

Data Structures for Network Problems

David Pisinger
pisinger@man.dtu.dk

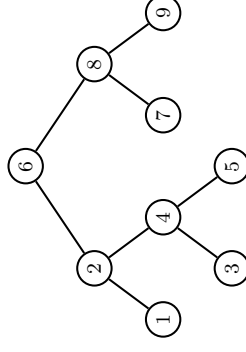
DTU Management
Technical University of Denmark

Data Structures – p. 1

Exercises

Exercise 2, answer

a Build a binary tree 6, 2, 4, 3, 8, 7, 1, 5, 9



Data Structures – p. 3

Exercises

Exercise 1, answer

- a $4n^3 + n^4 + n^3(\log n)^2$
 $O(n^4)$
- b $100n + n \log n + \sqrt{n}$
 $O(n \log n)$
- c $5n^7 + 100n^6 + n^8$
 $O(n^8)$
- d $n\sqrt{n} + n \log n + n \log(n^2)$
 $O(n\sqrt{n})$

Data Structures – p. 2

Exercises

Exercise 2, answer

- b $\text{successor}(i)$: First find (i) . If node has a right-child, follow the right-child and repeatedly choose left-child to the bottom of the tree. If node has no right-child, follow left-parent upward, until you meet first right-parent.
- c $\text{predecessor}(i)$: First find (i) . If node has a left-child, follow the left-child and repeatedly choose right-child to the bottom of the tree. If node has no left-child, follow right-parent upward, until you meet first left-parent.
- d $\text{delete}(i)$: First find (i) . If node has no children, delete it. If it has one child, delete i and move the tree up. If node has two children, find $\text{successor}(i)$, and move it to i 's position.

Data Structures – p. 4

Exercises

Exercise 3, answer

- a n^2 insert, n^2 delete, n^3 minimum
A sorted list works best, since minimum can be found in $O(1)$ and insert/delete can be done in $O(n)$ time. This gives overall time $O(n^3)$.
- b n insert, n find, n minimum
A balanced binary tree works best, since all operations take $O(\log n)$. This gives overall time $O(n \log n)$.
- c n^2 insert, n^2 find, n minimum
A balanced binary tree works best, since all operations take $O(\log n)$. This gives overall time $O(n^2 \log n)$.
- d n insert, n^2 find
A sorted table could be used. Find costs $O(\log n)$ while insert costs $O(n)$. Overall time is $O(n^2 \log n)$. Balanced binary trees can also be used.

Data Structures – p. 5

Exercises

Exercise 4, answer

- Inserting the n numbers into a binary balanced tree takes $O(n \log n)$. Repeatedly finding minimum and deleting the number takes $O(\log n + \log n)$. This is done n times, so the overall running time is $O(n \log n)$.
- This is the same running time as Quicksort when using the median for choosing the partition number t . However, Quicksort is an in-place algorithm, meaning that no additional space is needed during the calculation. Using a binary balanced tree, will demand some extra space of magnitude $O(n)$.

Data Structures – p. 6