HW 3

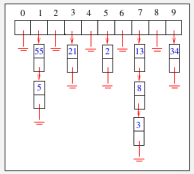
- for data structure operations –
  - show the structure after each operation
    - OK to show hashtable steps in a grid e.g. →

0	1	2	3	4	5	6	7	8	9
					2				
					2		3		
	5				2		3		
	5				2		3	8	

  - if intermediate steps are shown (e.g. after insert/delete, before restructure), clearly indicate which steps are the final result of each operation

HW 3

- #3 – hashtable with separate chaining
  - insert at the head of each bucket's list – Θ(1)
  - elements within a bucket are unsorted so insertion position doesn't matter for correctness – use the most efficient position
    - Θ(1) insertion at the end requires also maintaining tail pointers
    - if duplicate checking is done, a bucket's list must be searched – at that point adding at the tail is only Θ(1) additional work but insertion as a whole becomes expected Θ(n/N)



- #8 – how long does it take to search in a sorted array?
  - though the overall runtime for deletion is dominated by the shifting so sequential search instead of binary search doesn't affect the big-Oh
- #8 – where in a max heap will the smallest element be found?
  - n/2 leaves so the search time is still O(n) – doesn't improve big-Oh, just constant factors

HW 3

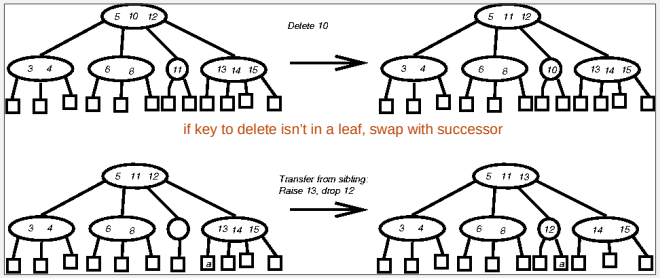
- errors in data structure operations
  - 3 if invalid structure, 7 if just incorrect operation

HW 3

- 2-4 trees
  - structure: a node can have 1-3 keys, *k* keys → *k*+1 children, all leaves at the same level
  - ordering: search tree
  - split and promote (only) when there's overflow – 4 keys in a node
  - underflow – transfer from an adjacent sibling via the parent if possible, merge with adjacent sibling (pulling a key from the parent) otherwise
  - overflow/underflow is handled the same at every non-root level – if promotion or merge creates overflow/underflow in parent, repeat
    - at the root, create new root with promoted key (overflow) or delete the root (underflow)

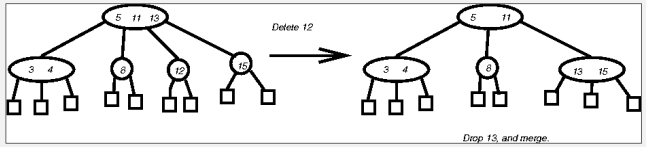
HW 3

- 2-4 tree deletion
  - transfer if there's an adjacent sibling with 2 or 3 keys



HW 3

- 2-4 tree deletion
  - merge if all adjacent siblings only have 1 key



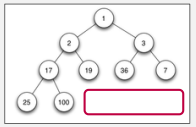
  - if dropping a key from the parent causes an underflow there, repeat – transfer if possible, merge otherwise
  - for underflow at the root, remove the root

HW 3

- hashtables
  - be careful of math errors
  - double hashing probe sequence is (h(k) + i h'(k)) mod N
    - the secondary hash function is applied to the original key – h'(k), not h(h'(k))
    - h'(k) specifies how many spaces to skip for the next spot to check
  - handling deletion
    - for sequential probing, reinsert all keys immediately following the deleted key
    - for quadratic probing and double hashing, use deletion markers

HW 3

- max heaps
  - structure: complete binary tree – missing leaves (gaps) only at the right end of the last level
  - ordering: parent ≥ children



  - insert as last element, bubble up (swap parent and child) to fix ordering
  - remove max: swap root and last, delete last, bubble root down (swap parent and largest child) to fix ordering