**Київський національний університет імені Тараса Шевченка**

**факультет комп’ютерних наук та кібернетики**

**кафедра інформаційних систем**

**Лабораторна робота № 3**

**“k-арна піраміда”**

**Виконав студент 2 курсу**

**групи К-29**

**Аксой Денис Гюралпович**

**2018**

1. **Зміст завдання та уточнення умови**

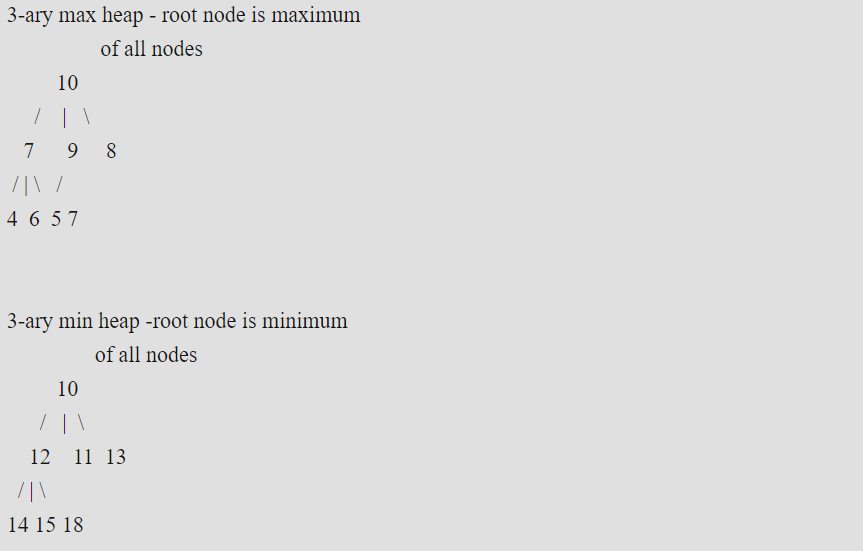
Представити k-арну піраміду у вигляді масиву. Розробити методи Exctract\_Max, Insert, Increase\_Key. Проаналізувати час роботи в термінах n та k, де n – кількість елементів массиву.

1. **Вступ до пояснення алгоритму**

K-арна піраміда – це узанальнення бінарної піраміди(К=2), в якій кожен вузол має K дітей замість 2. Як і бінарна піраміда К-арна підпорядковується двом таким властивостям:

1) Майже повне К-арне дерево, з усіма рівнями, що мають максимальну кількість вузлів крім останнього, що заповнюється зліва направо.

2) Як і бінарну піраміду, К-арну можна розділити на дві категорії: (а) Макс K-арна піраміда (ключ у корені більший, ніж у всіх нащадків, що рекурсивно правильно для всіх вузлів). (б) Міні К-арна піраміда(ключ у корені менше, ніж у всіх нащадків, що є рекурсивно правильним для всіх вузлів)



Висота повного К-арного дерева з n-вузлами визначається .

K-ary heap при використанні в реалізації черги пріоритетів дозволяє швидше виконувати операцію зменшення ключа в порівнянні з бінарної пірамідою (O ) для двійкової піраміди vs для K-aрної). Проте, це призводить до того, що складність операції extractMin зростає до в порівнянні зі складністю при використанні двійкої піраміди для черги з пріоритетами. Це дозволяє K-арній піраміді бути більш ефективною в алгоритмах, де операції скорочення пріоритету більш поширені, ніж операція extractMin (). Приклад: алгоритм Дейкстри для найкоротшого шляху одного вузла і алгоритм Прима для мінімального остовного дерева.

1. **Реалізація**

Реалізація

Припускаючи індексування масиву з нуля 0, масив являє собою K-арну піраміду таку, що для будь-якого вузла, який ми розглядаємо:

Батько вузла в індексі i (крім кореневого вузла) знаходиться за індексом (i-1) / k

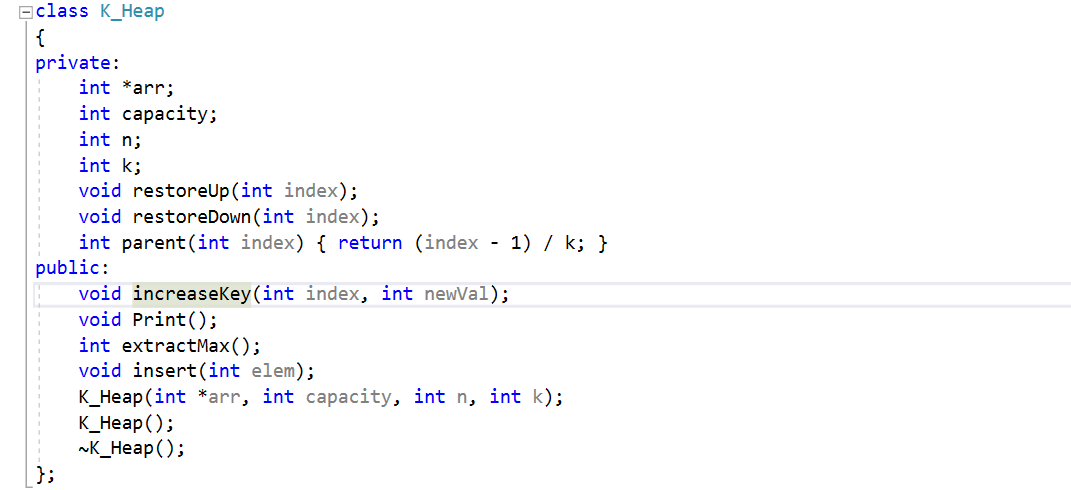
Діти вузла з індексом i знаходяться в індексах (k \* i) +1, (k \* i) +2 .... (k \* i) + k

Останній нелистовий вузол піраміди розміру n знаходиться за індексом (n-2) / k

Аналіз складності часу

* Для K-арної піраміди з n вузлами, максимальна висота даної піраміди буде . Таким чином, restoreUp () запускається для максимально разів(так як при кожній ітерації вузол зміщується на один рівень, для випадку restoreUp () або один рівень вниз у випадку restoreDown).
* restoreDown () викликає себе рекурсивно для k дітей. Тому часова складність цих функцій -
* • Insert і increaseKey операції викликаються restoreUp () один раз. Тобто складність - це .
* Оскільки extractMax () викликає restoreDown () один раз, його складність
* Часова складність побудованої піраміди O (n) (Аналіз схожий на бінарну піраміду)

1. **Основні модулі програми:**

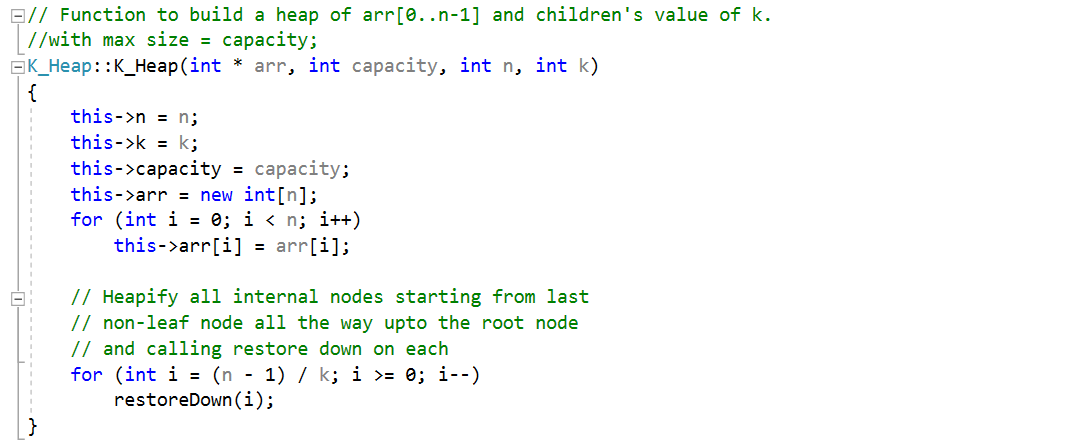


* Capacity – максимальна к-сть вузлів в піраміді
* N – к-сть вузлів в піраміді на даний момент
* K – арність піраміди
* buildHeap (): будує піраміду з вхідного масиву.

Ця функція запускає цикл, починаючи від останнього вузла, що не є листом, до кореневого вузла, викликаючи функцію restoreDown (також відома як maxHeapify) для кожного індексу, який відновлює переданий індекс у правильне положення піраміди шляхом переміщення вузла у напрямку кореня.

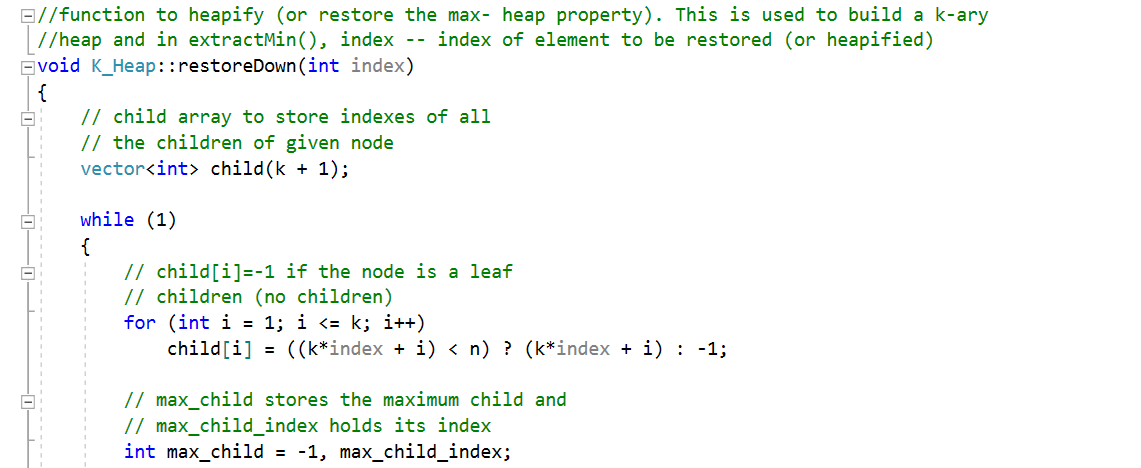
Чому ми починаємо цикл з останнього вузла, що не має листа?

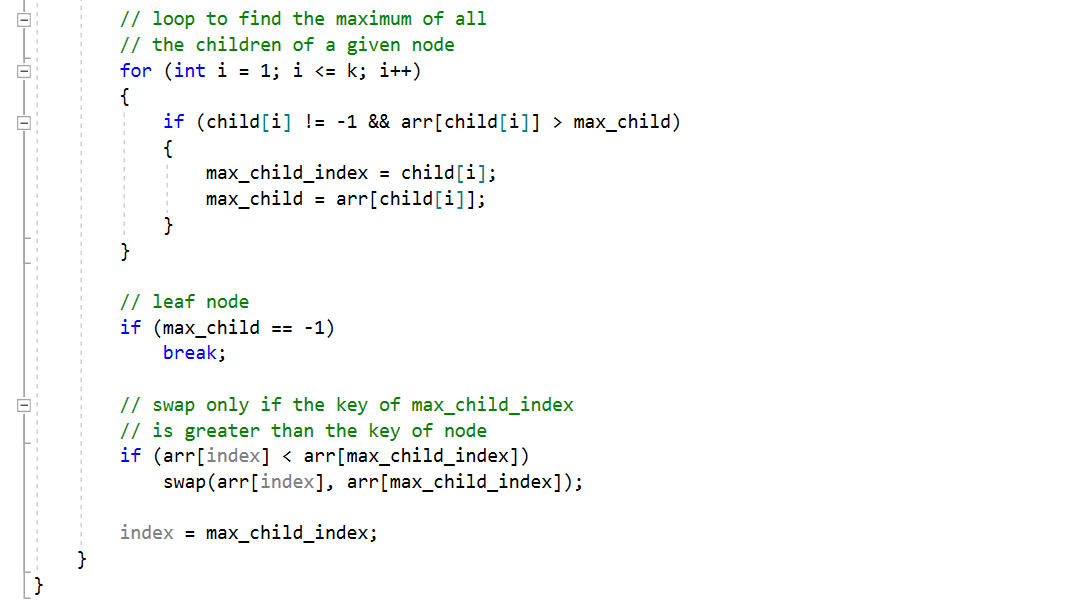
Тому що всі вузли після цього є листовими вузлами, які тривіально задовольняють властивості піраміди, оскільки вони не мають дітей, а отже, вже є коренями K-арної max піраміди.



* restoreDown () (або maxHeapify): Використовується для підтримання властивостей піраміди.

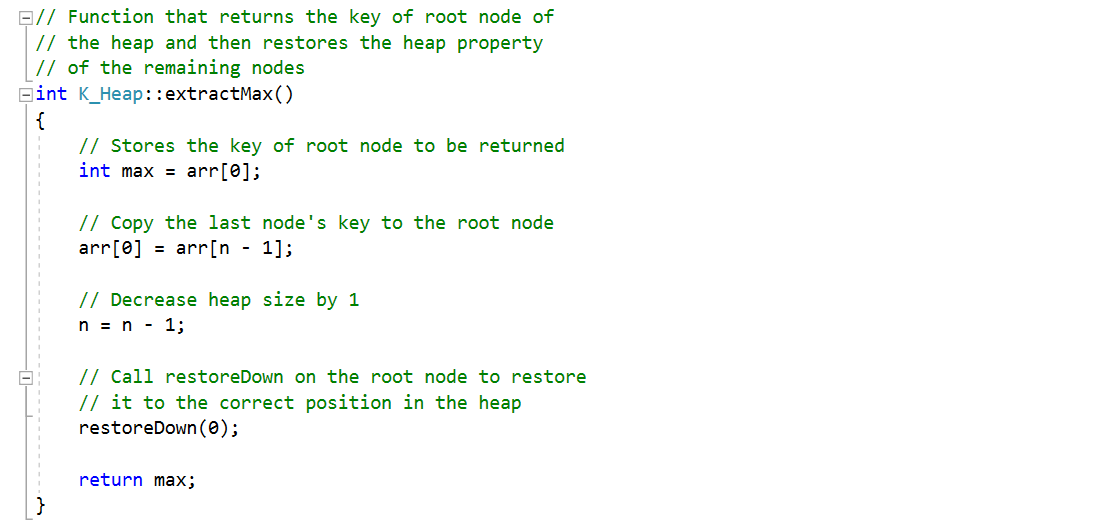
Вона запускає цикл, де знаходить максимум всіх дітей вузла, порівнює його зі власним значенням вузла і міняє місцями з максимальною дитиною, якщо max (значення всіх дітей)> (значення в вузлі). Вона повторює цей крок, поки вузол не відновлюється в початкове положення в піраміді.



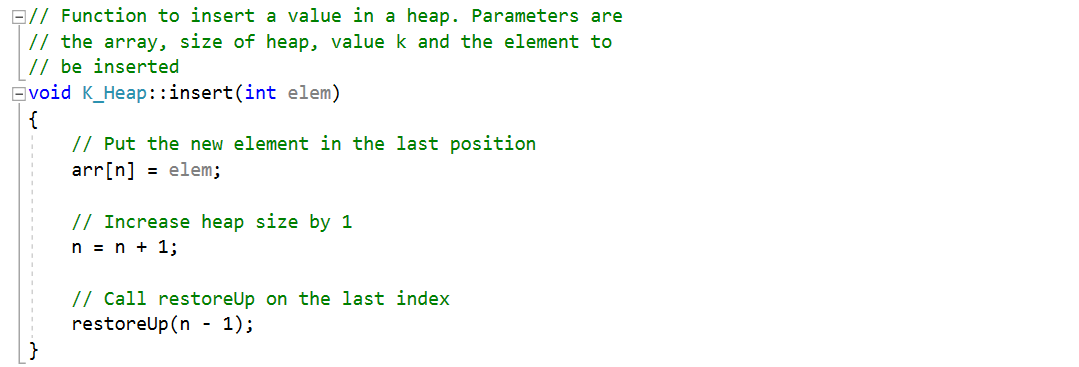


* extractMax (): Добування значення кореневого вузла.

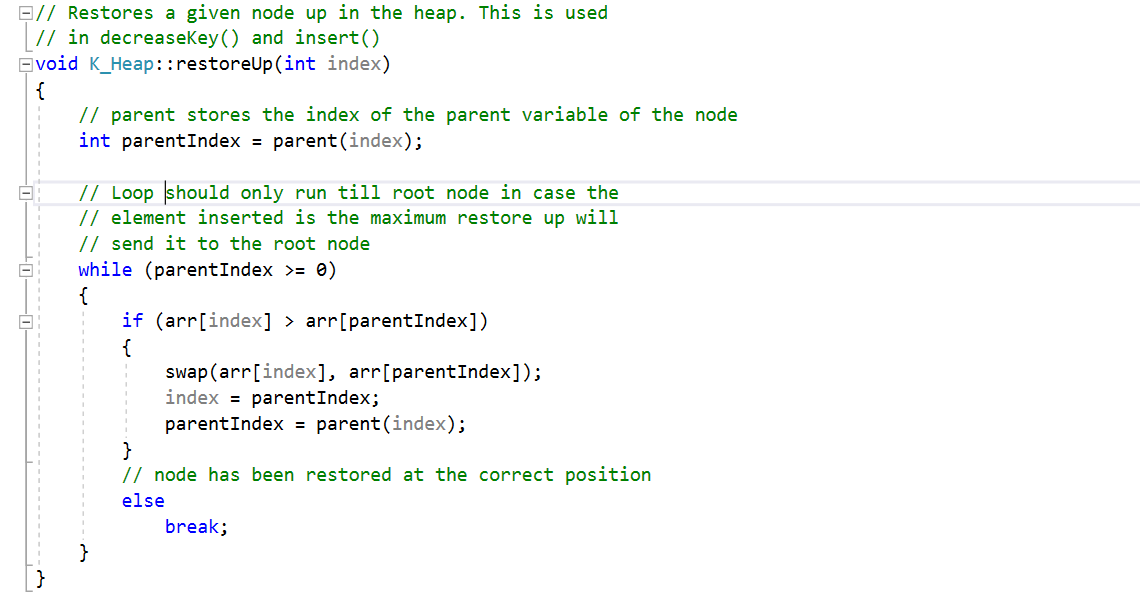
К-арна макс піраміда зберігає найбільший елемент у своєму корені. Метод повертає кореневий вузол, копіює останній вузол до першого, викликає відновлення на першому вузлі, зберігаючи таким чином властивість піраміди.



* insert (): Вставка вузла в піраміду. Це можна досягти шляхом вставки вузла в останню позиції та виклику restoreUp () на заданому індексі для відновлення правильному положенні вузла в піраміді.

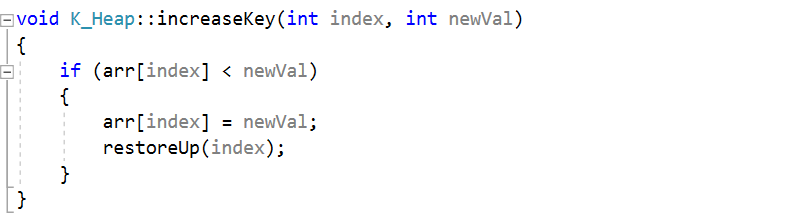


* RestoreUp() ітеративно порівнює значення даного вузла з його батьком, оскільки в максимальній піраміді батько завжди більший або рівний своїм вузлам для дітей, вузол міняється значеннями з батьком лише тоді, коли його ключ більше, ніж батько.



* increaseKey(): збільшення ключа вузла за заданим індексом

Записує нове значення в заданий вузол, якщо воно більше даного та викликає restoreUp(), щоб підтримати основну властивість піраміди.



1. Використані джерела:

<https://www.geeksforgeeks.org/binary-heap/>

[**https://cs.stackexchange.com/questions/6078/a-d-ary-heap-problem-from-clrs**](https://cs.stackexchange.com/questions/6078/a-d-ary-heap-problem-from-clrs)

**https://www.geeksforgeeks.org/k-ary-heap/**