Multi-Way Heaps

We are considering d-trees; a generalization of binary trees where each node has up to d children, where d is an integer.

As usual, let n denote the number of nodes in the tree, and define the height of a tree to be the maximal number of edges from the root to a leaf.

Question a: Determine the number of leaves as a function of d and n. $\square$

Question b: Determine the smallest possible height of a d-tree as a function of d and n. $\square$

We now use d-trees to implement a priority queue. As for a standard heap, we use the structural invariant that the tree is filled from top to bottom and the last layer from left to right.

Question c: Argue that the following complexities can be obtained (as usual, we assume that *DecreaseKey* gets a pointer to the element as an argument):

$$\begin{array}{lcl} \textit{Insert} & \in & O(\log_d n) \\ \textit{DeleteMin} & \in & O(d \log_d n) \\ \textit{DecreaseKey} & \in & O(\log_d n) \end{array}$$

$\square$

The Single Source Shortest Path Problem can be solved using Dijkstra's algorithm which is implemented using a priority queue. With n nodes and m edges, at most n *Insert* and *DeleteMin* operations and m *DecreaseKey* operations are carried out, and these operations dominate the running time of Dijkstra's algorithm. Thus, using a standard binary heap, the running time is $O(m \log n)$.

We now consider scenarios where $m = n \cdot f(n)$ for some function $f(n) \in \omega(1)$, i.e., a function f, where $f(n) \to \infty$ for $n \to \infty$. In other words, we assume that m grows asymptotically faster than n.

Question d: Show that for such graphs, we can obtain a faster implementation than with the binary heap by choosing an appropriate d and using a d-tree.

Hint: d does not have to be a constant, and recall that $\log_d n = \log n / \log d$. $\square$