Київський національний університет імені Тараса Шевченка

Факультет комп`ютерних наук та кібернетики

Алгоритми та складність

Завдання №8

“Реалізація піраміди Фібоначчі”

Виконав студент 2-го курсу

Групи К-29

Пащенко Дмитро Вікторович

2020

**Завдання**:

Реалізувати піраміду Фібоначчі, застосувавши її до моделі, описаної у Вашому варіанті.

**Варіант 15:**

Предметна область: Залізниця

Об'єкти: Дороги, Станції

Примітка: Є безліч залізничних доріг. У відомстві кожної дороги знаходиться безліч станцій.

**Теорія:**

* Піраміда Фібоначчі — структура даних, яка є ефективною реалізацією черги з пріоритетом.
* Піраміда Фібоначчі являє собою набір біноміальних дерев з такими властивостями:
  + вузли дерев відповідають елементам, що зберігаються в черзі;
  + корені дерев поєднані у двобічно зв'язаний список;
  + зберігається вказівник на корінь дерева, який відповідає елементу з найменшим ключем;
  + для кожного вузла зберігається його степінь, тобто кількість його дітей, а також чи він позначений;
  + Вимога розміру: якщо вузол u має степінь k, то піддерево з коренем u має щонайменше Fk+2 вузлів, де Fi — це i-те число Фібоначчі, тобто F0 = 0, F1 = 1 і Fi = Fi-1 + Fi-2 для .
* З теоретичної точки зору купи Фібоначчі особливо варто використовувати, коли кількість Extract-Min і Delete операцій мала порівняно з кількістю інших операцій.
* Усі оцінки складності наведених далі алгоритмів амортизаційні.

**Мова програмування:** С++.

**Інтерфейс користувача**

Введення даних відбувається через консоль.

Вхідні дані: команди

* 0 – завершити програму
* 1 – вставити залізницю (за назвою)
* 2 – вилучити мінімальний ключ
* 3 – перевірити наявність ключа в піраміді
* 4 – видалити залізницю (за назвою)
* 5 – вивести піраміду

Результат: фібоначчієва піраміда залізниць.

**Модулі програми:**

* **void FibonacciHeap<T>::Node::link(Node\* new\_child)**

Під'єднує зліва до даного вузла нового сина.

**Складність:** .

* **Node\* FibonacciHeap<T>::getMin() const**

Повертає мінімальний ключ.

**Складність:** .

* **void FibonacciHeap<T>::unite(FibonacciHeap<T> &to\_unite)**

Зливає вказану піраміду з даною, вказана піраміда очищується.

**Складність:** .

* **Node\* FibonacciHeap<T>::search(T const &key, Node\* current) const**

Шукає вершину за ключем.

**Складність:** , де n – кількість вузлів піраміди.

* **void FibonacciHeap<T>::insert(T const &key)**

Вставляє ключ в піраміду.

**Складність:** .

* **void FibonacciHeap<T>::consolidate()**

Зливає біноміальні дерева однакового степеня.

**Складність:** , де n – кількість вузлів піраміди.

* **T FibonacciHeap<T>::extractMin()**

Видаляє мінімальний ключ з піраміди й повертає його.

**Складність:** , де n – кількість вузлів піраміди.

* **void FibonacciHeap<T>::cascadingCut(Node\* current)**

Каскадно вилучає помічені вершини та додає їх до списку коренів, або ж помічає непоміченого батька.

**Складність:** .

* **void FibonacciHeap<T>::decreaseKey(Node\* aim, T const &new\_key)**

Змінює ключ вказаної вершини, якщо новий ключ менше.

**Складність:** .

* **void FibonacciHeap<T>::remove(T const &key)**

Видаляє ключ з піраміди.

**Складність:** , де n – кількість вузлів піраміди.

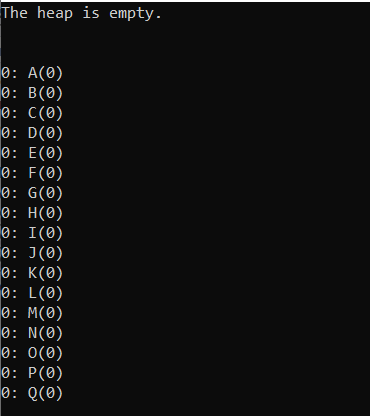
* **void FibonacciHeap<T>::print() const**

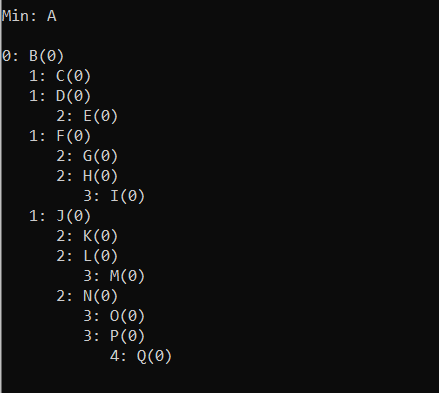
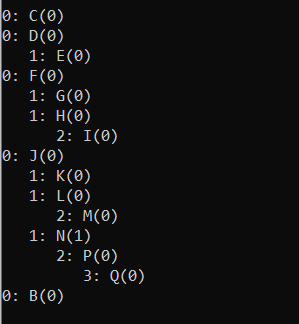
Друкує піраміду.

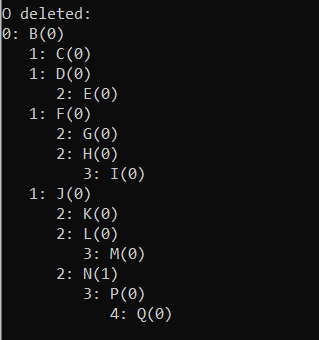
**Складність:** , де n – кількість вузлів піраміди.

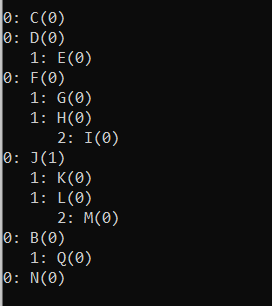
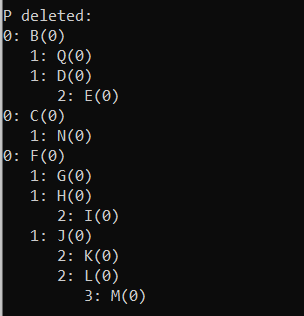
**Тестові приклади**

1. Створюю порожню піраміду. Задля зручності та незагромадженості називатиму залізниці великими буквами латинського алфавіту. Послідовно додав до піраміди ключі від «А» до «Q». Бачимо, що вершини додаються в двозв’язний список. Примітка в дужках означає поміченість вершини («m», якщо помічена, інакше «0»).



1. Вилучимо мінімальний ключ операцією extractMin(). Переконалися, що вилучився мінімальний ключ. Піраміда тепер почала ущільнення операцією consolidate(). Врешті-решт, маємо правильну структуру піраміди Фібоначчі. Кількість ключів після вилучення – 16. У двійковому записі – 10000. Тобто наразі піраміда складається з одного біноміального дерева степеня 4.
2. Видалимо ключ «O». Спочатку за лінійний час відбувається його знаходження в піраміді. Далі він заміняється (decreaseKey) на поточний мінімальний елемент – «В» – вилучається з піраміди й додається в список коренів. Його батько – N – помічається. Врешті решт, викликається процедура вилучення мінімального елемента – тобто, кореневої вершини «В». Діти «В» додаються в список коренів.
3. Операція consolidate() ущільнює піраміду. Отримуємо нову піраміду з 15 вершинами. Це без однієї вершини дерево В4, що задовольняє властивостям піраміди Фібоначчі.



1. Видаляю ключ «P». За попередньо описаною процедурою, ключ «P» стає ключем «В». Але цього разу вилучається не тільки він, але й каскадно вилучається його помічений батько N (причому мітка зникає). Далі, за процедурою видалення, перше біноміальне дерево з коренем «В» вилучається з піраміди. Діти вершини «В» відокремлюються від батька й додаються в список коренів.
2.  Нарешті викликається consolidate(). Дерево J підчіплюється до дерева F, дерево D підчіплюється до дерева B, дерево N підчіплюється до дерева C. Процедура завершується. Маємо три правильних біноміальних дерева.

**Література**

* Кормен, Лейзерсон, Рівест, Штайн. Алгоритми: побудова і аналіз, 2-е видання. – 2005
* <https://uk.wikipedia.org/wiki/%D0%A4%D1%96%D0%B1%D0%BE%D0%BD%D0%B0%D1%87%D1%87%D1%96%D1%94%D0%B2%D0%B0_%D0%BA%D1%83%D0%BF%D0%B0>