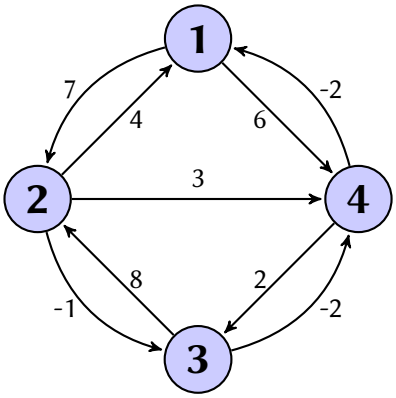
10.1 Negative Kantengewichte & Zyklen (5 Punkte)

Stellen Sie Ihr Verständnis des Bellman-Ford-Algorithmus auf die Probe.

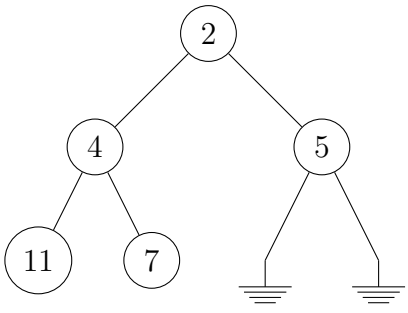
a) (3P) Gegeben sei folgender Graph. Der Startknoten s sei ein zusätzlicher Knoten, der mit jedem anderen Knoten über eine gerichtete Kante mit Gewicht 0 verbunden ist. Führen Sie händisch den Bellman-Ford-Algorithmus durch, und geben Sie für jeden Durchlauf der äußeren Schleife sowohl das Feld d der Abstände von s zu den einzelnen Knoten, als auch einen Graph (inkl. s) an, in dem Sie die durch d repräsentierten Wege markieren.



b) (2P) Eliminieren Sie die negativen Kantengewichte und zeichnen Sie den Graph (inkl. s) neu, wobei Sie an jedem Knoten das Potential vermerken und die neuen Kantengewichte verwenden.

10.2 Heaps (8+3* Punkte)

Ein binärer Heap ist ein vollständiger[1] Binärbaum, der einer lokalen Ordnung genügt. Der gespeicherte Wert eines jeden Knoten ist kleiner als die Werte seiner Kinder. Exemplarisch betrachte man folgenden Heap.



a) (2P) Geben Sie eine Strategie an, wie Sie eine solche Datenstruktur in einem eindimensionalen Array [ə'reɪ] speichern können, ohne aufwendige Pointerstrukturen benutzen zu müssen. Wie berechnet man die Indizes der Kindknoten oder den größten Index des Knoten, der noch Kinder hat?

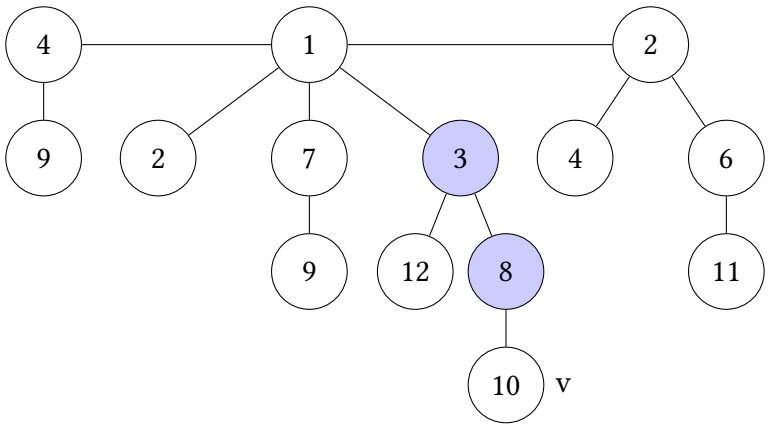
[1] falls die Anzahl der Knoten keinen vollständigen Baum erlaubt, sollen auf der untersten Ebene die Knoten von rechts fehlen, in der Grafik sind diese durch Null-Referenzen dargestellt.

b) (3P) Angenommen man habe einen gültigen Heap und entferne die Wurzel, sprich das kleinste Element. Geben Sie eine Operation an, die in logarithmischer Zeit bezogen auf die Anzahl der gespeicherten Elemente erneut einen gültigen Heap erzeugt.
c) (3P) Entwickeln Sie eine effiziente Strategie, wie Sie neue Elemente in den Heap in logarithmischer Zeit einfügen können.
d) (3*P) Für den Fall, dass am Anfang die Arrayeinträge willkürlich verteilt sind, muss erst noch die Heapeigenschaft hergestellt werden. Geben Sie eine effiziente Vorgehensweise an, die das in linearer Zeit bewältigt. Beweisen Sie die Richtigkeit Ihres Algorithmus.

10.3 Fibonacci-Heap (7 Punkte)

Gegeben sei folgender Fibonacci-Heap in vereinfachter Notation. Gehen Sie davon aus, dass der Heap, wie in der Vorlesung, Kindknoten als doppelt verkettete zykl. Liste speichert. Markierte Knoten sind blau gefärbt. Führen Sie nacheinander folgende Operationen aus und zeichnen Sie jeweils den resultierenden Heap, sowie die Zwischenstationen, die Sie für nötig erachten, um jemandem die Funktionsweise der Methoden zu erklären.

a) (2P) decreasekey(v,7) // der Wert von v soll auf 7 verringert werden
b) (5P) deletemin()



Zur Verbesserung der Übersicht entnehmen Sie Kindknoten, falls mehrere betroffen sind, immer der Reihe nach von links nach rechts und fügen Sie Knoten immer ans rechte Ende der entsprechenden Liste hinzu. Beim Konsolidieren gehen Sie ebenfalls in der Wurzelliste von links nach rechts vor und fügen zwei Teilbäume zusammen, indem der rechte Teilbaum zu einem Kind des Linken wird.