Algorytmy i struktury danych
Laboratorium 5

Termin oddania: 3 czerwca, 10:34

Zadanie 1. [10%]
Napisz program, który symuluje działanie kolejki priorytetowej. Struktura powinna umożliwiać przechowywanie wartości typu `int` wraz z ich priorytetem, będącym nieujemną liczbą całkowitą, przy czym najwyższym priorytetem jest wartość 0. Program nie powinien wymagać żadnych parametrów uruchomienia, a na standardowym wejściu przyjmuje dodatnią liczbę całkowitą M (liczba operacji), a następnie (w liniach 2–$(M+1)$) jedną z operacji:

- insert x p - wstaw do struktury wartość x o priorytecie p
- empty - wypisz wartość 1 dla pustej struktury, wartość 0 w p.p.
- top - wypisz wartość o najwyższym priorytecie lub pustą linię w przypadku braku elementów w strukturze
- pop - wypisz wartość o najwyższym priorytecie a następnie usuń ją ze struktury (wypisz pustą linię w przypadku braku elementów w strukturze)
- priority x p - dla każdego elementu o wartości x obecnego w strukturze ustawia priorytet p jeśli jest on wyższy od aktualnego priorytetu danego elementu
- print - wypisuje w jednej linii zawartość struktury w postaci (x_i, p_i), gdzie x_i to kolejne wartości przechowywane w kolejce a p_i odpowiadające im priorytety.

Dla liczby elementów w kolejce n, koszt operacji musi wynosić $O(\log n)$, dla wszystkich poleceń z pominięciem print.

Zadanie 2. [30%]
Korzystając ze struktury zaimplementowanej w **Zadaniu 1.** lub kopca Fibonacciego, zaimplementuj program realizujący algorytm Djikstry, dla podanego grafu skierowanego $G = (V, E)$, znajdujący najkrótsze ścieżki z wybranego wierzchołka $v \in V$ do każdego $\tilde{v} \in V$.
Program nie powinien wymagać żadnych parametrów uruchomienia. Po uruchomieniu programu, na standardowym wejściu, podajemy definicję grafu G oraz wierzchołek startowy v. Kolejno wczytywane są:

- liczba wierzchołków $n = |V|$ (przyjmujmy, że wierzchołki są etykietowane kolejnymi liczbami naturalnymi $\{1, \ldots, n\}$)
- liczba krawędzi $m = |E|$ (krawędzie są postaci (u, v, w), gdzie u to źródło krawędzi, v jest wierzchołkiem docelowym a w wagą krawędzi – zakładamy, że wagi są nieujemnymi liczbami rzeczywistymi, ale niekoniecznie spełniona jest nierówność trójkąta, ponadto przyjmujemy iż ścieżka z u do u zawsze istnieje i ma koszt 0)
- kolejno m definicji krawędzi w postaci `u v w`
- etykieta wierzchołka startowego.

Na standardowym wyjściu powinno zostać n linii, w formacie `id_celu waga_drogi`, natomiast na standardowym wyjściu błędów powinny być wypisane dokładne ścieżki (tzn. wierzchołki pośrednie i wagi) do każdego z wierzchołków docelowych oraz czas działania programu w milisekundach.

Zadanie 3. [40%]
Korzystając ze struktury z **Zadania 1.**, zaimplementuj algorytmy znajdujące dla podanego grafu nieskierowanego $G = (V, E)$ minimalne drzewa rozpinające.

Program powinien umożliwiać wykonanie algorytmu Prima (parametr uruchomienia `-p`) oraz algorytmu Kruskala (parametr uruchomienia `-k`). Niezależnie od parametru uruchomienia, dane wejściowe przyjmują postać:

- liczba wierzchołków $n = |V|$ (przyjmujmy, że wierzchołki są etykietowane kolejnymi liczbami naturalnymi $\{1, \ldots, n\}$)
- liczba krawędzi $m = |E|$ (krawędzie są postaci (u, v, w), gdzie $u \in V$ i $v \in V$ są połączonymi wierzchołkami a w wagą krawędzi – zakładamy, że wagi są nieujemnymi liczbami rzeczywistymi, ale niekoniecznie spełniona jest nierówność trójkąta, naturalnie krawędź z u do u zawsze istnieje i ma koszt 0 oraz $\forall_{u,\,v}\ (u, v, w) \iff (v, u, w)$, ponadto przyjmujemy, że graf jest spójny)
- kolejno m definicji krawędzi w postaci `u v w`

Na standardowym wyjściu powinny zostać wypisane, w kolejnych liniach, krawędzie `u v w` użyte w drzewie rozpinającym (przyjmujemy, że $u < v$) oraz łączną wagę drzewa rozpinającego.

Zadanie 4. [20%]
Załóżmy, że mamy pełny graf nieskierowany $G = (V, E)$, gdzie V oznacza zbiór wierzchołków i $|V| = n$, a $E = \{(v, u, w) : v \in V \wedge u \in V \wedge w \in \mathbf{R}^+\}$ oraz $|E| = m = \frac{n(n-1)}{2}$. Zakładamy, że wagi krawędzi spełniają nierówność trójkąta. Zaimplementuj algorytmy, w których startując z dowolnego wierzchołka, trawersujemy krawędzie do czasu odwiedzenia wszystkich wierzchołków, zgodnie z następującymi strategiami:

- błądzenia losowego, przed wykonaniem każdego kroku, losujemy z prawdopodobieństwem jednostajnym krawędź, którą pokonujemy
- będąc w wierzchołku v wybieramy krawędź o najniższej wadze prowadzącą do wciąż nieodwiedzonego wierzchołka
- wykorzystując **Zadanie 3.** budujemy minimalne drzewo rozpinające, a następnie szukamy 'cyklu Eulera', przechodząc każdą z krawędzi w drzewie dwukrotnie i podążamy ścieżką usuwając ponowne wystąpienia wierzchołków. Sprawdź czy drzewa zbudowane przy użyciu alg. Prima dają zawsze te same wyniki co zbudowane przy użyciu alg. Kruskala.

Na standardowym wejściu podawane są kolejno:

- liczba wierzchołków $n = |V|$ (przyjmujmy, że wierzchołki są etykietowane kolejnymi liczbami naturalnymi $\{1, \ldots, n\}$)
- kolejno m definicji krawędzi w postaci `u v w`

Wybór wierzchołka startowego powinien być dokonywany przez algorytm.
Na standardowym wyjściu powinny zostać wypisane trzy linie (kolejno dla strategii błądzenia losowego, wyboru krawędzi o najniższej wadze, drzewa rozpinającego) postaci: `k W M t`, gdzie `k` oznacza liczbę wykonanych kroków podczas odwiedzania wierzchołków, `W` łączny koszt trasy, `M` zużytą dodatkową pamięć (np. na pamiętanie odwiedzonych już wierzchołków), a `t` czas działania algorytmu (wraz z preprocessingiem, np. budowaniem drzewa). Na standardowym wyjściu błędów powinny być drukowane pełne trasy wykonane dla każdej ze strategii. Wykonaj testy dla $n \in \{5, 50, 500, \ldots, 500000\}$.

Zadanie 5. [20% (dodatkowe)]
Uzupełnij **Zadanie 4.** o strategię bazującą na programowaniu dynamicznym, wyszukując ścieżek dla pod problemów o rozmiarze $2 \leq i \leq n$. Znajdź zależność rekurencyjną oraz oszacuj asymptotyczną złożoność czasową takiego rozwiązania. Wykonaj testy dla wskazanych w **Zadaniu 4.** n i wyciągnij wnioski na temat liczby potrzebnych kroków, łącznego czasu oraz zużycia pamięciowego poszczególnych strategii.