**Київський національний університет імені Тараса Шевченка**

**Факультет комп’ютерних наук та кібернетики**

**Кафедра інформаційних систем**

**Алгоритми та складність**

**«Піраміда Фібоначчі»**

**Звіт**

**Виконав:**

студент групи К-29

Зайченко Сергій Павлович

**Київ-2016**

**1.Опис**

Купа Фібоначчі — [абстрактна структура даних](https://uk.wikipedia.org/w/index.php?title=%D0%90%D0%B1%D1%81%D1%82%D1%80%D0%B0%D0%BA%D1%82%D0%BD%D0%B0_%D1%81%D1%82%D1%80%D1%83%D0%BA%D1%82%D1%83%D1%80%D0%B0_%D0%B4%D0%B0%D0%BD%D0%B8%D1%85&action=edit&redlink=1), яка є ефективною реалізацією [черги з пріоритетом](https://uk.wikipedia.org/wiki/%D0%A7%D0%B5%D1%80%D0%B3%D0%B0_%D0%B7_%D0%BF%D1%80%D1%96%D0%BE%D1%80%D0%B8%D1%82%D0%B5%D1%82%D0%BE%D0%BC).

З теоретичної точки зору купи Фібоначчі особливо варто використовувати, коли кількість Extract-Min і Delete операцій мала порівняно з кількістю інших операцій. Наприклад, деякі алгоритми на графах можуть викликати Decrease-Key на кожному ребрі. Для насиченого графу [амортизаційний час](https://uk.wikipedia.org/wiki/%D0%90%D0%BC%D0%BE%D1%80%D1%82%D0%B8%D0%B7%D0%B0%D1%86%D1%96%D0%B9%D0%BD%D0%B8%D0%B9_%D0%B0%D0%BD%D0%B0%D0%BB%D1%96%D0%B7) Θ ( 1 )кожного виклику Decrease-Key складається у велику перевагу в порівнянні з Θ ( lg n ) у найгіршому випадку в [бінарній купі](https://uk.wikipedia.org/wiki/%D0%91%D1%96%D0%BD%D0%B0%D1%80%D0%BD%D0%B0_%D0%BA%D1%83%D0%BF%D0%B0). Це можна побачити на прикладі алгоритмів [про найкоротші шляхи з одного входу](https://uk.wikipedia.org/wiki/%D0%97%D0%B0%D0%B4%D0%B0%D1%87%D0%B0_%D0%BF%D1%80%D0%BE_%D0%BD%D0%B0%D0%B9%D0%BA%D0%BE%D1%80%D0%BE%D1%82%D1%88%D0%B8%D0%B9_%D1%88%D0%BB%D1%8F%D1%85) і [мінімального кістякового дерева](https://uk.wikipedia.org/wiki/%D0%9C%D1%96%D0%BD%D1%96%D0%BC%D0%B0%D0%BB%D1%8C%D0%BD%D0%B5_%D0%BA%D1%96%D1%81%D1%82%D1%8F%D0%BA%D0%BE%D0%B2%D0%B5_%D0%B4%D0%B5%D1%80%D0%B5%D0%B2%D0%BE). З практичної точки зору сталий множник прихований у складності алгоритму і складність у програмуванні купи Фібоначчі роблять її менш бажаною, ніж звичайну бінарну або d-арну купу для більшості застосувань.

В результаті розслабленої структури деякі операції можуть зайняти багато часу, а інші - дуже швидко. Для аналізу часу амортизації ми використовуємо потенційний метод, що полягає в тому, що ми робимо вигляд, що дуже швидкі операції займають трохи більше часу, ніж вони насправді роблять. Це додатковий час потім об'єднується і віднімається з фактичного часу роботи повільних операцій. Обсяг часу, збереженого для подальшого використання, вимірюється в будь-який даний момент потенційної функцією. Потенціал купи Фібоначчі визначається

**Потенціал = t + 2 м**  
де t - кількість дерев в купі Фібоначчі, а m - кількість зазначених вузлів. Вузол відзначений, якщо хоча б один з його дочірніх елементів був вирізаний, оскільки цей вузол був зроблений дочірнім елементом іншого вузла (всі корені не відмічені). Амортизований час для операції визначається сумою фактичного часу і c різниці в потенціалі, де c - константа (обрана так, щоб відповідати постійним коефіцієнтами в позначенні O для фактичного часу).

Щоб забезпечити швидке видалення і конкатенацію, коріння всіх дерев пов'язані з допомогою циклічного, двічі пов'язаного списку. Діти кожного вузла також пов'язані з використанням такого списку. Для кожного вузла ми зберігаємо ї’х кількість і чи позначений вузол. Крім того, ми зберігаємо покажчик на корінь, що містить мінімальний ключ.

Тепер **пошук мінімального елемента** тривіальний, тому що ми зберігаємо покажчик на вузол, що містить його. Це не змінює потенціал куч, тому як фактична, так і амортизована вартість є постійними.

Як згадано вище, **злиття** здійснюється просто шляхом об'єднання списків корінь дерев з двох куч. Це можна зробити в постійне час, і потенціал не зміниться, що знову приведе до постійного амортизованого часу.

Операція **вставки** працює, додаючи новий елемент правіше від мінімального. Це займає постійний час, а потенціал збільшується на одиницю, тому що число дерев збільшується. Таким чином, амортизована вартість залишається постійною.

**Виділення мінімального елемента** (так само, як видалення мінімуму) працює в три етапи. Спочатку ми беремо корінь, що містить мінімальний елемент, і видаляємо його. Його діти стануть корінням нових дерев. На цьому можна було б і завершити, оскільки ми отримали правильну Фібоначчієву купу. Однак, не складно побачити, що досі описані операції над купою, роблять список коренів довшим і довшим. Отже, проходження через увесь список коренів ставатиме обчислювально дорожчим. Отже, якщо ми вже маємо зробити якусь роботу у будь-якому випадку, то ми виконаємо очищення зараз, щоб уникнути більшої роботи у наступному. Тому, доти доки наявні два дерева з однаковим рангом, скажімо k, ми зливаємо ці дерева, щоб отримати дерево рангу k+1. Злиття полягає у порівнянні коренів і додавання дерева з більшим коренем як дочірнього до другого кореня. Зауважимо, що оскільки злиття може створити друге дерево рангу k+1 у колекції, один корінь може взяти участь у кількох злиттях.Якщо кількість дітей дорівнює d, для обробки всіх нових коренів потрібно час O (d), а потенціал збільшується на d -1. Отже, амортизований час роботи цього етапу дорівнює O (d) = O (log n).

Найгірший випадок.

Хоча купи Фібоначчі виглядають дуже ефективними, у них є два недоліки: "Вони складні, коли мова йде про їх кодування. Також вони не такі на практиці, порівняно з теоретично менш ефективними формами купи, оскільки в їх простій версії вони вимагають зберігання та маніпулювання чотирма вказівниками на вузол, у порівнянні з двома або трьома вказівниками на вузол, необхідний для інших структур ». []](https://translate.googleusercontent.com/translate_c?act=url&depth=1&hl=ru&ie=UTF8&prev=_t&rurl=translate.google.com&sl=en&sp=nmt4&tl=uk&u=https://en.wikipedia.org/wiki/Fibonacci_heap&xid=17259,1500004,15700002,15700023,15700105,15700124,15700149,15700168,15700173,15700201&usg=ALkJrhjsziZdvrE7IeKtT6wO08hYx83OCg#cite_note-FSST-3) Ці інші структури називаються " [Двійкова купа"](https://translate.googleusercontent.com/translate_c?act=url&depth=1&hl=ru&ie=UTF8&prev=_t&rurl=translate.google.com&sl=en&sp=nmt4&tl=uk&u=https://en.wikipedia.org/wiki/Binary_heap&xid=17259,1500004,15700002,15700023,15700105,15700124,15700149,15700168,15700173,15700201&usg=ALkJrhiyTycRBLQ0s1moonClL1a9g1_BzA) , "[Біноміальна купа"](https://translate.googleusercontent.com/translate_c?act=url&depth=1&hl=ru&ie=UTF8&prev=_t&rurl=translate.google.com&sl=en&sp=nmt4&tl=uk&u=https://en.wikipedia.org/wiki/Binomial_heap&xid=17259,1500004,15700002,15700023,15700105,15700124,15700149,15700168,15700173,15700201&usg=ALkJrhgYZ2taj7FhYsoGv1iA0j6fokzDiw) .

Хоча загальний час роботи послідовності операцій, що починаються з порожньої структури, обмежується межами, наведеними вище, деякі (дуже мало) операції в послідовності можуть зайняти дуже багато часу (зокрема видалення та видалення мінімуму мають лінійний час роботи в найгірший випадок). З цієї причини купи Фібоначчі та інші амортизовані структури даних можуть не підходити для [систем реального часу](https://translate.googleusercontent.com/translate_c?act=url&depth=1&hl=ru&ie=UTF8&prev=_t&rurl=translate.google.com&sl=en&sp=nmt4&tl=uk&u=https://en.wikipedia.org/wiki/Real-time_computing&xid=17259,1500004,15700002,15700023,15700105,15700124,15700149,15700168,15700173,15700201&usg=ALkJrhhG3eEi9IZgX7c5RXTkpCL3lyTF0A) . Можна створити структуру даних, яка має такий же час роботи у найгіршому випадку, як купа Фібоначчі яка має амортизовану продуктивність.

**2.Реалізація**

Програма пропонує скористатися функціями вставки, виділення мінмального елемента з структури і виведення мімального елемента на єкран. Функції видалення будь-якого елемента, обєднання двух куп і зменшення ключа доданні до реалізації лише для повноти струкутури. Піраміда створюється автоматично до початку роботи з користувачем,

