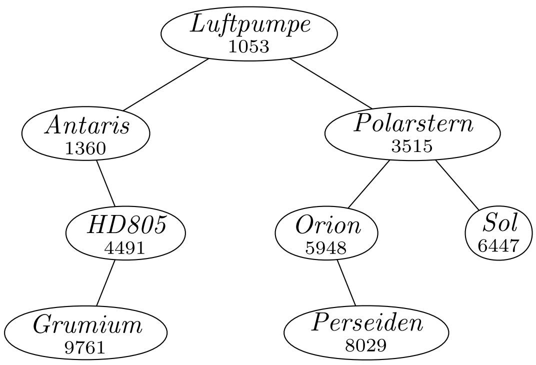
Datenstrukturen und Algorithmen, SS 2024
Prof. Dr. P. Rossmanith
D. Cloerkes, M. Gehnen, D. Mock, N. Speetzen

Datum: 03.05.2024, 14:00

Übungsblatt 03

Aufgabe T7

Wir haben diesen Treap:



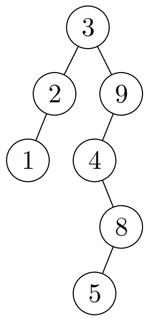
a) Fügen Sie den Schlüssel *Taurus* mit der Priorität 8719 ein.

b) Löschen Sie danach die *Luftpumpe*.

c) Fügen Sie jetzt die *Luftpumpe* wieder ein. Verwenden Sie die Priorität 2854.

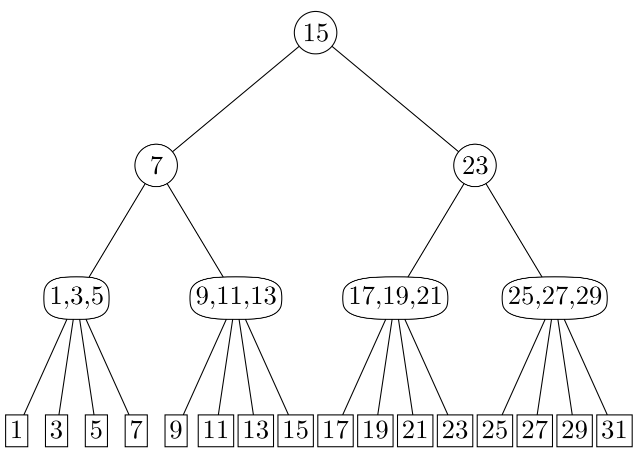
Aufgabe T8

Wir betrachten folgenden Splay-Baum:



Was passiert, wenn wir in diesen Baum nach dem Schlüssel 10 suchen, dann nach 1 suchen, dann 6 einfügen und schließlich 8 löschen?

Aufgabe T9



Wie sieht dieser $(2, 4)$-Baum jeweils nach dem Löschen der Schlüssel 9, 11, 13 und 17 aus? Zeichnen Sie den Baum nach jeder Löschoperation.

Aufgabe T10

Einst schickte Frau Mutter, mit eiligen Worten,
klein Timmi zum Markt: „Es gibt neue Waren!
Strukturen für Daten! Verschiedenste Sorten!
Bring mir eine Queue—doch müssen wir sparen.

Drei Groschen für eine, das sollte schon passen."
So lief Timmi fort, den Marktplatz voraus
doch kaum war er dort, konnt' er sich nicht lassen
„Zwei Groschen reichen doch sicherlich aus!"

Von Gier besiegt und mit Eis in der Hand
erschrak Timmi heftig, als er sich besann
„Drei Groschen die Queue" las er dort am Stand—
er zerbrach sich den Kopf und das Eis zerann.

Mit zwei Stacks im Rucksack kehrt er schließlich heim
—wegen reichlicher Ernte bekam er sie beide—
Doch scholt' ihn die Mutter „Das kann doch nicht sein!
Stacks gehen verkehrt, weshalb ich sie meide!"

„Eine Queue, liebe Mutter, bau ich drumherum:
Enqueue-en werd' ich nur in den ersten der beid'
Und will ich dequeue-en, so füll ich sie um.
Amortisiert wird das klappen—in konstanter Zeit."

Hilf klein Timmi! Seine simulierte Queue funktioniert wie folgt:

```
int dequeue() {
  if(right.isEmpty()) {
    while(!left.isEmpty())
      right.push(left.pop());
  }
  return right.pop();
}
```

enqueue legt also nur Elemente auf den linken Stack, *dequeue* dreht dann bei Bedarf den Inhalt des linken Stacks um: Es entfernt sukzessive alle Elemente und legt sie auf den rechten Stack. Solange der rechte Stack jetzt noch Elemente enthält, nimmt *dequeue* sie schlicht von diesem.

Kann Timmi auf diese Weise eine Queue *effizient* simulieren?

Nehmen Sie an, dass die simulierte Queue leer ist und dann n beliebige legale Operationen auf ihr ausgeführt werden (das bedeutet, dass *dequeue* nur aufgerufen wird, wenn die Queue nicht leer ist). Zeigen Sie mittels amortisierter Analyse, dass die Gesamtlaufzeit für alle Operationen $O(n)$ ist. Was ist eine geeignete Potentialfunktion? Wie ändert sie sich bei den beiden Operationen? Wie groß ist sie am Anfang und am Ende?

Aufgabe H8 (10 Punkte)
Betrachten Sie den Treap aus Aufgabe T7 (in der Aufgabenstellung, vor den Tutoraufgaben). Führen Sie folgende Operationen nacheinander auf ihm aus und zeichnen Sie jeweils die dabei entstehenden Treaps.

a) Fügen Sie *Aldebaran* mit Priorität 8719 ein.

b) Löschen Sie *Antaris*.

c) Fügen Sie *Dagobah* mit Priorität 2854 ein.

d) Löschen Sie *HD805*.

Aufgabe H9 (10 Punkte)
Entwerfen Sie einen effizienten Algorithmus für *split*(*key*), der für einen Splaybaum zwei Splaybäume ausgibt, einen mit allen Knoten kleiner *key* und einen mit allen Knoten größer *key*. Was können Sie über die amortisierte Laufzeit sagen, wenn wir folgende Operationen in beliebiger Reihenfolge und Kombination durchführen: Einfügen, Löschen, Suchen und *split*?

Aufgabe H10 (10 Punkte)

In der Vorlesung wurde für (a, b)-Bäume verlangt, dass $b \geq 2a - 1$ gilt. Daher gibt es $(2, 3)$-Bäume, aber keine $(3, 4)$-Bäume.

Warum gilt diese Einschränkung? Was für Probleme könnte es beispielsweise mit $(3, 4)$-Bäumen geben?

Aufgabe H11 (10 Punkte)

An einem malerischen Ort südlich des Informatikzentrums gibt es zwei Pizzerien, die ein gemeinsames Problem haben: Braune Pilze, die scheinbar zufällig immer an den unpassendsten Stellen aus dem Boden sprießen, und sicherlich nicht gut für das Geschäft sind.

Der Einfachheit halber nehmen wir an, dass die Restaurants lange Flure seien, wo es zwischen Eingang (links) und Küche (rechts) nur eine Dimension gibt. Außerdem nehmen wir an, dass nie an zwei Stellen gleichzeitig Pilze erscheinen. Ein Pilz erscheint also erst, nachdem der vorherige Pilz durch ein kompliziertes Verfahren („draufspringen“) entfernt wurde. Ignorieren der Pilze ist allerdings keine Option.

- Bei Francesco und Andrea geht stets derjenige zu dem Pilz, der am nähesten an ihm steht. Der andere bleibt stehen. Falls beide gleich weit weg sind, wird zufällig entschieden.
- Wenn Luigi und Mario in ihrer Pizzeria einen Pilz zwischen sich sehen, so laufen beide gleich schnell von beiden Seiten auf ihn zu, bis einer ihn erreicht hat. Ist der Pilz außerhalb der beiden, so geht nur der näheste zu dem Pilz und der weiter entfernte bleibt stehen.

Nehmen wir an, dass alle vier lauffaul sind: um welchen Faktor a muss das Personal der beiden Pizzerien mit den jeweiligen Strategien schlimmstenfalls mehr laufen, als wenn sie bereits die am Anfang des Tages gewusst hätten wo die Pilze wann auftauchen werden und einen perfekten Plan gemacht hätten (dies nennt man auch Approximationsgüte)?

Hinweis 1: Bei der Luigis und Marios Strategie bietet sich eine amortisierte Analyse an. Als Potentialfunktion der Strategie nach Pilz i eignet sich $\Phi(S_i) = 2 \cdot dist(L_i, L_i^{plan}) + 2 \cdot dist(R_i, R_i^{plan}) + dist(L_i, R_i)$, wobei L_i bzw. R_i die Positionen von der linken bzw. rechten Person in der Sitution nach Pilz i sei, und L_i^{plan} (bzw. R_i^{plan}) die Positionen nach einem perfekten Plan sind. $dist(a, b)$ sei die Distanz zwischen a und b.

Hinweis 2: Die Potentialfunktion ist am Anfang nicht null, sondern hat die Größe $dist(L_0, R_0)$. Damit können wir die Approximationsgüte mit einer additiven Konstante bestimmen (Mario und Luigi laufen also maximal a-mal so viel wie im perfekten Plan, plus einmalig zusätzlich bis zu $dist(L_0, R_0)$).

Hinweis 3: Für die Analyse nehmen wir an, dass sich immer erst der Kellner der perfekten Lösung bewegt, danach dann in unserer Strategie: wie häufig bei amortisierter Analyse bietet es sich an, eine Fallunterscheidung zu betrachten. Wie ändert sich die Potentialfunktion,

- wenn sich erst jemand in der perfekten Lösung bewegt?
- wenn der Pilz außerhalb von Mario und Luigi erscheint, und einer dorthin geht?
- wenn der Pilz innerhalb der beiden erscheint, und sich beide bewegen?

Nur wenn der sich jemand in perfekten Lösung sich bewegt, dürfen wir „Guthaben“in der Potentialfunktion aufbauen. Wenn wir danach jemanden nach unserer Strategie bewegen, müssen wir dies vom Guthaben bezahlen.