ДИСКРЕТНА МАТЕМАТИКА
Завдання на проект

Мета роботи

Поглибити та вдосконалити знання з теорії графів, задач на графах, набути навички реалізації алгоритмів, оцінювання ефективності їх роботи.

Загальні положення

Проекти виконуються індивідуально або бригадами по дві особи. Розподіл роботи між членами однієї бригади залишається на ваш розсуд.

Виконання проекту не є обов'язковим (але ми рекомендуємо :))

Проект присвячено аналізу деякого алгоритму з теорії графів, який обирається з переліку варіантів за узгодженням із викладачем. У кожній групі кожен варіант може виконувати не більше двох бригад.

Виконання проекту включає такі етапи:

1) реалізувати структуру даних «граф», засоби його визначення та візуалізації, необхідні базові операції; необхідно передбачити можливість представлення графів як матрицею суміжності, так і списками суміжності, а також перехід від одного представлення до іншого за потреби;

2) реалізувати випадкове генерування графів заданого розміру та заданої щільності;

3) реалізувати алгоритм згідно обраного варіанту;

4) провести обчислювальні експерименти та оцінити час виконання алгоритму для графів різного розміру;

5) проаналізувати одержані дані та зробити висновки щодо ефективності алгоритму, який розглядається.

Варіанти завдань

У всіх варіантах розглядаються тільки прості графи без петель.

1. Побудова матриці досяжності графу за допомогою DFS та BFS.
   - https://en.wikipedia.org/wiki/Depth-first_search
   - https://en.wikipedia.org/wiki/Breadth-first_search
2. Алгоритм Уоршелла (побудова матриці досяжності).
   - https://en.wikipedia.org/wiki/Transitive_closure
   - https://youtu.be/sUTueV0Pu6E?si=oCwn7m9v4Th_D8YP&t=1287
3. Алгоритм Беллмана-Форда (пошук шляхів мінімальної ваги).
   - https://en.wikipedia.org/wiki/Bellman–Ford_algorithm
   - https://youtu.be/hWNiSa7Jntg?si=CfJcTWSe6Mpar8cS
4. Алгоритм Флойда-Уоршелла (пошук шляхів мінімальної ваги).
   - https://en.wikipedia.org/wiki/Floyd–Warshall_algorithm
   - https://youtu.be/hWNiSa7Jntg?si=4IUIYmNInO3aVKNs&t=1015
5. Алгоритм Джонсона (пошук шляхів мінімальної ваги).
   - https://en.wikipedia.org/wiki/Johnson's_algorithm
6. Алгоритм топологічного сортування на основі DFS.
   - https://en.wikipedia.org/wiki/Topological_sorting
7. Алгоритм Кана для топологічного сортування
   - https://en.wikipedia.org/wiki/Topological_sorting
8. Пошук компонент сильної зв'язності (алгоритм Корасайю/Таржана).
   - https://en.wikipedia.org/wiki/Tarjan's_strongly_connected_components_algorithm
9. Алгоритм Крускала (пошук кістякових дерев мінімальної ваги).
   - https://en.wikipedia.org/wiki/Kruskal's_algorithm
10. Алгоритм Борувки (пошук кістякових дерев мінімальної ваги).
    - https://en.wikipedia.org/wiki/Borůvka's_algorithm

11. Алгоритм Брона-Кербоша (пошук максимальних клік графу)
    - https://en.wikipedia.org/wiki/Bron–Kerbosch_algorithm
12. Алгоритм MaxCliqueDyn (пошук максимальних клік графу)
    - https://en.wikipedia.org/wiki/MaxCliqueDyn_algorithm
13. Алгоритм Форда-Фалкерсона або будь-який інший алгоритм пошуку мінімального потоку.
    - https://en.wikipedia.org/wiki/Flow_network
    - https://en.wikipedia.org/wiki/Ford–Fulkerson_algorithm
14. Будь-який алгоритм перевірки графу на планарність.
    - https://en.wikipedia.org/wiki/Planarity_testing
15. Будь-який алгоритм розв'язання задачі про вершинне покриття
    - https://en.wikipedia.org/wiki/Vertex_cover
16. Будь-який алгоритм розв'язання задачі про пошук гамільтонового циклу
    - https://en.wikipedia.org/wiki/Hamiltonian_path_problem
17. Будь-який алгоритм розв'язання задачі комівояжера
    - https://en.wikipedia.org/wiki/Travelling_salesman_problem

Також, за умови узгодження із лектором, ви можете обрати якийсь інший графовий алгоритм, який вам цікавий, для дослідження та аналізу.

Методичні вказівки до виконання

1. У всіх варіантах розглядаються тільки прості графи без петель. Однак різні алгоритми призначені для роботи над різними класами графів:
   – у варіантах 11, 12, 14, 15, 16 розглядаються тільки неорієнтовані графи;
   – у варіантах 6, 7, 8 розглядаються тільки орієнтовані графи;
   – у варіантах 1, 2 можуть розглядатись як неорієнтовані, так і орієнтовані графи (ви можете обрати, з яким видом графів проводити експерименти);
   – у варіантах 9, 10 розглядаються неорієнтовані зважені графи;
   – у варіантах 3, 4, 5, 13, 17 розглядаються орієнтовані зважені графи.

Ваші реалізації типу даних «граф» повинні враховувати особливості того класу графів, який відповідає алгоритму з вашого варіанту. Також умова задачі, яку розв'язує алгоритм, може накладати додаткові обмеження на структуру графу (так, у варіантах 6 та 7 графи повинні бути ациклічними – відповідно, під час генерування графів треба одразу перевіряти їх на ациклічність, або передбачити у реалізації алгоритму відповідну поведінку із повертанням помилки, якщо граф містить цикли).

2. Генерування випадкових графів повинно задаватись двома параметрами: кількістю вершин n та щільністю δ, тобто відношенням кількості ребер до максимально можливої. Наприклад, при $n = 100$ максимальна кількість ребер у неорієнтованому графі складає $C_n^2 = 5050$ ребер, а в орієнтованому – $A_n^2 = 10100$ ребер; тому при щільності $\delta = 10\%$ неорієнтований граф такого розміру повинен містити приблизно 505 ребер, а орієнтований – приблизно 1010 ребер. Для генерування випадкових графів рекомендовано використовувати модель Ердеша-Реньї (https://en.wikipedia.org/wiki/Erdős–Rényi_model, також можна подивитись моє трішки хаотичне відео https://youtu.be/-Y_I6OGAqCM?si=YeZeN-QAZTb49TOk&t=1746).

Для випадкових зважених графів також необхідно генерувати вагу ребер. Можливі значення ваг залишаються на ваш розсуд.

3. Метою проведення чисельних експериментів у проекті є оцінка складності досліджуваного алгоритму в залежності від розміру та щільності графу. Відповідно, для проведення експериментів необхідно розглядати графи різного розміру, і для кожного зафіксованого розміру – різної щільності. Кількість експериментів при цьому повинна бути достатньою, щоб прибрати залежність від шумових факторів; час роботи алгоритму визначається в середньому. Наприклад, для кожної пари значень «розмір, щільність» треба виконати не менше 20 експериментів, причому для кожного експерименту генерувати окремий випадковий граф, і робити це таким чином, щоб генерування графів не йшло у час роботи алгоритму, який ви досліджуєте.

Розміри графів необхідно обирати в межах від 20 до 200 вершин, значення щільності – в залежності від задачі, яку розв'язує алгоритм. Так, для задачі перевірки зв'язності або перевірки планарності раціонально розглядати графи низької та середньої щільності, а для задач пошуку шляхів мінімальної ваги – середньої та високої щільності. Рекомендовано для проведення експериментів обрати п'ять різних значень щільності.

4. У більшості варіантів необхідно також оцінити вплив представлення графу на складність досліджуваного алгоритму. Відповідно, необхідно провести дві серії чисельних експериментів, в одній з яких розглядати графи, представлені матрицею суміжності, а в іншій – списками суміжності.

Зауваження. Хоча для багатьох алгоритмів рекомендовано те чи інше представлення графів (наприклад, списки суміжності для DFS), це не означає, що ці алгоритми не можуть працювати з іншими представленнями. Власне, одна із задач експериментів – показати, наскільки одне представлення буде ефективнішим за інше. Винятками є варіанти 2 та 4, в яких алгоритми принципово працюють лише із матрицями суміжності. У цих варіантах експерименти зі списками суміжності ставити не потрібно.

5. Варіанти 14-17 і частково варіант 13 містять маленький аналітичний елемент, оскільки в них вам необхідно розглянути існуючі алгоритми, направлені на розв'язання поставленої задачі, та обрати серед них той, який вам більше прийшовся до душі. У цих варіантах від вас не вимагається проведення якогось масштабного порівняльного аналізу; однак у звіті ви повинні навести коротеньке обґрунтування, чому ви обрали саме такий алгоритм, який обрали.

Відправною точкою для пошуку теоретичного матеріалу можуть стати (взагалі кажучи, будь-які) підручники з теорії графів та з теорії алгоритмів. Додаткові дані можна знайти у тематичних наукових статтях, які рекомендовано шукати через відкриті бази наукових публікацій: arXiv (https://arxiv.org/), Google Scholar (https://scholar.google.com), CiteSeerX (https://citeseerx.ist.psu.edu). Також можна з певною обережністю розглядати матеріали тематичних форумів на кшталт Reddit або MathExchange.

6. Мова програмування для реалізації обирається вами самостійно. Але категорично не рекомендується використовувати скриптові мови (такі як Java Script), функціональні мови програмування, асемблер та мову BrainFuck.

Оформлення звіту з виконання проекту

Звіт з виконання проекту здається у вигляді PDF-файлу. Звіт повинен містити:

1) інформацію про виконавців проекту, достатню для їх однозначної ідентифікації (бажано на титульній сторінці);
2) формальний опис алгоритму, який досліджується, зокрема:
   - чітка постановка задачі, яку розв'язує алгоритм;
   - вхідні та вихідні дані для алгоритму;
   - опис алгоритму псевдокодом;
   - теоретичні оцінки складності;
3) коротенькі коментарі щодо програмної реалізації: які рішення та типи даних було вами обрано та реалізовано, чому саме такі;
4) посилання на GitHub-репозиторій із усіма вихідними кодами ваших програмних реалізацій;
5) опис експериментальної частини:
   - схема експериментів, вибір параметрів (розміри, щільність);
   - результати експериментів у вигляді таблиць та графіків; зокрема, для кожної форми представлення графу та для кожного обраного рівня щільності повинен бути графік швидкості роботи алгоритму від розміру графу; графіки можна суміщати, якщо їх візуальне сприйняття від цього не погіршується;
   - аналіз одержаних результатів та їх інтерпретація, порівняння із теоретичними оцінками, порівняння впливу різних форм представлення графу;
6) висновки до роботи.

Не включайте у звіт тексти ваших програм, заощаджуйте електронний папір. Посилання на git буде більш ніж достатньо.

Оцінювання проекту

Оцінювання проекту відбувається за такими критеріями:

1) коректність та якість програмного коду: до 5 балів;
2) постановка та проведення експериментів: до 3 балів;
3) представлення одержаних результатів: до 3 балів.
4) оформлення звіту, якість написаного тексту: до 4 балів;

При оформленні звіту з виконання проекту у системі LaTeX ставиться +2 додаткових бали поза шкалою оцінювання. У цьому випадку разом із PDF-файлом звіту ви повинні здати й tex-івські вихідні файли.