

Algorithmik Wintersemester 2023/24

Heaps

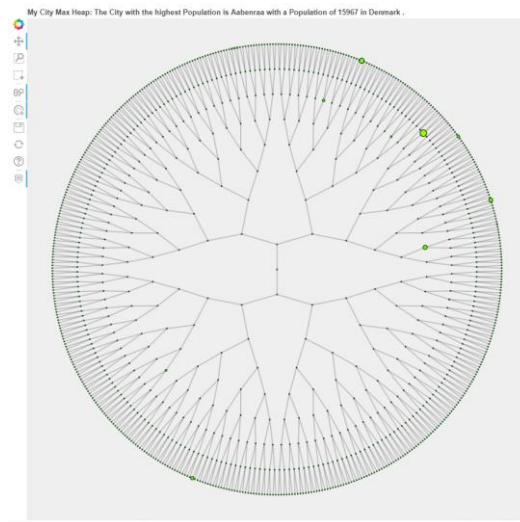
Fakultät für Informatik und Ingenieurwissenschaften - Institut für Informatik

Prof. Dr. Daniel Gaida und Tim Yago Nordhoff

Aufgabenstellung

1. Implementieren Sie für eine gegebene unsortierte Liste von Städten innerhalb des gegebenen Python-Projekts einen Max Heap. An der Wurzel Ihres Heap soll daher die Stadt mit der höchsten Einwohnerzahl zu finden sein.

- Nach Abschluss der Aufgabe sollten Sie Ihren Heap in dieser Visualisierung wiedererkennen mit der Stadt mit der höchsten Einwohnerzahl im Zentrum.



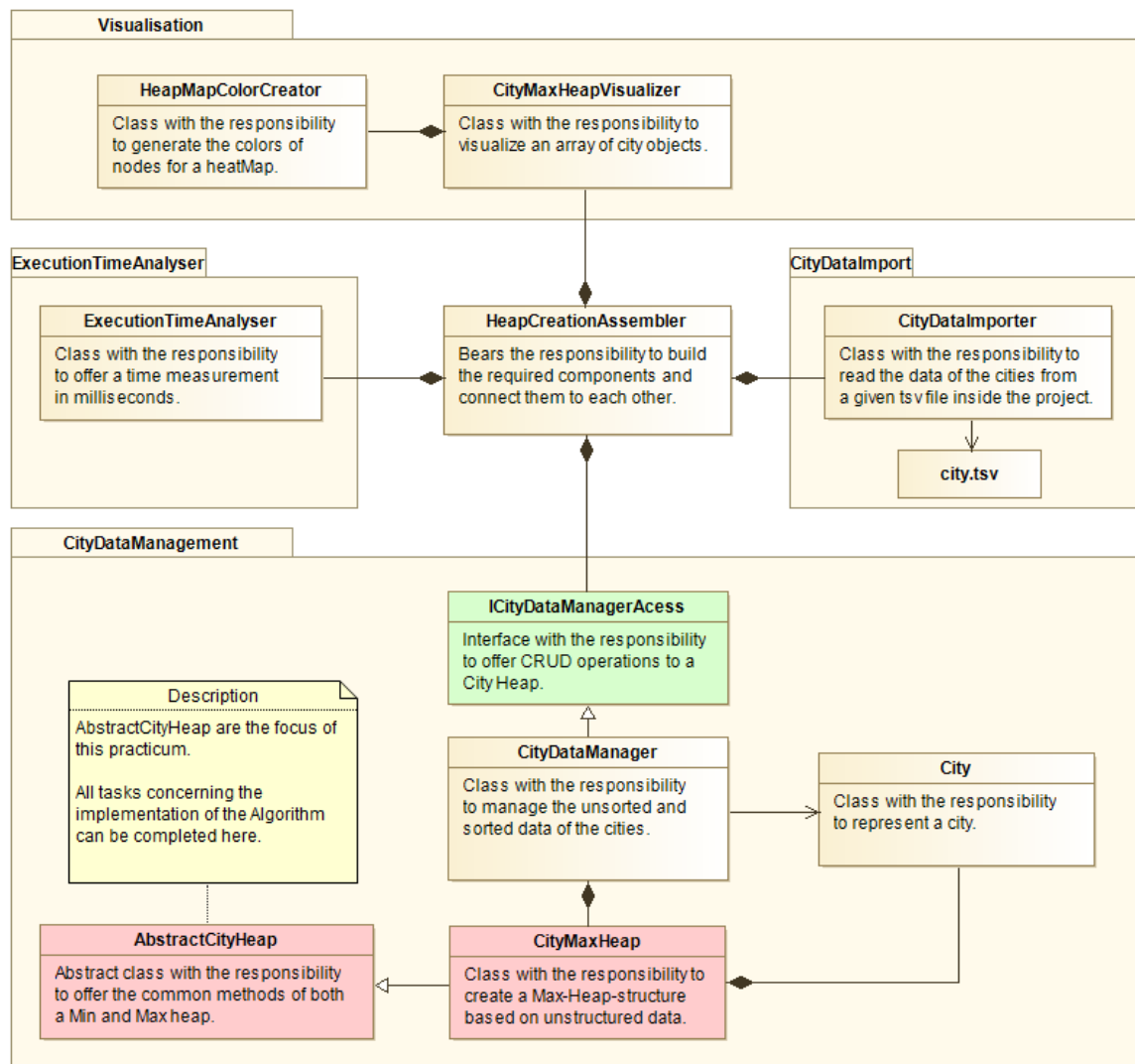
- Ihnen sollten die korrekten Ausführungszeiten sowohl für eine iterative und rekursive Implementierung als auch über Floyd's Heap Construction Algorithmus angezeigt werden.
 - Die Stadt Hobbiton sollte Ihrem Heap hinzugefügt worden sein und als größte Stadt sollte sie direkt wieder entfernt werden.
2. Innerhalb des Terminals erkennen Sie die Ausführungszeit einer Sortierung über Pythons TimSort. Vergleichen Sie diesen mit der Ausführungszeit Ihrer Heap Implementierung. Bei der Ausführung können Sie erkennen, dass die benötigte Zeit zum Erzeugen der Heap Struktur mehr Zeit benötigt als eine Sortierung über TimSort.

Begründen Sie, unter welchen Bedingungen der Einsatz eines Heaps dennoch performanter ist. Tipp: Denken Sie z.B. an ein Ticket System mit Prioritäten basierend auf vergangener Zeit und Kritikalität.

Gegebene Struktur

Der Startpunkt des Projekts ist die Klasse „HeapCreationAssembler“. Diese Klasse baut die notwendigen Komponenten auf. Sie unterteilt sich in folgende Pakete (Python Packages):

- CityDataImport
- Visualization
- ExecutionTimeAnalyser
- CityDataManagement



CityDataImport:

Dieses Package trägt die Verantwortlichkeit, die Daten des *.tsv Files in ein für die Applikation passendes Format zu übertragen. (Parser)

Visualization:

Dieses Package trägt die Verantwortlichkeit, Ihren Heap zu visualisieren. Wenn keine geeignete Heap Struktur gefunden wird, werden stattdessen die unsortierten Daten der Städte verwendet. Zu diesem Zweck werden die Bibliotheken [Bokeh](#) und [NetworkX](#) eingesetzt.

ExecutionTimeAnalyser:

Dieses Package trägt die Verantwortlichkeit, Operationen zur Zeitmessung bereitzustellen. Alle drei Packages sind bereits implementiert. Ihr Fokus liegt daher rein auf dem folgenden Package.

CityDataManagement:

Dieses Package trägt die Verantwortlichkeit, Ihren Max Heap aufzubauen und Operationen gegenüber anderen Modulen bereitzustellen. Dies betrifft bspw. das Anlegen eines neuen Heaps, die Rückgabe der Stadt mit der höchsten Bevölkerung oder das Einfügen und das Entfernen von Städten. Auch diese Schnittstellen Methoden sind bereits implementiert.

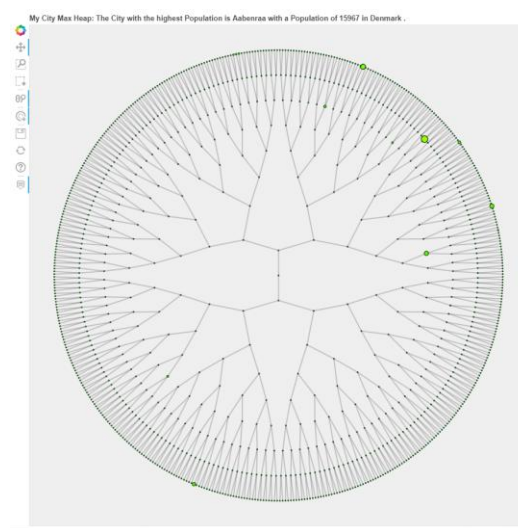
Ihr Fokus liegt dadurch auf den Klassen **AbstractCityHeap** und **CityMaxHeap**.

AbstractCityHeap offeriert dabei alle Methoden die sowohl ein Max Heap als auch ein Min Heap gemeinsam haben. Und unterteilt sich in zwei Bereiche.

- 1) Abstrakte Methoden die sowohl in einem Max Heap als auch einem Min Heap vorhanden sind, jedoch jeweils unterschiedliche Implementierungen erfordern. Diese sollten in der Klasse **CityMaxHeap** implementiert werden.
- 2) Gemeinsame Methoden, die in einem Max Heap als auch einem Min Heap identisch sind. Ob diese auch identisch sind, ist natürlich von Ihrer jeweiligen Implementierung abhängig. Für dieses Praktikum konzentrieren wir uns jedoch auf die Implementierung eines Max Heaps. Ob die von Ihnen implementierten Methoden auch kompatibel zu einem Min Heap sind, wird nicht explizit geprüft. Sehen Sie diese Aufteilung daher als Hilfestellung zum Verständnis der Gemeinsamkeiten und Unterschiede von Heaps.

Ihr Heap soll dabei Objekte sortieren. Hierzu finden Sie bereits die Klasse City, welche über drei Attribute verfügt: name, country und population. Wir suchen nach der Stadt mit der höchsten Population.

Nach dem Ausführen der Applikation sollte Ihnen folgende Visualisierung im Browser angezeigt werden:



Dies entspricht der Visualisierung der ersten 1023 Städte innerhalb der unsortierten Daten. Möchten Sie sich alle Städte anzeigen lassen, kommentieren Sie Zeile 57 aus und entfernen den Kommentar in Zeile 58 der Klasse HeapCreationAssembler.

```

55 # Visualisation
56 dataToVisualize : List[City] = self.cityDataManager.getMaxHeapAsList()
57 #amountOfNodesToCreate = 1023
58 amountOfNodesToCreate = len(cityData) #all cities, use this for science at the price of performance ;)
59 self.visualizeHeap(dataToVisualize, amountOfNodesToCreate, cityData)

```

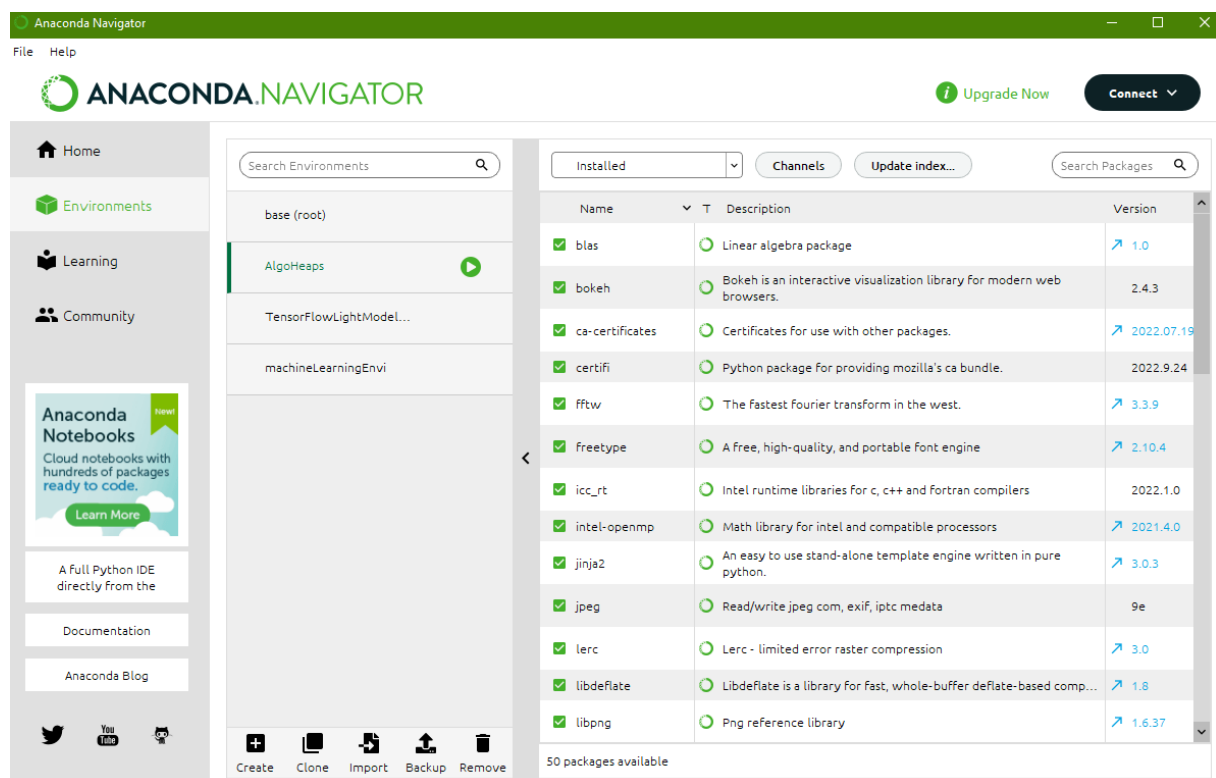
Hinweis

Die durch uns vorgegebene Struktur dient als Hilfestellung.

Sollten Sie stattdessen Ihre eigene Heapstruktur implementieren wollen, dürfen Sie davon abweichen. Wichtig ist, dass die Schnittstellenmethoden in `ICityDataManagerAccess` korrekt bedient werden, um eine Kompatibilität mit den anderen Modulen der Applikation zu gewährleisten.

Einsatz von Anaconda

Für die lokale Arbeit mit Python auf Ihrem Rechner empfehlen wir den Einsatz von Anaconda mit der Erweiterung Anaconda-Navigator. Dies offeriert Ihnen die Möglichkeit, virtuelle Umgebungen anzulegen und somit Python Packages zu isolieren. Dies vereinfacht Ihnen die Verwaltung von Python Projekten gegenüber einer einfachen lokalen Installation über pip.



Zudem finden Sie im Projekt die Datei „algo_heaps_enviroment.yaml“ diese können Sie Importieren und somit eine Virtual Enviroment aufsetzen, die bereits alle Pakete enthält, die Sie für dieses Praktikum benötigen.

How to: Connect VSC with Anaconda VR:

Als IDE empfehlen wir Visual Studio Code mit den Erweiterungen: Pylance, Python und GitLens.

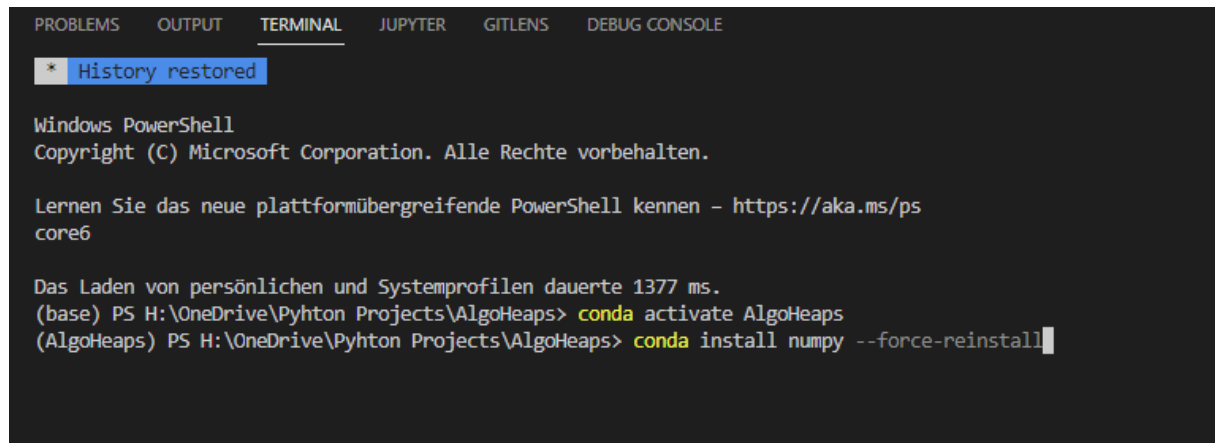
Troubleshooting:

Während der Vorbereitung dieser Applikation für dieses Praktikum kam es vereinzelt zu Fehlermeldungen über nicht gefundene Module innerhalb der importierten Python Pakete. Also Fehlermeldungen, die nicht auf dieses Pythonprojekt zurückzuführen sind. Dies ist bedingt durch Abhängigkeiten der verwendeten Pakete untereinander. Sollte ein solcher Fehler bei Ihnen auftreten, erzwingen Sie eine saubere Neuinstallation der Pakete numpy, pillow und scipy:

```
conda install numpy --force-reinstall
```

```
conda install pillow --force-reinstall
```

```
conda install scipy --force-reinstall
```



```
PROBLEMS  OUTPUT  TERMINAL  JUPYTER  GITLENS  DEBUG CONSOLE
* History restored

Windows PowerShell
Copyright (C) Microsoft Corporation. Alle Rechte vorbehalten.

Lernen Sie das neue plattformübergreifende PowerShell kennen - https://aka.ms/ps
core6

Das Laden von persönlichen und Systemprofilen dauerte 1377 ms.
(base) PS H:\OneDrive\Pyhton Projects\AlgoHeaps> conda activate AlgoHeaps
(AlgoHeaps) PS H:\OneDrive\Pyhton Projects\AlgoHeaps> conda install numpy --force-reinstall
```