Київський національний університет імені Тараса Шевченка
Факультет комп’ютерних наук та кібернетики
Кафедра інтелектуальних програмних систем

Алгоритми та складність

Лабораторна робота №2
«Біноміальна Піраміда»

Варіант 20. Тип даних T1

Виконав студент 2-го курсу
Групи ІПС-21

Шевнюк Михайло Олексійович

2025

Завдання
Реалізувати біноміальну піраміду (heap) для зберігання раціональних чисел.

Підтримує операції: створення порожньої піраміди(**O(1))** | вставка елемента (**O(log n)**) | злиття двох пірамід(**O(log n)** | отримання та вилучення мінімуму(**O(log n)**

Теорія
Біноміальна піраміда — це множина біноміальних дерев, у якій для кожного порядку k існує не більше одного дерева. Ключі в кожному дереві утворюють мін-піраміду. Кількість дерев не перевищує $\lfloor \log_2 n \rfloor + 1$, де n — число вузлів.

Основні властивості забезпечують ефективне злиття пірамід, завдяки чому `insert` реалізується як злиття з пірамідою з одного вузла.

Алгоритм побудови

1 Злиття кореневих списків

З’єднати два впорядковані за ступенем списки коренів у **O(log n)**.

Пройтися результатом, поєднуючи дерева однакового степеня операцією `link` (молодше дерево стає лівим сином старшого).

2 Вставка

Створити піраміду з одного вузла й застосувати `merge` — оцінка **O(log n)**.

3 Вилучення мінімуму

Знаходимо мінімальний корінь (не більше log n перевірок).

Від’єднуємо його дітей, розгортаємо порядок та `merge` з поточною пірамідою.

Складність
• Побудова – O(n) очікувано;
• Пошук – O(1) у найгіршому випадку;
• Пам’ять – O(n).

Мова програмування
C++

Модулі програми

1. **Node** — вузол дерева: `value`, `degree`, вказівники `parent`, `sibling`, `child`; метод `link()`.
2. **BinomialHeap<T, Comp>** — основний контейнер.
    a. Сховище — однозв’язний список коренів.
    b. Компаратор `Comp` (тип-параметр, за замовчуванням `less<T>`).
    c. RAII: рекурсивний деструктор, копіювання/пересилання заборонено.
3. **Student** — id (auto-increment), `name`.
4. **Group** — `title` (регекс `^[A-Z]{1,2}[0-9]{2}$` приймає *PI11*, *K11* тощо), вектор `unique_ptr<Student>`.
5. **main()** — демонстрація: створення піраміди зі студентських груп, `insert`, `extractMin`, `merge`.

Інтерфейс користувача
Користувач не вводить дані вручну; демонстраційний набір слів заданий у коді. При бажанні можна зчитати набір із файлу.

Тестовий приклад

```
//============================================================================
//  DEMONSTRATION
//============================================================================
int main() {
    try {
        //── Rational demo ──────────────────────────────────────
        vector<Rational> fracs = { {n:3, d:10}, {n:1, d:2}, {n:5, d:6}, {n:7, d:8},
                                   {n:2, d:3}, {n:9, d:10}, {n:11, d:12}, {n:13, d:14} };
        BinomialHeap<Rational> ratHeap;
        for (const auto& r :const Rational & : fracs) ratHeap.insert(r);

        cout << "NEW TREE (Rational)\n";
        ratHeap.print();

        cout << "EXTRACT MIN\n";
        ratHeap.extractMin();
        ratHeap.print();

        cout << "\n──────────────────────────────────────────────────────────\n\n";

        //── Integer demo ───────────────────────────────────────
        BinomialHeap<int> intHeap;
        for (int i = 0; i < 10; ++i) intHeap.insert(i);

        cout << "NEW TREE (int)\n";
        intHeap.print();

        cout << "EXTRACT MIN\n";
        intHeap.extractMin();
        intHeap.print();
    }
    catch (const exception& e) {
        cerr << "Error: " << e.what() << '\n';
        return 1;
    }
    return 0;
}
```

Результат

NEW TREE (Rational)

Binomial tree #0 (degree 3)

└─3/10

├─2/3

│ ├─11/12

│ │ └─13/14

│ └─9/10

├─5/6

│ └─7/8

└─1/2

EXTRACT MIN

Binomial tree #0 (degree 0)

└─1/2

Binomial tree #1 (degree 1)

└─5/6

└─7/8

Binomial tree #2 (degree 2)

└─2/3

├─11/12

│ └─13/14

└─9/10

NEW TREE (int)

Binomial tree #0 (degree 1)

└─8

* └─9*

Binomial tree #1 (degree 3)

└─0

* ├─4*

* │ ├─6*

* │ │ └─7*

* │ └─5*

* ├─2*

* │ └─3*

* └─1*

EXTRACT MIN

Binomial tree #0 (degree 0)

└─1

Binomial tree #1 (degree 3)

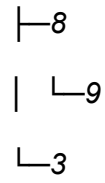
└─2

* ├─4*

* │ ├─6*

* │ │ └─7*

* │ └─5*



Висновки

Біноміальна піраміда оптимальна, коли потрібне часте злиття структур (наприклад, у чергах подій паралельних процесів). RAII-рефакторинг та шаблонний компаратор зробили код безпечним щодо витоків і гнучким щодо типів даних.

Використана література

1. Томас Г. Кормен та ін. «Алгоритмы. Построение и анализ» (розд. 19).
2. CLRS repository — *binomial-heap* reference.
3. Курс лекцій «Алгоритми та складність», КНУ (2025).