Київський національний університет імені Тараса Шевченка

Факультет комп`ютерних наук та кібернетики

Кафедра інтелектуальних програмних систем

Алгоритми та складність

Завдання№3

**d-арні піраміди**

Виконала студентка 2-го курсу

Групи К-28

Панченко Тетяна Андріївна

2020

**Завдання**

d-арні піраміди схожі на бінарні, лише їх вузли, відмінні від листя, мають не по 2, а по d дочірніх елементів. Представте d-арну піраміду у вигляді масиву (якою буде її висота для n елементів?). Розробіть ефективні реалізації процедур Extract\_Max, Insert та Increase\_Key, призначених для роботи з d-арною незростаючою пірамідою. Проаналізуйте час роботи цих процедур і виразіть їх в термінах n та d.

**Теорія**

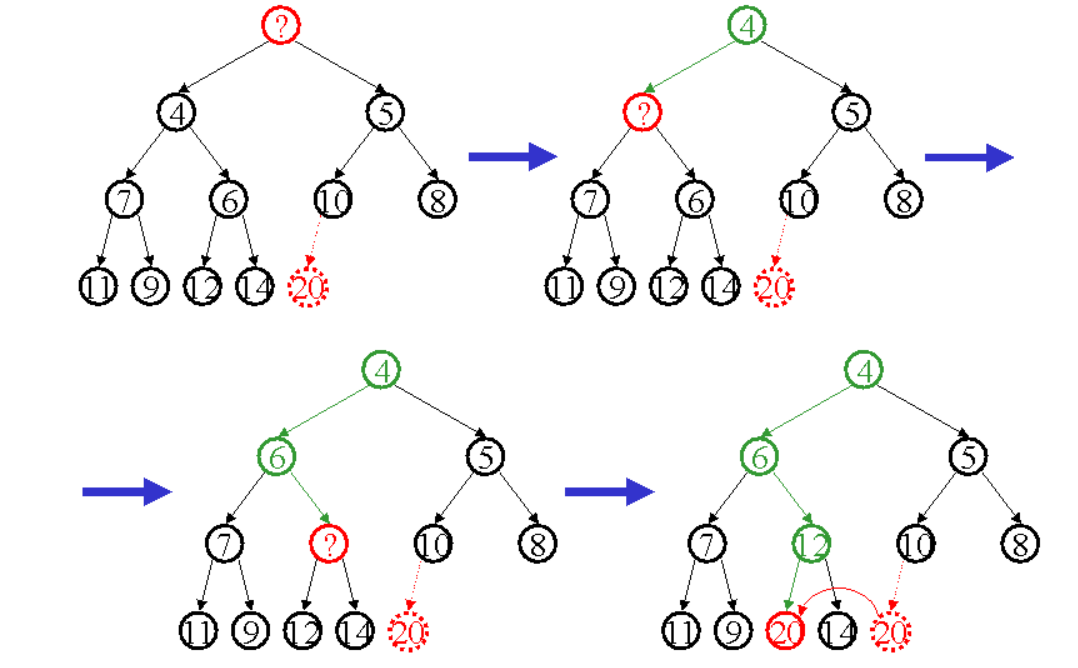
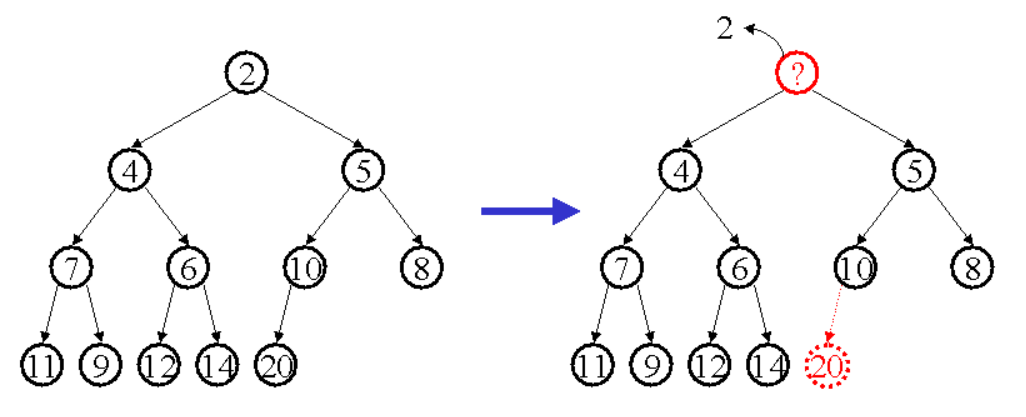
Для реалізації піраміди модифікуємо структуру Heap для d-арних вузлів.

**Купа** або **піраміда** ([англ.](https://uk.wikipedia.org/wiki/%D0%90%D0%BD%D0%B3%D0%BB%D1%96%D0%B9%D1%81%D1%8C%D0%BA%D0%B0_%D0%BC%D0%BE%D0%B2%D0%B0" \o "Англійська мова) *heap*) — спеціалізована [деревовидна](https://uk.wikipedia.org/wiki/%D0%94%D0%B5%D1%80%D0%B5%D0%B2%D0%BE_(%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)" \o "Дерево (структура даних)) [структура даних](https://uk.wikipedia.org/wiki/%D0%A1%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" \o "Структура даних), в якій існують певні властивості впорядкованості: якщо *В* — [вузол](https://uk.wikipedia.org/wiki/%D0%94%D0%B5%D1%80%D0%B5%D0%B2%D0%BE_(%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)" \o "Дерево (структура даних)) нащадок *A* — тоді ключ(*A*) ≥ ключ(*B*). З цього випливає, що елемент з найбільшим ключем завжди є кореневим вузлом. Купа є однією із найефективніших реалізацій абстрактного типу даних, який має назву [черга 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" \o "Черга з пріоритетом).

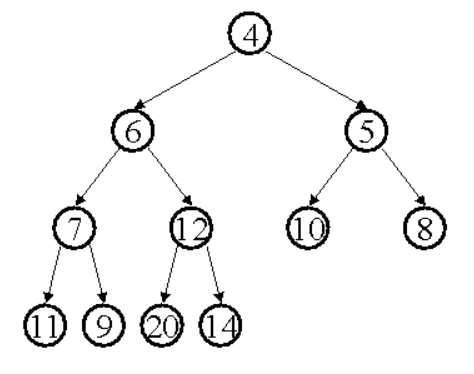
**Алгоритм**

**Extract\_max**:

1. Беремо перший елемент масиву (він є максимальним, так як піраміда незростаюча), його ми і повернемо.
2. Присвоюємо значення останнього елементу масиву першому і зменшуємо його розмір на один (таким чином видаляючи перший елемент)
3. Порівнюємо значення поточного вузла з усіма дітьми і обмінюємо з найбільшим серед них. Повторюємо цей крок, поки елемент не «стане на своє місце», тобто поки не почне виконуватись властивість піраміди.  
     
   Приклад роботи алгоритму для неспадаючої піраміди:



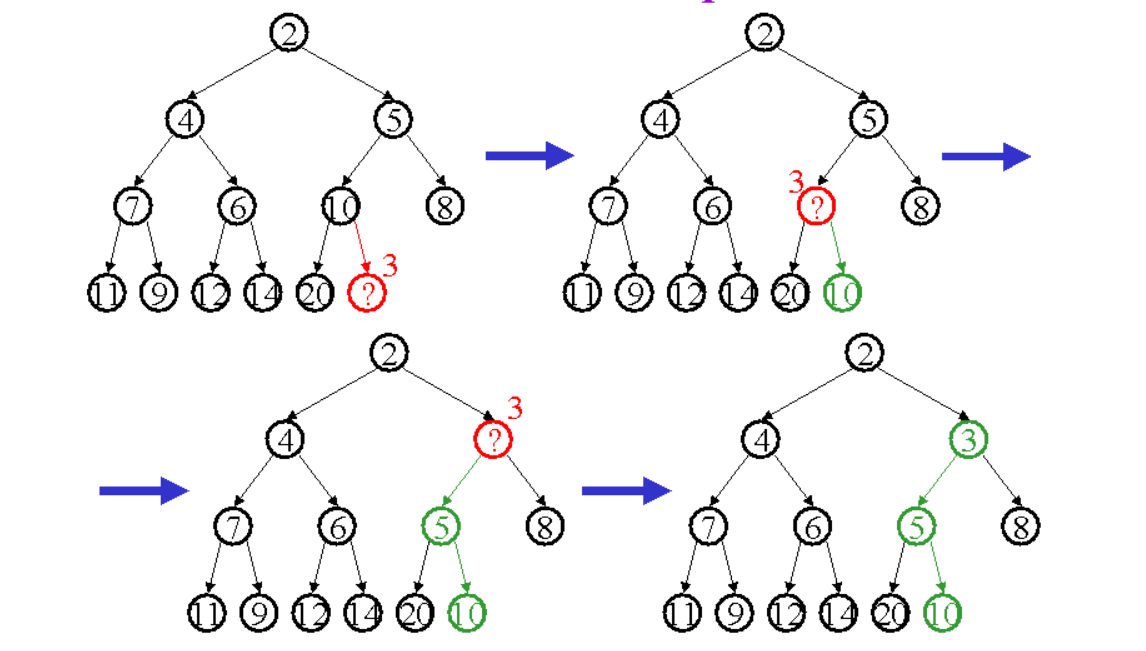
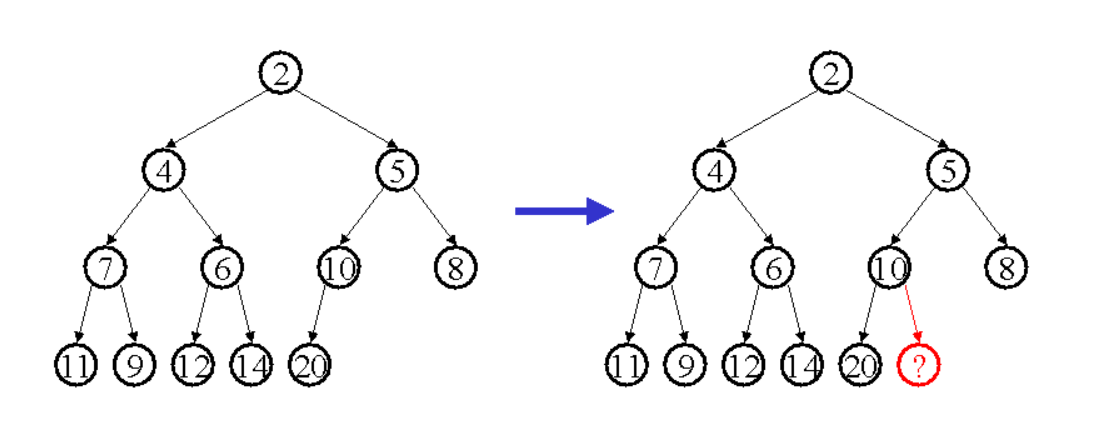
Результат:



**Insert**:

1. Додаємо в кінець масиву вузол -∞
2. За допомогою функції Increase\_key, замінюємо значення останнього елементу на задане та «піднімаємо» його по піраміді, поки не почне виконуватись властивість.

Приклад роботи алгоритму для неспадаючої піраміди:



**Increase\_key:**

1. Якщо нове значення ключа менше за попереднє, то алгоритм завершується
2. Присвоємо вузлу нове значення ключа.
3. Якщо поточний вузол є коренем або ключ батька більше за нове значення, то алгоритм завершує роботу, інакше міняємо місцями значення батьківського і поточного вузла і повторюємо цей крок для батька, поки не почне виконуватись властивість.

**Складність**

Складність роботи для кожної з функцій , так як в них відбувається перевірка властивості піраміди, складність якої пропорційна її висоті, решта операцій - O(1)

**Мова програмування**

С++

**Модулі програми**

Клас Heap:

Підтримка основної властивості купи:

void heapify(int index)

Вставка елемента в піраміду:

void Insert(int data)

«Витягування» максимального елементу:

int Extract\_max()

Збільшення ключа вибраного елементу піраміди:

void Increase\_key(int key, int step, int index)

**Інтерфейс користувача**

Вхідні дані вводяться з консолі і виводяться також в консоль.

**Тестові приклади**

Для d = 3

1. Вхід : 5 4 7 7 8 3 5

Після формування піраміди(heapify): 8 7 4 3 5 5 7

Після видалення максимального: 7 5 4 3 5 7

1. Вхід: 71 13 12 23 6 7 8 0 6 5 5

Після видалення максимального: 23 13 12 6 6 7 8 4 1 0 5 5

Після вставки «9»: 23 13 12 9 6 7 8 4 1 0 5 5 6

Після збільшення «5» : 25 13 12 23 6 7 8 4 1 0 9 5 6

**Висновки**

Усі реалізовані функції мають складність .

**Література**

1. Лекція №5
2. <https://en.wikipedia.org/wiki/D-ary_heap>
3. <https://www.geeksforgeeks.org/binary-heap/>
4. <https://uk.wikipedia.org/wiki/D-%D0%B0%D1%80%D0%BD%D0%B0_%D0%BA%D1%83%D0%BF%D0%B0>
5. https://courses.cs.washington.edu/courses/cse326/00wi/handouts/lecture5/sld014.htm