Лабораторна робота № 5

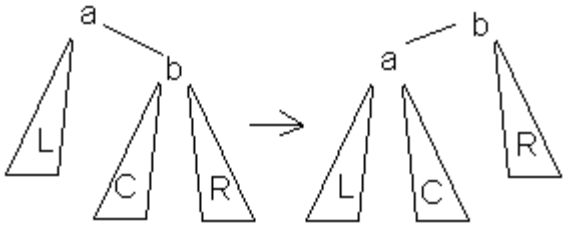
Бінарне дерево. Граф. Пошук в ширину

Балансування бінарного дерева пошуку

Завжди бажано , щоб всі шляхи в дереві від кореня до листів мали приблизно однакову довжину, тобто щоб глибина і лівого , і правого піддерев була приблизно однакова в будь-якому вузлі. В іншому випадку втрачається продуктивність.

У виродженому випадку може виявитися , що все ліве дерево порожньо і на кожному рівні є тільки праві дерева, і, в такому випадку, дерево вироджується у список (що йде вправо). Пошук (а значить , і видалення і додавання) у такому дереві по швидкості дорівнює пошуку в списку і набагато повільніше пошуку в збалансованому дереві.

Для балансування дерева застосовується операція "поворот дерева" . Поворот наліво виглядає так:



- було Left(A) = L, Right(A) = B, Left(B) = C, Right(B) = R
- поворот міняє місцями A і B, отримаємо Left(A) = L, Right(A) = C, Left(B) = A, Right(B) = R
- також змінюється у вузлі Parent (A) посилання, раніше вказувала на A, після повороту - вказує на B.

Поворот направо виглядає так само, досить замінити в наведеному вище прикладі все Left на Right і назад.

Досить очевидно, що поворот не порушує впорядкованість дерева, і приводить до передбачуваної (+1 або -1) зміни глибини всіх піддерев.

Завдання

1. Соціальну мережу можна представити у вигляді орієнтованого графа, де вершинами є облікові записи користувачів. Використовуючи пошук в ширину знайдіть всіх друзів певного користувача, а також друзів його друзів. Виведіть їх список. Перевірте чи знають вони початкового користувача

2. Реалізуйте бінарне дерево пошук. А також методи додавання у нього певного значення, та повороту дерева вліво та вправо.

3. Соціальну мережу можна представити у вигляді орієнтованого графа, де вершинами є облікові записи користувачів. Використовуючи пошук в ширину знайдіть який корисувач є максимально "віддаленим" від вас і через скількох інших людей ви з ним звязані. Виведіть список цих людей.

4. Реалізуйте бінарне дерево пошук. А також методи додавання у нього певного значення, пошуку значення та видалення. Після операцій додавання та видалення дерево має лишатись бінарним деревом пошуку.

5. Соціальну мережу можна представити у вигляді орієнтованого графа, де вершинами є облікові записи користувачів. Використовуючи пошук в ширину знайдіть спільних друзів, та спільних друзів друзів двох користуваів. Виведіть їх список.

6. Допоможіть інтернет-крамниці! Крамниція має звязок з соціальною мережею, і коли хотось купує в ній якійсь товар, то ця інформація зберігається в описі користувача. Використовуючи пошук в ширину разробіть сервіс рекомендацій для інтернет-крамниці. Він має визначати чи рекомендувати певний товар користувачеві, якщо його придбало більше 50% його друзів та більше 70% друзів його друзів.

7. Задачу пошуку виходу з лабіринту, представленого наступною схемою (внизу), зручно вирішувати за допомогою методу пошуку в ширину. Нехай прямокутний лабіринт представляється матрицею L[m, n], де значення L[i, j] кодують тип клітини (i, j): вільний простір (0), стіна (1), вхід (2) або вихід (3).

```
2 1 0 0 0
0 1 0 1 0
0 1 0 1 0
0 0 0 1 3
```

   Побудуйте граф з m * n вершин (кожна вершина відповідає одній клітці лабіринту), та пошуком в ширину знайти найкоротший шлях від входу до виходу.

8. Реалізуйте алгоритм сортування на основі бінарного дерева. Алгоритм полягає у тому, що елементи вхідного списку по черзі додаються в дерево бінарного пошуку, а потім виконується обхід дерева у результаті якого отримуємо відсортований список.
   Умовно процедура обходу дерева буде наступна:

```
def traverse(tree):
      traverse(tree.left)
      print tree.cargo
      traverse(tree.right)
```

9. Використовуючи алгоритм пошуку в ширину на графі, знайдіть чи є спільна ділянка маршруту з вершини A -> C та в B -> C. Знайдіть яка довжина цієї дільник та які вершини на ній містяться.