Berufsakademie Sachsen

Staatliche Studienakademie Leipzig

Minimalgerüste mit Kruskal

Eingereicht von: Matthias Thurow

CS13-2

Sebastian Wolff

CS13-1

Inhalt

Verbale Erklärung

Es gab verschiedene Möglichkeiten das Problem zu lösen, wir haben uns für eine objektorientierte Variante entschieden. Dazu haben wir Eclipse IDE verwendet um in Java zu programmieren. Als Objekte sehen wir die Knoten, welche man zu einem Baum zusammenknüpfen kann, und die Kanten welche die Verbindungen zwischen den Knoten darstellen.

Als erstes haben wir die Objekte „Knoten“ und „Kante“ erzeugt mit den nötigen Eigenschaften und Methoden. Ein Knoten besitzt einen übergeordneten Knoten (Parent), es sei denn der Knoten selbst ist der Wurzelknoten. Ein Knoten kennt auch seinen Wurzelknoten. Als Eigenschaften besitzt ein Knoten zwei Flags die gesetzt werden können. Diese beschreiben ob der Knoten schon besucht wurde und ob der Knoten selber ein Wurzelknoten ist.

Die Kanten beinhalten lediglich die Informationen welche Knoten im Ursprungsgraphen miteinander verbunden sind und dem jeweiligem Gewicht der Kante.

Als erstes wird die Datei eingelesen und die unterschiedlichen Knoten werden erstellt. Anschließend werden diese in eine Liste geschrieben. Danach werden die Kanten des Ursprungsgraphen erstellt indem die Datei nochmals ausgelesen wird. Die erstellten Kanten werden in eine Prioritätsqueue eingeordnet, sodass am Anfang der Queue die Kante mit dem geringsten Gewicht zu finden ist. Das Auslesen der Datei ist somit abgeschlossen und es wird die erstellte Queue abgearbeitet.

Solange Elemente in der Queue vorhanden sind wird das erste Element der Queue entnommen und untersucht. Wenn beide Knoten noch nicht besucht worden sind, dann wird der Erste Knoten als Parent des Zweiten Knoten eingetragen. Zusätzlich wird angegeben, dass der erste Knoten nun ein Wurzelknoten ist. Diese Information wird auch bei beiden Knoten gespeichert. Es wird bei beiden Knoten auch der Eintrag vorgenommen, dass sie besucht worden sind. Wenn beide Knoten schon besucht worden sind, dann wird geprüft ob beide denselben Wurzelknoten besitzen. Ist dies der Fall, dann wird der Eintrag verworfen und nichts an den Knoten verändert, da ansonsten ein Zyklus entstehen würde.

Haben jedoch beide unterschiedliche Wurzelknoten dann wird der Erste Knoten Parent des Zweiten Knoten. Außerdem wird auch die Wurzel des Zweiten Knoten zu einem normalen Knoten indem das Flag „isRoot“ entfernt wurde. Alle Knoten des zweiten Baumes bekommen als Wurzelknoten nun denselben Wurzelknoten des ersten Baumes.

Die letzte Möglichkeit wäre, dass ein Knoten bereits besucht, und der andere nicht besucht wurde. In diesem Fall wird der bereits besuchte Knoten dem nicht besuchten Knoten als Parent eingetragen.

Diese Fälle werden bei jeder Iteration durch die Queue betrachtet. Da alle Knoten auch in einer extra Liste gespeichert worden sind kann man einen Output für den User generieren. Nachdem ein Element aus der Queue untersucht wurde wird über die Liste iteriert und es werden alle Knoten betrachtet die noch nicht besucht worden sind. Diese werden dann dem Benutzer später angezeigt. So kann der Benutzer die Entstehung des generierten Baumes nachvollziehen.

Komplexität

Das Einlesen der Knoten und Kanten hat die Komplexität E, da je nach Anzahl der Kanten die entsprechende Zahl der Zeilen gelesen werden muss. Das Iterieren über die Elemente hat die gleiche Komplexität E, danach muss dann aber jedes Mal die Queue neu organisiert werden, was die Komplexität erhöht: E Schleifendurchläufe * log(E) zur Reorganisation. Die Kontrolle, ob eine Kante hinzugefügt werden kann ist auch nicht aufwändiger, als das Reorganisieren der Queue, sodass die endgültige Komplexität Θ(E log(E)) ist.

Aufteilung der Programmierung

Wer welchen Teil programmiert hat kann an den Kommentaren im Quelltext erkannt werden. Immer, wenn dort ein Name aufgeführt wird, wurde der Teil bis zum nächsten Namen von demjenigen programmiert. Diese Unterteilung war nur in der Launcher-Klasse notwendig

Programmausgabe

-- Es wurde die Datei Daten9A.txt verwendet. –

added edge 9 - 20 Gewicht: 1
Noch nicht besucht: 1 10 11 12 13 14 15 16 17 18 19 2 21 22 23 24 25 26 27 28 29 3 30 4 5 6 7 8
Rootknoten: 9

added edge 19 - 24 Gewicht: 1
Noch nicht besucht: 1 10 11 12 13 14 15 16 17 18 2 21 22 23 25 26 27 28 29 3 30 4 5 6 7 8
Rootknoten: 19 9

added edge 14 - 7 Gewicht: 1
Noch nicht besucht: 1 10 11 12 13 15 16 17 18 2 21 22 23 25 26 27 28 29 3 30 4 5 6 8
Rootknoten: 14 19 9

added edge 25 - 24 Gewicht: 1
Noch nicht besucht: 1 10 11 12 13 15 16 17 18 2 21 22 23 26 27 28 29 3 30 4 5 6 8
Rootknoten: 14 19 9

added edge 15 - 19 Gewicht: 1
Noch nicht besucht: 1 10 11 12 13 16 17 18 2 21 22 23 26 27 28 29 3 30 4 5 6 8
Rootknoten: 14 19 9

added edge 6 - 19 Gewicht: 2
Noch nicht besucht: 1 10 11 12 13 16 17 18 2 21 22 23 26 27 28 29 3 30 4 5 8
Rootknoten: 14 19 9

added edge 29 - 22 Gewicht: 2
Noch nicht besucht: 1 10 11 12 13 16 17 18 2 21 23 26 27 28 3 30 4 5 8
Rootknoten: 14 19 29 9

added edge 29 - 3 Gewicht: 3
Noch nicht besucht: 1 10 11 12 13 16 17 18 2 21 23 26 27 28 30 4 5 8
Rootknoten: 14 19 29 9

added edge 18 - 15 Gewicht: 4
Noch nicht besucht: 1 10 11 12 13 16 17 2 21 23 26 27 28 30 4 5 8
Rootknoten: 14 19 29 9

added edge 3 - 9 Gewicht: 5
Noch nicht besucht: 1 10 11 12 13 16 17 2 21 23 26 27 28 30 4 5 8
Rootknoten: 14 19 29

added edge 13 - 5 Gewicht: 5
Noch nicht besucht: 1 10 11 12 16 17 2 21 23 26 27 28 30 4 8
Rootknoten: 13 14 19 29

added edge 4 - 30 Gewicht: 7
Noch nicht besucht: 1 10 11 12 16 17 2 21 23 26 27 28 8
Rootknoten: 13 14 19 29 4

added edge 4 - 23 Gewicht: 7
Noch nicht besucht: 1 10 11 12 16 17 2 21 26 27 28 8
Rootknoten: 13 14 19 29 4

added edge 9 - 28 Gewicht: 10
Noch nicht besucht: 1 10 11 12 16 17 2 21 26 27 8
Rootknoten: 13 14 19 29 4

added edge 23 - 1 Gewicht: 10
Noch nicht besucht: 10 11 12 16 17 2 21 26 27 8
Rootknoten: 13 14 19 29 4

added edge 23 - 15 Gewicht: 11
Noch nicht besucht: 10 11 12 16 17 2 21 26 27 8
Rootknoten: 13 14 29 4

added edge 16 - 9 Gewicht: 11
Noch nicht besucht: 10 11 12 17 2 21 26 27 8
Rootknoten: 13 14 29 4

added edge 8 - 21 Gewicht: 14
Noch nicht besucht: 10 11 12 17 2 26 27
Rootknoten: 13 14 29 4 8

added edge 12 - 20 Gewicht: 15
Noch nicht besucht: 10 11 17 2 26 27
Rootknoten: 13 14 29 4 8

added edge 10 - 20 Gewicht: 19
Noch nicht besucht: 11 17 2 26 27
Rootknoten: 13 14 29 4 8

added edge 12 - 18 Gewicht: 19
Noch nicht besucht: 11 17 2 26 27
Rootknoten: 13 14 29 8

added edge 7 - 11 Gewicht: 20
Noch nicht besucht: 17 2 26 27
Rootknoten: 13 14 29 8

added edge 21 - 12 Gewicht: 21
Noch nicht besucht: 17 2 26 27
Rootknoten: 13 14 8

added edge 25 - 26 Gewicht: 21
Noch nicht besucht: 17 2 27
Rootknoten: 13 14 8

added edge 30 - 17 Gewicht: 21
Noch nicht besucht: 2 27
Rootknoten: 13 14 8

added edge 18 - 13 Gewicht: 22
Noch nicht besucht: 2 27
Rootknoten: 14 8

added edge 25 - 2 Gewicht: 22
Noch nicht besucht: 27
Rootknoten: 14 8

added edge 27 - 29 Gewicht: 22
Alle Knoten wurden besucht.
Rootknoten: 14 8

added edge 16 - 14 Gewicht: 23
Alle Knoten wurden besucht.
Rootknoten: 8

Hinweis, die "_" dienen nur zur Einrueckung der einstelligen Zahlen.

Kanonische Ausgabe:
Root: 1 10 11 12 13 14 15 16 17 18 19 2 9_ 21 22 23 24 25 26 27 28 29 3 30 4 5 6 7 8 9
Node: 1 10 11 12 13 14 15 16 17 18 19 2 20 21 22 23 24 25 26 27 28 29 3 30 4 5 6 7 8 9

Root: 1 10 11 12 13 14 15 16 17 18 19 2 9_ 21 22 23 19 25 26 27 28 29 3 30 4 5 6 7 8 9
Node: 1 10 11 12 13 14 15 16 17 18 19 2 20 21 22 23 24 25 26 27 28 29 3 30 4 5 6 7 8 9

Root: 1 10 11 12 13 14 15 16 17 18 19 2 9_ 21 22 23 19 25 26 27 28 29 3 30 4 5 6 14 8 9
Node: 1 10 11 12 13 14 15 16 17 18 19 2 20 21 22 23 24 25 26 27 28 29 3 30 4 5 6 7_ 8 9

Root: 1 10 11 12 13 14 15 16 17 18 19 2 9_ 21 22 23 19 19 26 27 28 29 3 30 4 5 6 14 8 9
Node: 1 10 11 12 13 14 15 16 17 18 19 2 20 21 22 23 24 25 26 27 28 29 3 30 4 5 6 7_ 8 9

Root: 1 10 11 12 13 14 19 16 17 18 19 2 9_ 21 22 23 19 19 26 27 28 29 3 30 4 5 6 14 8 9
Node: 1 10 11 12 13 14 15 16 17 18 19 2 20 21 22 23 24 25 26 27 28 29 3 30 4 5 6 7_ 8 9

Root: 1 10 11 12 13 14 19 16 17 18 19 2 9_ 21 22 23 19 19 26 27 28 29 3 30 4 5 19 14 8 9
Node: 1 10 11 12 13 14 15 16 17 18 19 2 20 21 22 23 24 25 26 27 28 29 3 30 4 5 6_ 7_ 8 9

Root: 1 10 11 12 13 14 19 16 17 18 19 2 9_ 21 29 23 19 19 26 27 28 29 3 30 4 5 19 14 8 9
Node: 1 10 11 12 13 14 15 16 17 18 19 2 20 21 22 23 24 25 26 27 28 29 3 30 4 5 6_ 7_ 8 9

Root: 1 10 11 12 13 14 19 16 17 18 19 2 9_ 21 29 23 19 19 26 27 28 29 29 30 4 5 19 14 8 9
Node: 1 10 11 12 13 14 15 16 17 18 19 2 20 21 22 23 24 25 26 27 28 29 3_ 30 4 5 6_ 7_ 8 9

Root: 1 10 11 12 13 14 19 16 17 19 19 2 9_ 21 29 23 19 19 26 27 28 29 29 30 4 5 19 14 8 9
Node: 1 10 11 12 13 14 15 16 17 18 19 2 20 21 22 23 24 25 26 27 28 29 3_ 30 4 5 6_ 7_ 8 9

Root: 1 10 11 12 13 14 19 16 17 19 19 2 29 21 29 23 19 19 26 27 28 29 29 30 4 5 19 14 8 29
Node: 1 10 11 12 13 14 15 16 17 18 19 2 20 21 22 23 24 25 26 27 28 29 3_ 30 4 5 6_ 7_ 8 9_

Root: 1 10 11 12 13 14 19 16 17 19 19 2 29 21 29 23 19 19 26 27 28 29 29 30 4 13 19 14 8 29
Node: 1 10 11 12 13 14 15 16 17 18 19 2 20 21 22 23 24 25 26 27 28 29 3_ 30 4 5_ 6_ 7_ 8 9_

Root: 1 10 11 12 13 14 19 16 17 19 19 2 29 21 29 23 19 19 26 27 28 29 29 4_ 4 13 19 14 8 29
Node: 1 10 11 12 13 14 15 16 17 18 19 2 20 21 22 23 24 25 26 27 28 29 3_ 30 4 5_ 6_ 7_ 8 9_

Root: 1 10 11 12 13 14 19 16 17 19 19 2 29 21 29 4_ 19 19 26 27 28 29 29 4_ 4 13 19 14 8 29
Node: 1 10 11 12 13 14 15 16 17 18 19 2 20 21 22 23 24 25 26 27 28 29 3_ 30 4 5_ 6_ 7_ 8 9_

Root: 1 10 11 12 13 14 19 16 17 19 19 2 29 21 29 4_ 19 19 26 27 29 29 29 4_ 4 13 19 14 8 29
Node: 1 10 11 12 13 14 15 16 17 18 19 2 20 21 22 23 24 25 26 27 28 29 3_ 30 4 5_ 6_ 7_ 8 9_

Root: 4 10 11 12 13 14 19 16 17 19 19 2 29 21 29 4_ 19 19 26 27 29 29 29 4_ 4 13 19 14 8 29
Node: 1 10 11 12 13 14 15 16 17 18 19 2 20 21 22 23 24 25 26 27 28 29 3_ 30 4 5_ 6_ 7_ 8 9_

Root: 4 10 11 12 13 14 4_ 16 17 4_ 4_ 2 29 21 29 4_ 4_ 4_ 26 27 29 29 29 4_ 4 13 4 14 8 29
Node: 1 10 11 12 13 14 15 16 17 18 19 2 20 21 22 23 24 25 26 27 28 29 3_ 30 4 5_ 6 7_ 8 9_

Root: 4 10 11 12 13 14 4_ 29 17 4_ 4_ 2 29 21 29 4_ 4_ 4_ 26 27 29 29 29 4_ 4 13 4 14 8 29
Node: 1 10 11 12 13 14 15 16 17 18 19 2 20 21 22 23 24 25 26 27 28 29 3_ 30 4 5_ 6 7_ 8 9_

Root: 4 10 11 12 13 14 4_ 29 17 4_ 4_ 2 29 8_ 29 4_ 4_ 4_ 26 27 29 29 29 4_ 4 13 4 14 8 29
Node: 1 10 11 12 13 14 15 16 17 18 19 2 20 21 22 23 24 25 26 27 28 29 3_ 30 4 5_ 6 7_ 8 9_

Root: 4 10 11 29 13 14 4_ 29 17 4_ 4_ 2 29 8_ 29 4_ 4_ 4_ 26 27 29 29 29 4_ 4 13 4 14 8 29
Node: 1 10 11 12 13 14 15 16 17 18 19 2 20 21 22 23 24 25 26 27 28 29 3_ 30 4 5_ 6 7_ 8 9_

Root: 4 29 11 29 13 14 4_ 29 17 4_ 4_ 2 29 8_ 29 4_ 4_ 4_ 26 27 29 29 29 4_ 4 13 4 14 8 29
Node: 1 10 11 12 13 14 15 16 17 18 19 2 20 21 22 23 24 25 26 27 28 29 3_ 30 4 5_ 6 7_ 8 9_

Root: 29 29 11 29 13 14 29 29 17 29 29 2 29 8_ 29 29 29 29 26 27 29 29 29 29 29 13 29 14 8 29
Node: 1_ 10 11 12 13 14 15 16 17 18 19 2 20 21 22 23 24 25 26 27 28 29 3_ 30 4_ 5_ 6_ 7_ 8 9_

Root: 29 29 14 29 13 14 29 29 17 29 29 2 29 8_ 29 29 29 29 26 27 29 29 29 29 29 13 29 14 8 29
Node: 1_ 10 11 12 13 14 15 16 17 18 19 2 20 21 22 23 24 25 26 27 28 29 3_ 30 4_ 5_ 6_ 7_ 8 9_

Root: 8 8_ 14 8_ 13 14 8_ 8_ 17 8_ 8_ 2 8_ 8_ 8_ 8_ 8_ 26 27 8_ 8_ 8 8_ 8 13 8 14 8 8
Node: 1 10 11 12 13 14 15 16 17 18 19 2 20 21 22 23 24 25 26 27 28 29 3 30 4 5_ 6 7_ 8 9

Root: 8 8_ 14 8_ 13 14 8_ 8_ 17 8_ 8_ 2 8_ 8_ 8_ 8_ 8_ 8_ 27 8_ 8_ 8 8_ 8 13 8 14 8 8
Node: 1 10 11 12 13 14 15 16 17 18 19 2 20 21 22 23 24 25 26 27 28 29 3 30 4 5_ 6 7_ 8 9

Root: 8 8_ 14 8_ 13 14 8_ 8_ 8_ 8_ 8_ 2 8_ 8_ 8_ 8_ 8_ 8_ 27 8_ 8_ 8 8_ 8 13 8 14 8 8
Node: 1 10 11 12 13 14 15 16 17 18 19 2 20 21 22 23 24 25 26 27 28 29 3 30 4 5_ 6 7_ 8 9

Root: 8 8_ 14 8_ 8_ 14 8_ 8_ 8_ 8_ 8_ 2 8_ 8_ 8_ 8_ 8_ 8_ 27 8_ 8_ 8 8_ 8 8 8 14 8 8
Node: 1 10 11 12 13 14 15 16 17 18 19 2 20 21 22 23 24 25 26 27 28 29 3 30 4 5 6 7_ 8 9

Root: 8 8_ 14 8_ 8_ 14 8_ 8_ 8_ 8_ 8_ 8 8_ 8_ 8_ 8_ 8_ 8_ 27 8_ 8_ 8 8_ 8 8 8 14 8 8
Node: 1 10 11 12 13 14 15 16 17 18 19 2 20 21 22 23 24 25 26 27 28 29 3 30 4 5 6 7_ 8 9

Root: 8 8_ 14 8_ 8_ 14 8_ 8_ 8_ 8_ 8_ 8 8_ 8_ 8_ 8_ 8_ 8_ 8_ 8_ 8_ 8 8_ 8 8 8 14 8 8
Node: 1 10 11 12 13 14 15 16 17 18 19 2 20 21 22 23 24 25 26 27 28 29 3 30 4 5 6 7_ 8 9

Root: 8 8_ 8_ 8_ 8_ 8_ 8_ 8_ 8_ 8_ 8_ 8 8_ 8_ 8_ 8_ 8_ 8_ 8_ 8_ 8_ 8_ 8 8_ 8 8 8 8 8 8
Node: 1 10 11 12 13 14 15 16 17 18 19 2 20 21 22 23 24 25 26 27 28 29 3 30 4 5 6 7 8 9

-- Alle Kanten des Minimalgeruests im Ueberblick --
{ 9 , 20 , 1 }
{ 19 , 24 , 1 }
{ 14 , 7 , 1 }
{ 25 , 24 , 1 }
{ 15 , 19 , 1 }
{ 6 , 19 , 2 }
{ 29 , 22 , 2 }
{ 29 , 3 , 3 }
{ 18 , 15 , 4 }
{ 3 , 9 , 5 }
{ 13 , 5 , 5 }
{ 4 , 30 , 7 }
{ 4 , 23 , 7 }
{ 9 , 28 , 10 }
{ 23 , 1 , 10 }
{ 23 , 15 , 11 }
{ 16 , 9 , 11 }
{ 8 , 21 , 14 }
{ 12 , 20 , 15 }
{ 10 , 20 , 19 }
{ 12 , 18 , 19 }
{ 7 , 11 , 20 }
{ 21 , 12 , 21 }
{ 25 , 26 , 21 }
{ 30 , 17 , 21 }
{ 18 , 13 , 22 }
{ 25 , 2 , 22 }
{ 27 , 29 , 22 }
{ 16 , 14 , 23 }