LABORATORIUM 3, Kopce

Zespół: Zuzanna Filipkowska, Aleksandra Sypuła

Środowisko: Visual Studio Code

Link do repozytorium: https://gitlab-stud.elka.pw.edu.pl/zfilipko/aisdi\_2021\_104.git

**Podział pracy:**

Zuzanna Filipkowska:

* wyświetlanie kopców na ekranie
* generowanie pliku tekstowego z czasami tworzenia kopców
* tworzenie, zapisywanie wykresów

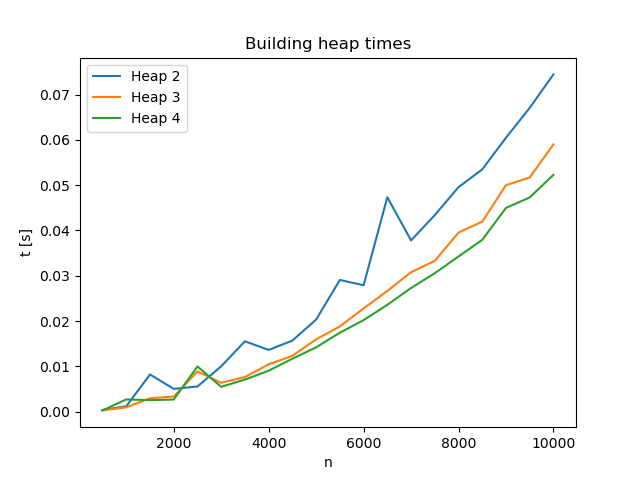
Aleksandra Sypuła:

* wstawianie elementu do kopca, tworzenie kopca
* testy jednostkowe
* generowanie czasu przetwarzania operacji tworzenia dla kopców

**Pliki:**

* Build\_heap.py – funkcje pozwalające utworzyć kopce o każdej arności; kopce tworzone są poprzez dodawanie kolejnych elementów z podanej listy do kopca, a następnie przywracanie właności kopca
* Test\_Build\_heap.py – testy do funkcji tworzących kopce i wstawiania elementu do kopca
* Heap.py – plik umożliwiający generowanie wykresów z pomiarami czasu tworzenia kopców
* show.py – funkcja wyświetlająca na ekranie kopce o danej arności
* printed\_heaps.txt – plik wygenerowany z wykorzystaniem show.py obrazujący kopce dla każdej arności
* input.txt – plik tekstowy z listami danymi jako parametry funkcji do utworzenia kopców
* wyniki\_heap\_k2/k3/k4.txt – trzy pliki tekstowe: po jednym dla każdej arności z czasami tworzenia kopców dla zwiększającej się liczby elementów
* Heap\_2/3/4\_times.png – trzy wykresy dla poszczególnych arności, wykresy czasu tworzenia kopców w zależności od liczby elementów
* Heap\_all\_times.png – wykres zbiórczy, umożliwiający porównanie czasów tworzenia kopców o różnej arności

Wykres przedstawiajacy zależność czasu tworzenia kopca o podanej arności od liczby elementów:



**Wnioski:**

Na złożoność operacji tworzenia kopca składają się: przejście po całej tablicy wejściowej i dodawanie do nowej listy-kopca po jednym elemencie oraz po dodaniu nowego elementu wywoływanie funkcji przywracającej własności kopca. Funkcja przywracająca własności kopca ma złożoność O(logkn), gdzie k – arność kopca, n – liczba elementów w danym kopcu, a złożoność operacji dodawania kolejno elementów do kopca: O(n), n – liczba elementów w liście. Ostatecznie utworzenie kopca podanym sposobem daje złożoność O(nlogkn).

Na wykresie możemy zauważyć, że zgodnie z założeniami czasy dla kopca o k=4 są najmniejsze, po nim dla k=3 i najwolniejsze dla kopca binarnego; log2n > log3n > log4n. Otrzymane wykresy są zbliżone kształtem do funkcji nlogkn, z uwzględnieniem, że w trakcie wykonywania obliczeń, nasz komputer był obciążony jednocześnie innymi procesami, które mogły rzutować na ostateczne wyniki czasów.

Wyświetlanie kopców na ekranie (zawartość pliku printed\_heaps.txt). Brak węzła został przedstawiony jako ” - -”.



**Testy**

Testy zostały napisane przy użyciu framework’u pytest.

Uruchomienie: pytest Test\_Build\_heap.py