Konzeption der Unterrichtseinheit

Einführung in die Heap-Datenstruktur und das darauf basierende Sortierverfahren Heapsort

Mentorierte Arbeit – Fachdidaktik Informatik FS 23

Manuel Wettstein

22. Juni 2023

1 Analyse des Unterrichtsstoffes

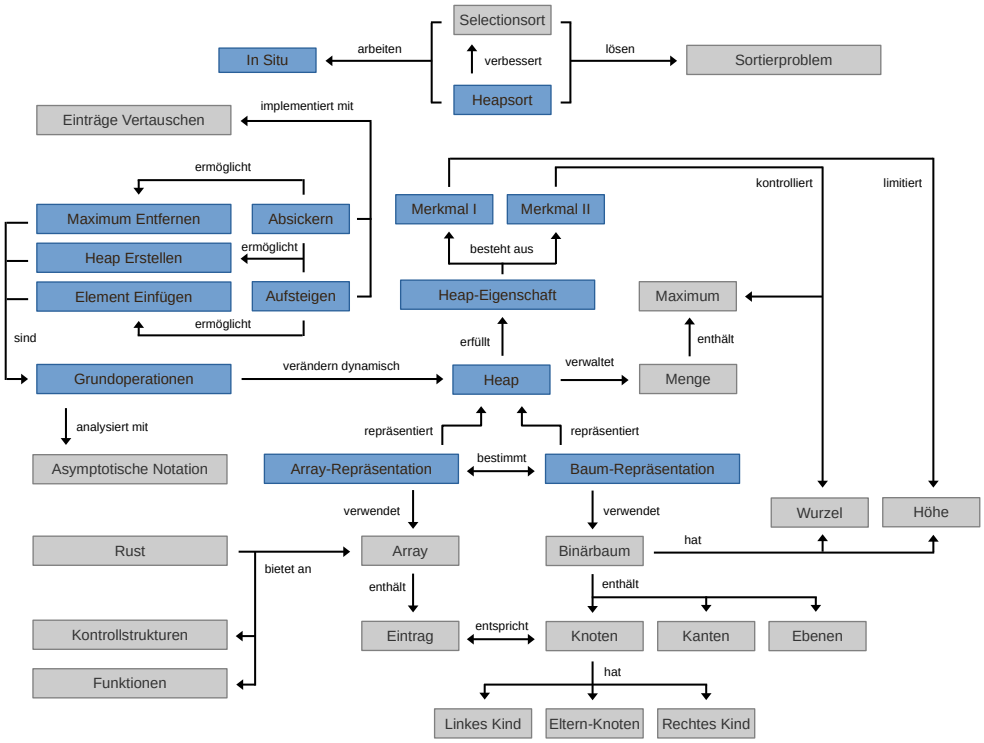
Mit Hilfe einer Concept Map, die sich sowohl auf der nächsten Seite als auch in einem beigelegten separaten Dokument befindet, wollen wir hier kurz das vorausgesetzte Vorwissen der Studierenden und den neu zu vermittelnden Unterrichtsstoff analysieren. Zur besseren Übersicht erscheinen in der graphischen Darstellung die bereits bekannten Begriffe jeweils in grauen Rechtecken und die neuen Begriffe in blauen Rechtecken.

Der zentrale neue Begriff ist derjenige des *Heaps*, der auch in der Mitte der Graphik zu finden ist. Im ersten Abschnitt dieser Unterrichtseinheit soll es darum gehen, diesen zentralen Begriff erst einmal genauer zu definieren, und zwar mit Hilfe der aus zwei unterschiedlichen *Merkmalen* bestehenden *Heap-Eigenschaft*. Weiter wird untersucht, wie diese beiden Merkmale die *Höhe* und das *Maximum* (respektive die *Wurzel*) eines Heaps beeinflussen. Wir setzen voraus, dass der Begriff des *Binärbaums* und die zugehörigen Grundbegriffe wie zum Beispiel *Knoten* und *Kanten* den Studierenden bereits bekannt sind.

Der zweite Abschnitt dient dazu, die kompakte *Array-Repräsentation* eines Heaps einzuführen und mit der *Baum-Repräsentation* des vorhergehenden Abschnittes zu kontrastieren. Insbesondere wird erarbeitet, wie durch einfaches Rechnen mit Indizes die entsprechenden Kind- und Eltern-Knoten gefunden werden können. An dieser Stelle sollen die Studierenden zum ersten Mal auch etwas programmieren, und zwar eine Methode für die automatische Überprüfung der Heap-Eigenschaft eines mittels Array-Repräsentation gegebenen Heaps.

Im dritten Abschnitt sollen die Studierenden zuerst in einer geleiteten Aufgabe die Vorgänge des *Aufsteigens* und des *Absickerns* eines Knotens selbständig entdecken. Mit Hilfe dieser beiden Vorgänge werden dann die drei *Grundoperationen* eines Heaps implementiert und deren Laufzeiten analysiert. Dabei handelt es sich um (1) das Einfügen eines zusätzlichen Elements, (2) das Entfernen des aktuellen Maximums und (3) das Erstellen eines neuen Heaps. Letztere Operation kann sowohl durch Aufsteigen als auch durch Absickern realisiert werden. Die effizientere dieser zwei Möglichkeiten wird mit Hilfe der Konstruktion eines Worst-Case-Inputs bestimmt.

Basierend auf den erarbeiteten Methoden aus den ersten drei Abschnitten wird im vierten Abschnitt eine einfache Implementierung des Sortierverfahrens *Heapsort* angegeben und damit die Laufzeit des als bekannt vorausgesetzten Sortierverfahrens *Selectionsort* verbessert. Die wichtige Eigenschaft, dass neben der Eingabesequenz kein zusätzlicher Speicherplatz verwendet wird, also dass der Algorithmus *in situ* arbeitet, wird kurz thematisiert.



2 Leitidee

Datenstrukturen im Allgemeinen sind unerlässliche und zentrale Objekte in der Informatik und beim Programmieren. Für Problemstellungen, welche den schnellen Zugriff auf gewisse Datensätze erfordern, wird oft erst durch die geschickte Wahl einer adäquaten Datenstruktur das Design eines effizienten Algorithmus ermöglicht. Ein tief greifendes Verstehen, das über die Kenntnis von einfachen linearen Strukturen wie Arrays und verlinkte Listen hinausgeht, ist deshalb für die Studierenden einer Einführungsvorlesung für Informatik an der Hochschule unentbehrlich.

Beim Heap handelt es sich vermutlich um eine der schlichtesten, aber trotzdem auch um eine der nützlichsten solcher Datenstrukturen, denn er erlaubt die wohl einfachste Implementierung einer so genannten Prioritätenliste. Gleichzeitig kann der Heap auf natürliche Weise zu anderen grundlegenden Konzepten aus der Informatik in Bezug gesetzt werden, wie das Sortierproblem oder Binärbäume. Der Heap und das darauf basierende Sortierverfahren eignen sich deshalb besonders gut für einen ersten ernsthaften Schritt in die umfangreiche Welt der Datenstrukturen.

3 Dispositionsziele

1. Die Studierenden sind sich der Bedeutung von Datenstrukturen in der Informatik bewusst und sind deswegen motiviert, sich weiter mit diesem Thema auseinanderzusetzen.
2. Die Studierenden verstehen, dass die geschickte Wahl einer Datenstruktur einen signifikanten Einfluss auf die Effizienz eines sonst identischen Algorithmus haben kann.
3. Wann immer beim Programmieren ein Datensatz angefordert wird, überlegen sich die Studierenden, ob dieser Schritt mit einer passenden Datenstruktur beschleunigt werden kann.
4. Die Studierenden kennen die Funktionsweise des Heaps im Detail und verwenden diese Datenstruktur in Anwendungen auf natürliche Weise als Prioritätenliste.
5. Die Studierenden kennen die Funktionsweise von Heapsort im Detail und sind sich der Komplexität des Sortierproblems bezüglich Laufzeit sowie Speicherverbrauch bewusst.

4 Operationalisierte Lernziele

1. Die Studierenden entscheiden bei kleinen Beispielen von Heaps mit maximal vier Ebenen und mit konkreten Zahlen, ob die Heap-Eigenschaft erfüllt ist oder nicht.
2. Die Studierenden enumerieren alle Heaps von einer sehr kleinen gegebenen Menge M von maximal fünf natürlichen Zahlen.
3. Die Studierenden erklären, in welcher Weise die Heap-Eigenschaft die Höhe und das Maximum eines Heaps beeinflusst.
4. Die Studierenden übersetzen kleine konkrete Beispiele der Array-Repräsentation eines Heaps mit maximal 15 Einträgen in die Baum-Repräsentation und umgekehrt.
5. Die Studierenden berechnen für den gegebenen Index eines Eintrags der Array-Repräsentation die Indizes seiner Eltern- und Kind-Knoten.
6. Die Studierenden führen die Vorgänge des Aufsteigens und des Absickerns an kleinen konkreten Beispielen von Heaps mit maximal vier Ebenen vor.
7. Die Studierenden erklären, wie folgende drei Grundoperationen mit Hilfe von Aufsteigen und Absickern realisiert werden: (1) Einfügen eines zusätzlichen Elements, (2) Entfernen des aktuellen Maximums sowie (3) Erstellen eines neuen Heaps.
8. Die Studierenden konstruieren eine Liste mit n natürlichen Zahlen, für welche beide Varianten für die Erstellung eines neuen Heaps möglichst viele Vertauschungen vornehmen müssen.
9. Die Studierenden erklären auf intuitive Weise, welche dieser beiden Varianten für die Erstellung eines neuen Heaps im schlimmsten Fall schneller zum Erfolg führt.
10. Die Studierenden erklären, wie das bekannte Sortierverfahren Selectionsort mit Hilfe eines Heaps ohne zusätzlichen Speicherbedarf zur Laufzeit $O(n \log n)$ beschleunigt werden kann.