# Kopce

Adam Kraś 325177 Krzysztof Król 325178

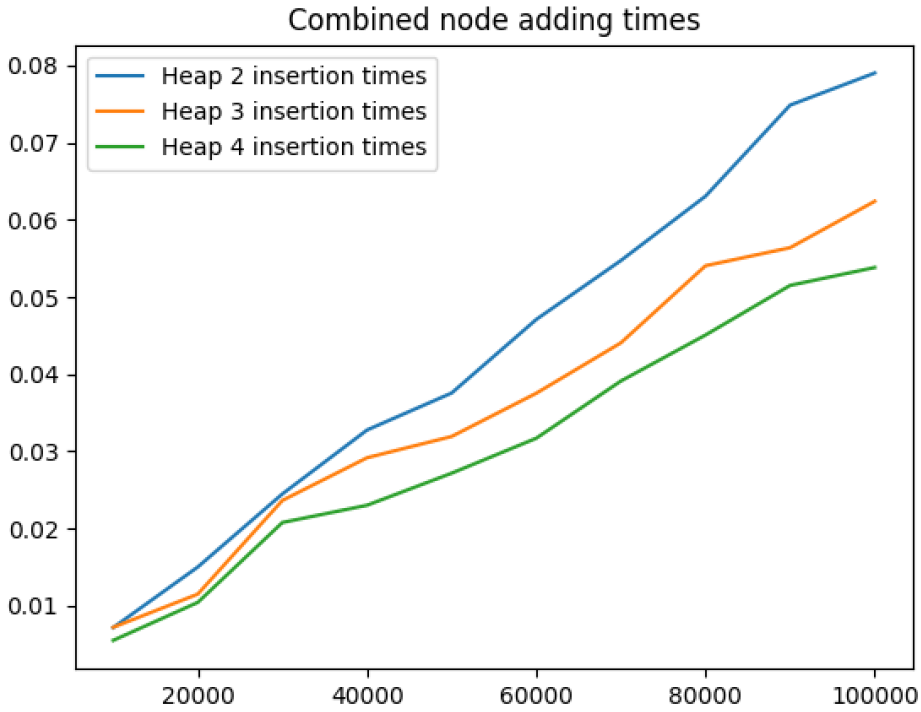
## Podział pracy

W tym zadaniu laboratoryjnym podzieliliśmy się pracą wedle schematu

1. Adam Kraś
    a. Dodawanie węzłów w kopcu
    b. Wyświetlanie kopca na ekranie
2. Krzysztof Król
    a. Usuwanie węzłów w kopcu
    b. Funkcja main() – mierzenie czasów i rysowanie wykresów

## Tworzenie kopca

W celu przetestowania jak szybko trwa tworzenie kopca 2-arnego, 3-arnego i 4-arnego stworzyliśmy listy o losowych n liczbach (n =10000, 20000, …, 100000) i dla każdej z tych wartości n mierzyliśmy ile czasu zajmowało zapełnienie kopca. Tak wygląda wykres przedstawiający wyniki naszych pomiarów:

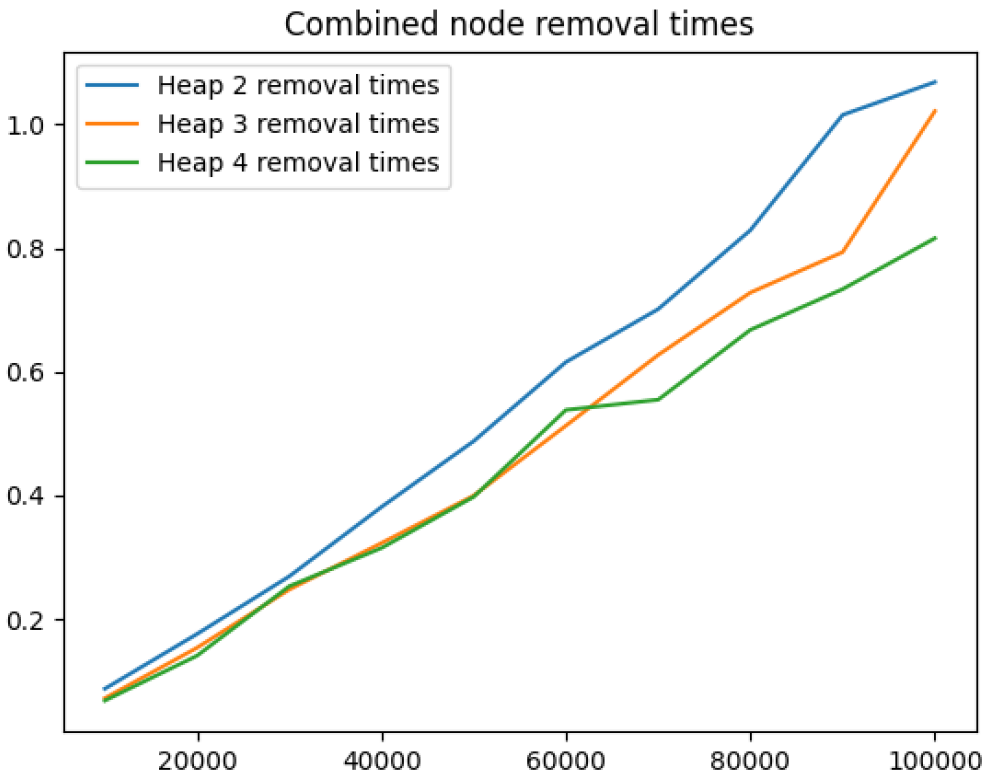



W celu uzyskania wykresu jak powyżej wystarczy uruchomić moduł main.py.

Jak więc widać dla kopca n-arnego im większe jest n, tym mniej trwa tworzenie kopca. Jest to związane z tym, iż dla większego n jest mniej poziomów w kopcu, czyli tym samym podczas procesu dodawania jest mniej porównań, zatem tym samym zajmuje to mniej czasu.

## Usuwanie korzenia kopca

W celu przetestowania jak szybko trwa usuwanie szczytu kopca 2-arnego, 3-arnego i 4-arnego stworzyliśmy listy o losowych n liczbach (n =10000, 20000, …, 100000) i dla każdej z tych wartości n tworzyliśmy kopce (takie same, jak te, które w podpunkcie wyżej służyły do mierzenia czasu tworzenia), a następnie mierzyliśmy ile czasu zajmuje usunięcie całego kopca usuwając po kolei kolejne wierzchołki kopca ustalane podczas usuwania poprzedniego. Tak wygląda wykres przedstawiający wyniki naszych pomiarów:




W celu uzyskania wykresu jak powyżej wystarczy uruchomić moduł main.py.

Jak więc widać dla kopca n-arnego im większe jest n, tym mniej trwa usuwanie wierzchołka kopca. Jest to spowodowane tym, iż pomimo faktu, że porównań wartości synów jest więcej (kiedy używamy metody heapify_down()), to z uwagi na mniejszą ilość poziomów i tak zajmuje to mniej czasu, co widać na wykresie powyżej.

## Rysowanie kopca

Do rysowania kopca napisaliśmy metodę klasy Heap, print_heap(). Działa ona uniwersalnie dla kopca n-arnego niezależnie od wartości n. Węzły będące na tym poziomie są napisane w tej samej linijce i przy pomocy strzałki wskazują na wartość swojego rodzica. Wygląda to tak:

```
2-ary heap:
9
    8 —> 9 5 —> 9
        6 —> 8 7 —> 8 1 —> 5 4 —> 5
            0 —> 6 3 —> 6 2 —> 7
Removing element from heap
8
    7 —> 8 5 —> 8
        6 —> 7 2 —> 7 1 —> 5 4 —> 5
            0 —> 6 3 —> 6
Removing element from heap
7
    6 —> 7 5 —> 7
        3 —> 6 2 —> 6 1 —> 5 4 —> 5
            0 —> 3
3-ary heap:
9
    5 —> 9 8 —> 9 2 —> 9
        0 —> 5 3 —> 5 4 —> 5 1 —> 8 6 —> 8 7 —> 8
Removing element from heap
8
    5 —> 8 7 —> 8 2 —> 8
        0 —> 5 3 —> 5 4 —> 5 1 —> 7 6 —> 7
Removing element from heap
7
    5 —> 7 6 —> 7 2 —> 7
        0 —> 5 3 —> 5 4 —> 5 1 —> 6
4-ary heap:
9
    7 —> 9 8 —> 9 2 —> 9 3 —> 9
        0 —> 7 4 —> 7 5 —> 7 6 —> 7 1 —> 8
Removing element from heap
8
    7 —> 8 1 —> 8 2 —> 8 3 —> 8
        0 —> 7 4 —> 7 5 —> 7 6 —> 7
Removing element from heap
7
    6 —> 7 1 —> 7 2 —> 7 3 —> 7
        0 —> 6 4 —> 6 5 —> 6
```

Aby zobaczyć przykładowe wypisanie kopców do terminala należy uruchomić moduł heap.py.