

gerichteter Distanzgraph angegeben durch Adjazenzmatrix (Graph M A P R N)

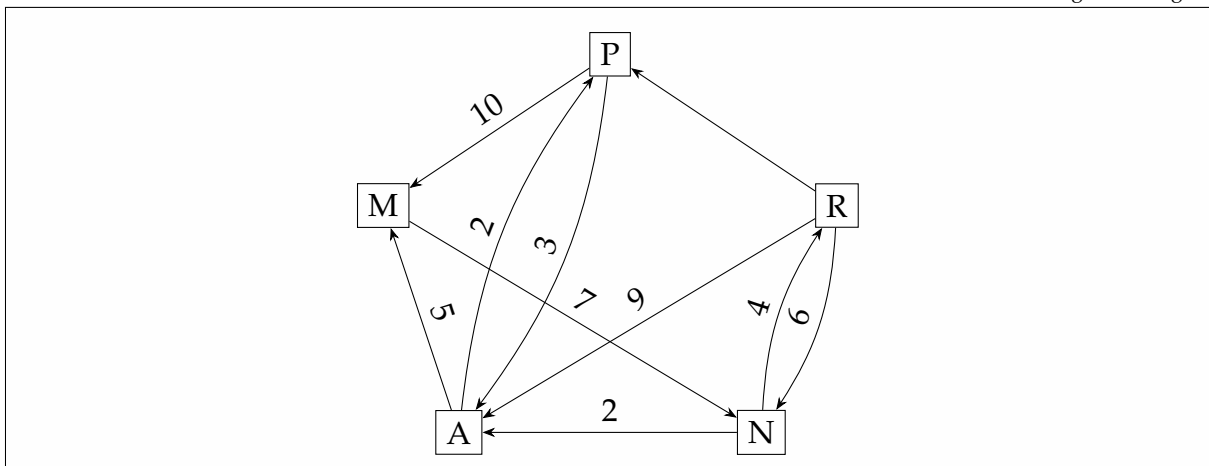
Stichwörter: Algorithmus von Dijkstra, Halde (Heap)

Ein gerichteter Distanzgraph sei durch seine Adjazenzmatrix gegeben (in einer Zeile stehen die Längen der von dem Zeilenkopf ausgehenden Wege.)

$$\begin{array}{c}
 \begin{array}{ccccc}
 & A & M & N & P & R \\
 A & * & 5 & - & 2 & - \\
 M & - & * & 7 & - & - \\
 N & 2 & - & * & - & 4 \\
 P & 3 & 10 & - & * & - \\
 R & 9 & - & 6 & 1 & *
 \end{array}
 \end{array}
 \left(\begin{array}{c}
 \\
 \\
 \\
 \\
 \\
 \end{array} \right)$$

(a) Stellen Sie den Graph in der üblichen Form dar.

Lösungsvorschlag



(b) Bestimmen Sie mit dem Algorithmus von Dijkstra ausgehend von M die kürzeste Wege zu allen anderen Knoten.

Lösungsvorschlag

Nr.	besucht	A	M	N	P	R
0		∞	0	∞	∞	∞
1	M	∞	0	7	∞	∞
2	N	9		7	∞	11
3	A	9			11	11
4	P				11	11
5	R					11

nach	Entfernung	Reihenfolge	Pfad
M → A	9	3	M → N → A
M → M	0	1	
M → N	7	2	M → N
M → P	11	4	M → N → A → P
M → R	11	5	M → N → R

- (c) Beschreiben Sie wie ein Heap als Prioritätswarteschlange in diesem Algorithmus verwendet werden kann.

Lösungsvorschlag

Ein Heap kann in diesem Algorithmus dazu verwendet werden, den nächsten Knoten mit der kürzesten Distanz zum Startknoten auszuwählen.

- (d) Geben Sie die Operation „Entfernen des Minimums“ für einen Heap an. Dazu gehört selbstverständlich die Restrukturierung des Heaps.

Lösungsvorschlag

```
static int NO_VALUE = Integer.MIN_VALUE;

static void heapify(int a[], int i, int last) {
    int smallest = i;
    int l = 2 * i + 1;
    int r = 2 * i + 2;

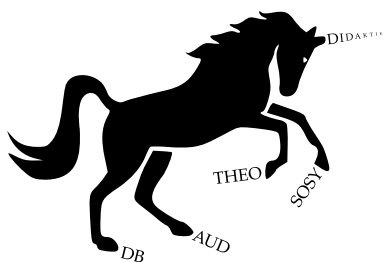
    if (l <= last && a[l] != NO_VALUE && a[l] < a[smallest]) {
        smallest = l;
    }

    if (r <= last && a[r] != NO_VALUE && a[r] < a[smallest]) {
        smallest = r;
    }

    if (smallest != i) {
        int swap = a[i];
        a[i] = a[smallest];
        a[smallest] = swap;
        heapify(a, smallest, last);
    }
}

static int removeMin(int a[]) {
    int result = a[0];
    int last = a.length - 1;
    while (last > 0 && a[last] == NO_VALUE) {
        last--;
    }
    a[0] = a[last];
    a[last] = NO_VALUE;
    heapify(a, 0, last);
    return result;
}
```

Code-Beispiel auf Github ansehen: [src/main/java/org/bschlangaul/aufgaben/aud/baum/Heap.java](https://github.com/bschlangaul/aufgaben/aud/baum/Heap.java)



Die Bschlangaul-Sammlung

Hermine Bschlangaul and Friends

Eine freie Aufgabensammlung mit Lösungen von Studierenden für Studierende zur Vorbereitung auf die 1. Staatsexamensprüfungen des Lehramts Informatik in Bayern.



Diese Materialsammlung unterliegt den Bestimmungen der Creative Commons Namensnennung-Nicht kommerziell-Share Alike 4.0 International-Lizenz.

Hilf mit! Die Hermine schafft das nicht allein! Das ist ein Community-Projekt! Verbesserungsvorschläge, Fehlerkorrekturen, weitere Lösungen sind herzlich willkommen - egal wie - per Pull-Request oder per E-Mail an hermine.bschlangaul@gmx.net. Der TeX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden: https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Module/30_AUD/90_Graphen/10_Dijkstra/Aufgabe_Graph-M-A-P-R-N.tex