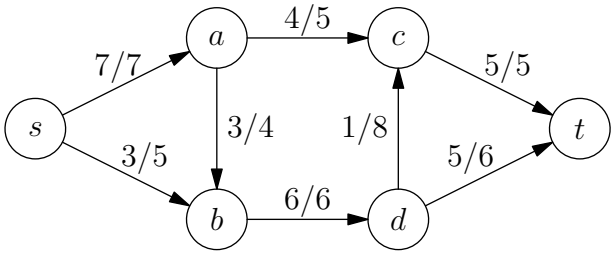
RWTH Aachen **SoSe 2011**
Lehrgebiet Theoretische Informatik
Emmes—Langer—Reidl—Rossmanith 12. September 2011

Klausur zur Vorlesung Datenstrukturen und Algorithmen

Aufgabe 1 (10 Punkte)

Gegeben sei das rechts abgebildete Flußnetzwerk zusammen mit einem Fluß.

1) Konstruieren Sie das zugehörige Residualnetzwerk.
2) Geben Sie einen augmentierenden Pfad an.
3) Wie sieht der Fluß aus, nachdem sie ihn entlang Ihres Pfades augmentierten?



Aufgabe 2 (10 Punkte)

In der Vorlesung wurden Splay-Bäume vorgestellt, deren Arbeitsweise auf den ersten Blick etwas umständlich wirkt. Wir betrachten in dieser Aufgabe eine einfachere Version der Splay-Bäume: Die übliche Splay-Operation wird durch folgende naive Variante ersetzt: Wird auf einen Knoten die Splay-Operation angewendet, dann wird einfach wiederholt ein *zig* oder *zag* auf ihn angewendet, bis der Knoten zur Wurzel wird.

Konstruieren Sie eine Folge von $O(n)$ Operationen (Einfügen, Suchen und/oder Löschen), so daß ein anfangs leerer Splay-Baum mit dieser Einschränkung für diese $\Omega(n^2)$ Zeit benötigt. Dadurch zeigen wir, daß die vielleicht ungewöhnlich erscheinenden *zig-zig*s und *zag-zag*s wohl doch sinnvoll sind.

Aufgabe 3 (10 Punkte)

Konstruieren Sie einen optimalen Suchbaum, der die Schlüssel 1, 2, 3 und 4 mit den Zugriffswahrscheinlichkeiten 1/6, 1/4, 1/4 und 1/3 enthält.

Verwenden Sie das Verfahren aus der Vorlesung. Konstruieren Sie insbesondere die Tabelle für die e_{ij}-Werte. Was ist die Bedeutung von e_{ij}?

Aufgabe 4 (10 Punkte)

Gegeben sei ein unsortiertes Array von nicht-leeren Strings. Entwerfen Sie einen Algorithmus, der den am häufigsten in diesem Array vorkommenden String bestimmt – sollten dies mehrere sein, so reicht die Ausgabe einer dieser Strings.

Die Gesamtlänge der Strings sei mit n bezeichnet. Stellen Sie sicher, daß der von Ihnen angegebene Algorithmus höchstens $O(n)$ Schritte benötigt und maximal $O(n)$ zusätzlichen Platz beansprucht (d.h. verwendete Variablen und Arrays dürfen insgesamt höchstens $O(n)$ Speicher belegen).

Begründen Sie die Laufzeit Ihrer Lösung!

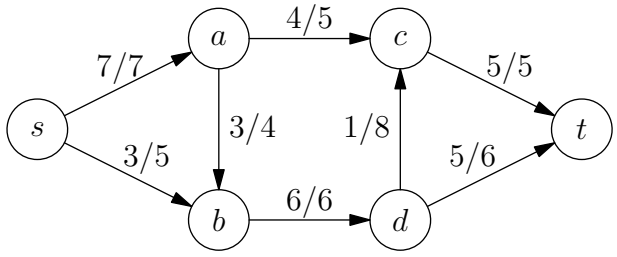
RWTH Aachen **SoSe 2011**
Lehrgebiet Theoretische Informatik
Emmes—Langer—Reidl—Rossmanith 12. September 2011

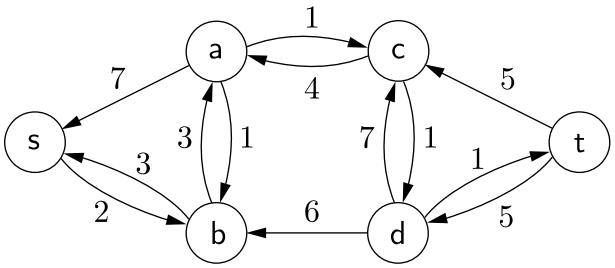
Klausur zur Vorlesung Datenstrukturen und Algorithmen

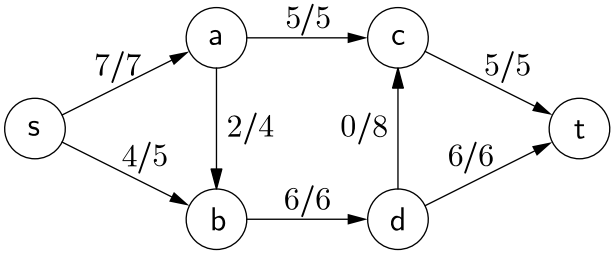
Aufgabe 1 (10 Punkte)

Der einzige augmentierende Pfad in diesem Flußnetzwerk lautet (s, b, a, c, d, t) und erlaubt eine vergrößerung des Flusses um eine Einheit (die Kante a, c etwa hat nur noch eine Kapazität von einer Einheit übrig).

Nach der Augmentierung ist der Fluß (rechts abgebildet) maximal—dies kann z.B. einfach anhand des Schnittes $V - t, t$ gesehen werden, denn dieser enthält nur kritische Kanten.







Aufgabe 2 (10 Punkte)

Sei $n \in \mathbf{N}$ beliebig. Wir fügen in einen zu Beginn leeren Splaytree zunächst die Zahlen $1, \ldots, n$ in aufsteigender Reihenfolge ein. Danach löschen wir, ebenfalls in aufsteigender Reihenfolge, ebendiese Zahlen $1, \ldots, n$. Dieses sind offenbar $2n$ Operationen.

Durch Induktion über n sieht man leicht ein, daß das Ergebnis der Einfügeoperationen ein geordneter Pfad ist, genauer: ein Baum ohne rechte Kinder, dessen einziges Blatt die 1 ist, deren Vaterknoten die 2 usw., und in dessen Wurzel die n steht. (Nehmen wir dazu an, es wurden bereits die Elemente $1, \ldots, n-1$ auf die beschriebene Weise eingefügt, so daß nach Induktionsvoraussetzung die $n-1$ nun in der Wurzel steht. Wird nun n eingefügt, wird die als rechtes Kind der $n-1$ eingefügt und dann sofort gesplayt, steht danach also in der Wurzel.)

Wenn wir nun aus einem solchen geordneten Pfad das Blatt löschen, wird dieses zunächst in die Wurzel gesplayt. Hat der Pfad Länge k, benötigt dieses also $\Omega(k)$ Schritte. Nach dem Löschen ist der Splay-Tree immer noch ein geordneter Pfad mit dem kleinsten Element, nun der 2, ganz unten. Durch jede Löschoperation verändert sich die Höhe des Baums also immer nur um eins, wodurch das Löschen aller Elemente $\Omega(\sum_{i=1}^{n} n - i) = \Omega(n^2)$ Schritte benötigt.

Aufgabe 3 (10 Punkte)

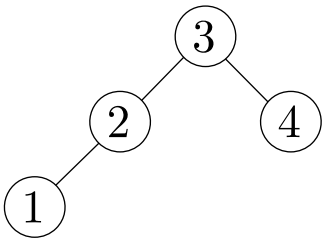
In den folgenden Tabellen sind die Werte, der Lesbarkeit habler, mit einem Faktor von 12 skaliert (ein Eintrage 5 ist also streng genommen 5/12). Die Zugriffswarscheinlichkeiten der Schlüssel 1, 2, 3 und 4 lauteten 1/6, 1/4, 1/4 und 1/3 (skaliert also 2,3,3 und 4).

$12 \cdot w_{i,j}$	1	2	3	4
1	2	5	8	12
2	0	3	6	10
3	0	0	3	7
4	0	0	0	4

$12 \cdot e_{i,j}$	1	2	3	4
1	$2^{(1)}$	$7^{(2)}$	$13^{(2)}$	$23^{(3)}$
2	0	$3^{(2)}$	$9^{(2,3)}$	$17^{(3)}$
3	0	0	$3^{(3)}$	$10^{(4)}$
4	0	0	0	$4^{(4)}$

Der Wert $e_{i,j}$ ist dabei gerade die erwartete Anzahl an Vergleichen in einem optimalen Suchbaum, der die Werte i bis j enthält. Die Berechnung eines Eintrages $e_{i,j}$ erfolgt über die Rekursionsformel

$$e_{i,j} = \min_{i \le r \le j} (e_{i,r-1} + e_{r+1,j}) + w_{i,j}$$



wobei $e_{i,j} = 0$ gilt falls $j < i$. Da die Werte $e_{i,i}$ bekannt sind —sie sind gerade $w_{i,i}$—kann die Berechnung per dynamischer Programmierung von der Diagonale der Tabelle aus geschehen. Dabei merken wir uns den Wert r, welcher das aktuelle $e_{i,j}$ minimiert (in der Tabelle in Klammern angeben) um den Bauim anschließend zu rekonstruieren: r ist nämlich gerade die Wurzel des optimalen Suchbaums, der die Werte $i, \ldots, j$ enthält. Somit lesen wir die die Wurzel des gesamten Baum am Eintrag für $e_{1,4}$ als 3 ab. Der rechte Teilbaum kann nur den Schlüssel 4 enthalten, zur Kontrolle prüfen wir trotzdem den Eintrag für $e_{4,4}$—natürlich lautet dieser 4. Die Wurzel des linken Teilbaums von 3 ist schließlich am Eintrag $e_{1,2}$ zu erkennen: es ist die 2. Insgesamt ergibt sich dann der oben abgebildete Suchbaum.

Aufgabe 4 (10 Punkte)

Wir verwenden einen modifizierten Trie, in welchem wir zu jedem String zusätzlich eine Zählervariable speichern, die angibt, wie oft dieser String im Array vorkommt.

Der Algorithmus geht das Array der Strings von vorne bis hinten durch und fügt jeden String nacheinander in den Trie ein. Ist er dort bereits vorhanden, erhöhen wir die zugehörige Zählervariable um eins; wenn nicht, setzen wir diese Zählervariable auf den Wert 1. Gleichzeitig führen wir in zwei zusätzlichen Variablen Buch über den bisher am häufigsten vorgekommenen String und seine Häufigkeit. Wenn das Array abgearbeitet ist, wird dieser häufigste String ausgegeben.

Zur Laufzeit: Das Einfügen eines Strings w in einen Trie benötigt $O(|w|)$ Operationen. Da die Strings nicht leer sind, dauert die Einfügephase also $O(\sum_w |w|) = O(n)$ Schritte. Der Trie selbst benötigt ebenso nur $O(n)$ Speicher, die zusätzlichen Zählervariablen pro Knoten des Tries daher ebenso.