UNIVERSITY COMPUTER SCIENCE • EXERCISE SHEET

5.4 Trees and heaps

Name: _____ Class: _____ Date: _____ **Total: 24 marks**

KEY WORDS heap • binary search tree

Objective

Walk trees by their invariants and sift heaps by their array indices.

1 Key knowledge and worked examples

■ The invariants

BST: left $<$ node $<$ right – inorder reads sorted ✓. Min-heap: parent $\leq$ children, complete shape, children of index i at $2i+1$ and $2i+2$ ✓.

■ Worked example: heights

Perfect trees hold $2^{h+1}-1$ nodes: 7 at height 2, 15 at 3, 31 at 4 ✓. Sorted insertion $1, 2, 3, 4$ instead builds a chain of height $3 = n-1$ ✓.

2 Practice

2.1 Inserting $5, 3, 8, 1$ into an empty min-heap leaves which array? [3]

2.2 In a BST, where does the successor of a node with a right child live? [3]

2.3 A heap array holds 9 elements. Give the indices of the children of index 2. [3]

3 Exam-style questions

3.1 The keys $50, 30, 70, 20, 40, 60, 80$ build a BST in that order.

Draw or describe the tree and state its height. Give the comparison sequence for searching 40 and for searching 45 (absent), with each comparison count. Insert 35 and give the new inorder walk. State the height that would have resulted from inserting the same keys in sorted order instead. [8]

3.2 A min-heap starts as the array $[1, 3, 8, 5]$.

State the array after `heappush(2)`: give the index 2 first lands at, each sift-up swap, and the final parent-child check. Then state which index `heappop` removes, what moves to the root, and the array after one sift-down. [7]

Solutions

2.1

- $[1, 3, 8, 5]$ ✓.
- 3 sifted over 5; 1 over 5 then 3; 8 stayed ✓.

2.2

- The leftmost node of the right subtree ✓.

2.3

- Children of $i = 2$: indices 5 and 6 ✓.

3.1

- Tree: 50 root; $30, 70$ level 1; $20, 40, 60, 80$ level 2 – height 2 ✓.
- Search 40: 50 (left), 30 (right), 40 – 3 comparisons ✓.
- Search 45: $50, 30, 40$ then a null right link ($45 > 40$) – 3 comparisons, absent ✓.
- After inserting 35 (left child of 40), inorder reads $20, 30, 35, 40, 50, 60, 70, 80$ ✓.
- Sorted insertion builds a chain of height 6 ✓.

3.2

- `heappush(2)` lands at index 4 (next free slot) ✓.
- Sift-up: swap with 5 (index 1) $\rightarrow [1, 2, 8, 5, 3]$; check parent 1 at index 0: $1 \leq 2$, stop ✓.
- Final array $[1, 2, 8, 5, 3]$ – parents $1 \leq 2, 8$; $2 \leq 5, 3$ ✓.
- `heappop` removes index 0 (value 1); the last element 3 moves to the root ✓.
- One sift-down against the smaller child 2: swap $\rightarrow [2, 3, 8, 5]$ ✓.