JĘZYKI SKRYPTOWE I ICH ZASTOSOWANIA

ZADANIE 2: GRAFY PROSTE

1. Zadanie

Wybierz jedną ze struktur danych reprezentujących małe grafy proste oraz jeden zestaw operacji wykonywanych na tychże grafach i zaimplementuj całość w `C`/`C++` w taki sposób, by była widoczna w `Pythonie` jako klasa wewnątrz modułu `simple_graphs`.

2. Wybór struktury i zestawu operacji

Na potrzeby tego zadania przyjmujemy, że małe grafy proste mają zbiór wierzchołków będący podzbiorem zbioru $\{0, 1, \ldots, 15\}$. Grafy tego typu da się zareprezentować w pamięci komputera przy pomocy struktur danych, które:

(1) przechowują zbiór wierzchołków w postaci 16-bitowej liczby, w której i-ty bit zawiera informację o tym, czy i jest wierzchołkiem grafu;
(2) przechowują relację sąsiedztwa wierzchołków w jednej z poniższych postaci:
   (a) struktury `AdjacencyMatrix`, czyli 16-elementowej tablicy 16-bitowych liczb, której i-ty element przechowuje na j-tym bicie informację o tym, czy wierzchołek i sąsiaduje z wierzchołkiem j;
   (b) struktury `AdjacencyList`, czyli 16-elementowej tablicy, której i-ty element to posortowana rosnąco lista sąsiadów wierzchołka i;
   (c) struktury `EdgesList`, czyli posortowanej leksykograficznie listy krawędzi grafu, w której krawędź łącząca wierzchołek i z wierzchołkiem j reprezentowana jest jako para $(\min\{i, j\}, \max\{i, j\})$;
   (d) struktury `IncidenceMatrix`, czyli listy 16-bitowych liczb, której i-ty element przechowuje na j-tym bicie informację o tym, czy wierzchołek j jest incydentny z i-tą krawędzią.

Wybierz jedną z powyższych struktur i jedną z operacji opisanych w pliku `graphs.py` jako dodatkowe, po czym zgłoś wybraną strukturę i operację prowadzącemu zajęcia w celu rezerwacji.

3. Implementacja

Po potwierdzeniu rezerwacji zaimplementuj wybraną strukturę wraz z zestawem operacji, na który składa się wybrana operacja dodatkowa i wszystkie operacje opisane w pliku `graphs.py` jako podstawowe. Implementację zrealizuj w `C`/`C++` w taki sposób, by zawierała się w jednym pliku i by po kompilacji oraz instalacji była widoczna w `Pythonie` jako klasa wewnątrz modułu `simple_graphs`. Klasa ta powinna:

(1) mieć nazwę podaną przy wybranej do implementacji strukturze;
(2) przechodzić test polegający na wykonaniu polecenia `./test.py -t ST`, gdzie `ST` to wybrana struktura;
(3) przechodzić test polegający na wykonaniu polecenia `./test.py -t ST OP`, gdzie `ST` to wybrana struktura, a `OP` wybrana operacja dodatkowa.

Oba powyższe testy zostaną uruchomione w środowisku, na które składa się Gentoo Linux, kompilator `gcc`/`g++` 13.2 i `Python` w wersji 3.12. Łączny czas trwania obu testów nie może przekraczać 24 godzin, a zużyta pamięć 4GiB. Opis procesu łączenia `C`/`C++` z `Pythonem` można znaleźć w oficjalnej dokumentacji `Pythona`, na stronie

`https://docs.python.org/3/extending/extending.html`

Opis formatu `g6`, niezbędnego do poprawnej realizacji zadania, można znaleźć na stronie

`https://users.cecs.anu.edu.au/~bdm/data/formats.txt`

4. Ocena

Aby uzyskać ocenę pozytywną, należy oddać do sprawdzenia kod źródłowy. Efektem tego sprawdzenia jest ocena:

(1) celująca, jeśli kod spełnia wszystkie powyższe wymagania i jest najszybszy wśród wszystkich implementacji tej samej struktury;
(2) bardzo dobra, jeśli kod spełnia wszystkie wymagania, ale nie jest najszybszy;
(3) niedostateczna, w pozostałych przypadkach.

Podstawą do oceny szybkości jest informacja o czasie trwania testu raportowana przez pierwszy z wykonywanych testów, tj. `./test.py -t ST`.