierzchołków wierzchołek i dodajemy jego sąsiadów do listy.

2.1 BFS

Przeszukujemy graf sekwencyjnie. Najpierw przeszukujemy te odległe o $k \geq 0$ od źródła, potem te $k + 1$, itd. Na liście ten algorytm jest realizowany przez traktowanie listy jako kolejki. Słabo sobie radzi z

2.2 DFS

Nie przeszukujemy grafu w konkretnej kolejności. Po prostu przeszukujemy graf jak popadnie. Realizowany jest z reguły rekurencyjnie, lub na liście poprzez traktowanie jej jako stos.

2.3 Problem najkrótszych ścieżek

Dla grafu o ważonych krawędziach, można określić najkrótszą (najmniej kosztowną) ścieżkę pomiędzy dwoma wierzchołkami. Znalezienie takiej ścieżki to problem najkrótszych ścieżek.

2.3.1 Algorytm Djikstry

Jest to algorytm na rozwiązywanie problemu najkrótszych ścieżek. Sprowadza się on do DFS lub BFS, z osobną strukturą danych na przechowywanie "odległości"od startu dla każdego wierzchołka. W momencie iteracji po wierzchołkach, sąsiednim wierzchołkom jest przypisywana suma odległości aktualnie odwiedzanego wierzchołka i wagi połączenia do sąsiedniego wierzchołka, jeśli odległość do sąsiada jest większa od tej sumy.

2.3.2 A*

Jest to ulepszona wersja algorytmu djikstry, która wykorzystuje jakiejś heurystyki, do indeksowania kolejki priorytetowej. W ten sposób najlepiej rokujące wierzchołki są odwiedzane najpierw.

3 Algorytmy zachłanne

Algorytm, co w danym oknie kontekstowym, wykonuje lokalnie optymalne działanie to algorytm zachłanny. Np.: problem plecakowy lub wydawania reszty klasycznie rozwiązuje się zachłannie

3.1 Problem wydawania reszty

Mając zbiór nominałów, stwórz kolekcję instancji nominałów w taki sposób, aby suma ich byłą zgodna z wejściem. Niektóre wersje problemu dodają wymóg, jak najmniejszej ilości monet (instancji). Klasycznym rozwiązaniem jest dodawanie do kolekcji największego nominału, mniejszego od sumy dotychczas dodanych nominałów.

3.2 Problem MST

Celem jest znalezienie pod-grafu, o jak najmniejszej sumie wag krawędzi. Klasycznym rozwiązaniem jest posortowanie krawędzi zgodnie z ich wagą, a potem dobieranie krawędzi o najmniejszych wagach.

4 Programowanie dynamiczne

Opiera się na dzieleniu problemu na pod-problemy. Z reguły celem jest stworzenie optymalnej pod struktury, w której rozwiązywanie problemu jest o wiele prostsze lub nawet trywialne.

5 Algorytmy wyliczeniowe

Są to algorytmy, które generują cały zbiór możliwych rozwiązań. Jeśli taki algorytm wykonuje ten cel sprawdzając wszystkie kombinacje wejść jest to algorytm siłowy (pełnego przeglądu).

6 Algorytmy sortowania

6.1 Bubble-sort

Dla każdego elementu przesuwaj ten element w prawo aż napotkasz albo koniec tabeli, albo większy element. W pewnym sensie w bubble sort elementy bąblują na koniec tabeli. $O(n^2)$

6.2 Insert sort

Dla każdego elementu, przesuwaj w lewo tak długo jak jest ten element mniejszy od poprzedniego. $O(n^2)$, ale w praktyce lepszy od bubble-sort

6.3 Quick sort

Dla tabeli w danym zakresie, przesuń wszystkie elementy mniejsze od środkowego na lewo od środkowego, a wszystkie większe na prawo od środkowego. Proces powtórz na dwóch pod-tabelach na lewo i prawo.