Algorytmy i Złożoność Obliczeniowa

Projekt 2 – (29 kwietnia 2025)

mgr inż. Damian Mroziński

Zakres projektu
C/C++, algorytmy grafowe, analiza wyników, praca na plikach, skryptowanie

Jakie są cele projektu?

- Zapoznanie z różnymi algorytmami grafowymi, ich implementacja oraz analiza efektywności.
- Nabycie umiejętności dopasowania programu do wymagań / istniejących klas.
- Nauczenie się podstawowego skryptowania.
- Mądre planowanie projektu, aby nie komplikować sobie życia.
- Korzystanie z gotowych datasetów oraz ich szukanie.

Problemy do rozwiązania:

1. Wyznaczanie minimalnego drzewa rozpinającego (MST) - algorytm Prima oraz algorytm Kruskala.
2. Wyznaczanie najkrótszej ścieżki w grafie - algorytm Dijkstry oraz algorytm Forda-Bellmana.
3. Wyznaczanie maksymalnego przepływu - algorytm Forda-Fulkersona (na 5.5).

Sposoby reprezentacji grafu:

- reprezentacja macierzowa (macierz incydencji),
- reprezentacja listowa (lista następników/poprzedników).

Obowiązują następujące założenia

1. Wszystkie struktury powinny być alokowane i zwalniane dynamicznie (zgodnie z rozmiarem instancji).
2. Struktury należy zaimplementować samodzielnie. Nie wolno korzystać z gotowych rozwiązań.
3. Domyślnie waga/koszt/przepustowość jest liczbą naturalną.
4. Program musi kompilować się **bez ostrzeżeń kompilatora** (kompilatora, nie IDE).
5. Program musi przejść test debuggera, tj. nie mieć wycieków pamięci i innych błędów.

6. Należy zmierzyć czas pracy algorytmu (w milisekundach). Do pomiaru czasu nie wlicza się ładowanie danych z pliku, generowanie grafu, czy zapis wyników do pliku.

7. Kod musi być sformatowany spójnie i zawierać komentarze.

8. Do pomiaru czasu należy zaprojektować klasę `Timer`, zgodnie z podanym plikiem nagłówkowym. Publiczna część musi być dokładnie taka sama, nie większa i nie mniejsza.

```
class Timer
{
  public:
  Timer();       // Initialize and prepare to start.
  void reset();  // Reset timer.
  int start();   // Start timer.
  int stop();    // Stop timer.
  int result();  // Return elapsed time [ms].

  private:
  // Everything else you need, both fields and methods.
}
```

Zachowując zadany nagłówek, można łatwo podmienić plik klasy, np. zastąpić implementację pomiaru dla Windowsa wersją dla Linuxa i ponownie skompilować program. Jeśli publiczna część nagłówka pozostaje bez zmian, wszystko zadziała poprawnie.

9. Wygenerowane grafy muszą być grafami spójnymi. Można najpierw wygenerować drzewo rozpinające, a następnie losować kolejne krawędzie aż do uzyskania wymaganej gęstości.

10. Plik z danymi wejściowymi zawsze jest zbudowany w ten sam sposób:

    (a) W pierwszej linii znajduje się liczba krawędzi oraz liczba wierzchołków (rozdzielone tabulatorem, w tej kolejności).

    (b) Wierzchołki numerowane są w sposób ciągły, od zera.

    (c) W liniach 2-... znajduje się opis krawędzi (każda w osobnej linii). Wartości są rozdzielone tabulatorem, i są to: wierzchołek początkowy, wierzchołek końcowy oraz waga/przepustowość.

    (d) Dla problemu MST pojedyncza krawędź jest krawędzią nieskierowaną, natomiast dla algorytmów najkrótszej drogi i maksymalnego przepływu jest krawędzią skierowaną.

11. Program musi móc pokazać graf (w obu formach), a także rozwiązanie problemów, czyli na przykład najkrótszą ścieżkę razem z jej kosztem.

12. Należy przeprowadzić weryfikację poprawności na przykład wczytanych wartości. Czy liczba krawędzi w stworzonej i załadowanej strukturze zgadza się z danymi wejściowymi?

13. **BADANIA** Dla obu reprezentacji grafu należy dokonać pomiaru czasu działania algorytmów w zależności od rozmiaru grafu oraz jego gęstości (liczba krawędzi w stosunku do liczby krawędzi dla grafu pełnego). Należy zbadać minimum siedem różnych reprezentatywnych liczb wierzchołków oraz gęstości 25,50 oraz 99%.

14. Pojedynczy pomiar jest niemiarodajny, zwłaszcza ze względu na możliwy rozrzut wyników w grafach. Aby badania zostały wykonane prawidłowo, należy wygenerować dla danej pary wierzchołki+gęstość np 50 losowych instancji, a wyniki w sprawozdaniu uśrednić.

15. Należy zachować spójność badań, aby można było sensownie porównywać wyniki.

16. Sterowanie programem odbywa się za pomocą argumentów do metody głównej. Należy wprowadzić dwa tryby działania (no, technicznie trzy). Są to: tryb pojedynczego testu, gdzie danymi wejściowymi jest plik, a także tryb badań, gdzie grafy są losowane. Można użyć tego helpa jako bazy i dostosować go pod siebie. Strona pomocy musi ściśle odpowiadać temu, czego program będzie od nas oczekiwał. **Dla problemu najkrótszej ścieżki należy móc podać wierzchołek początkowy i końcowy.**

```
FILE TEST MODE:
    Usage:
      ./YourProject --file <problem> <algorithm> <inputFile> [outputFile]

    <problem>     Problem to solve (e.g. 0 - MST, 1 - shortest path)
    <algorithm>   Algorithm for the problem
                    For MST (e.g. 0 - all, 1 - Prim's, ...)
                    For shortest (e.g. 0 - all, 1 - Dijkstra, ...)
    <inputFile>   Input file containing the graf.
    [outputFile]  If provided, solved problem will be stored there.

BENCHMARK MODE:
    Usage:
      ./YourProject --test <problem> <algorithm> <size> <density> <count>
                           <outputFile>

    <problem>     Problem to solve (e.g. 0 - MST, 1 - shortest path)
    <algorithm>   Algorithm for the problem
                    For MST (e.g. 0 - all, 1 - Prim's, ...)
                    For shortest (e.g. 0 - all, 1 - Dijkstra, ...)
    <size>        Number of nodes.
    <density>     Density of edges.
    <count>       How many times test should be repeated (with graph regen).
    <outputFile>  File where the benchmark results should be saved
                  (every measured time is stored in seperate line).
```

```
HELP MODE:
    Usage:
      ./YourProject --help

    Displays this help message.

    Notes:
    - The help message will also appear if no arguments are provided.
    - Ensure that either --file or --test mode is specified;
      they are mutually exclusive.
```

Ocenianie - wymagane algorytmy i dodatkowe zadania

3.0 Zaimplementować po jednym algorytmie dla problemów 1,2 i przeprowadzić badania. Obiektowość nie jest wymagana. Obie reprezentacje.

4.0 Zaimplementować po dwa algorytmy dla problemów 1,2 i przeprowadzić badania. Obiektowość jest wymagana. Obie reprezentacje.

5.0 Jak na 4.0. Dodatkowo należy przeprowadzić badania na dużym wybranym datasecie znalezionym w Internecie i porównać na nim swoje implementacje z dowolnym gotowym rozwiązaniem/narzędziem/implementacją. Do datasetów proponuje zerknąć na, chociażby https://snap.stanford.edu/data/index.html. W sytuacji gdy dataset nie będzie nam pasował pod jakimś względem (nie mówię o konieczności dodania na przykład liczby węzłów do pierwszej linii, to można zrobić zawsze, bo dopasowujemy plik wejściowy do naszych potrzeb) i naprawdę nie możemy znaleźć innego, to można go zmodyfikować, opisując te zmiany w sprawozdaniu.

5.5 Jak na 5.0. Dodatkowo należy rozwiązać problem 3 z jednym algorytmem i przeprowadzić badania.

JEŚLI KTOŚ MA INNY POMYSŁ, może chciałby sprawdzić inny algorytm czy inne zadanie. Przetestować coś inaczej, skupić się bardziej na datasetach i porównaniu z większą liczbą gotowych rozwiązań, może przeprowadzić bardziej szczegółowe badania zamiast czegoś innego. Za zgodą prowadzącego można coś zmienić.

W razie pytań, pozostaję do dyspozycji.