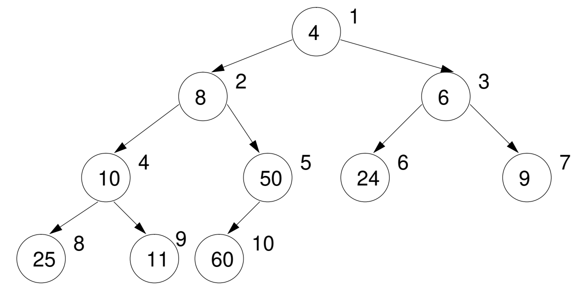
CSE 326
Lecture 17

d-heaps

Winter Quarter 1999

Standard Binary Heap

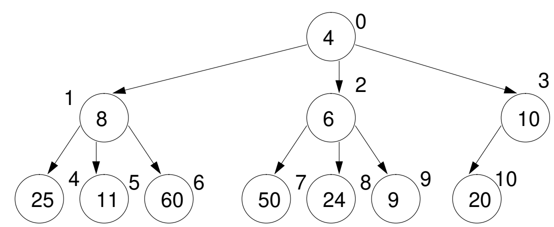


1	2	3	4	5	6	7	8	9	10		
4	8	6	10	50	24	9	25	11	60		

d-Heaps

- Use a complete d-ary tree.
- Each node has d children instead of just 2.
- Store the root at 0.
- The children of i are stored in di+1, di+2, ..., di+d.
- The parent of i is stored in (i-1)/d

3-Heap



0	1	2	3	4	5	6	7	8	9	10	
4	8	6	10	25	11	60	50	24	9	20	

Are d-Heaps Better Than Binary Heaps?

- Theory says that 4-heaps are probably better.
- 4-heaps take fewer instructions per operation than 2-heaps
- 4 heaps can take better advantage of cache memory than 2 heaps

The Theory

- d-heaps are better than binary heaps for insertion both in worst case and average case.
- For some d, the d-heap is better than binary heaps for delete_min.

  Assuming that all delete-min's goto the leaves the time to delete_min is

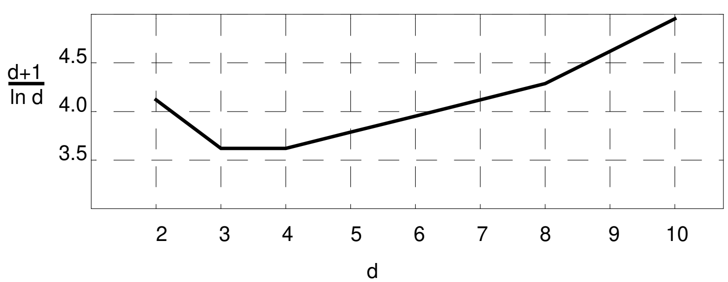
$$(ad + b) \log_d N$$

For simplicity let a = b = 1, then the time per delete_min is

$$(d + 1) \log_d N = (d + 1) \ln N / \ln d$$

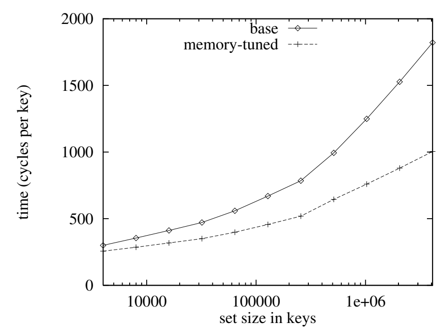
d	2	3	4	5	6	7	8	9	10
(d+1) / ln d	4.33	3.64	3.61	3.73	3.91	4.11	4.33	4.55	4.78

Plot of d vs. (d+1) / ln d



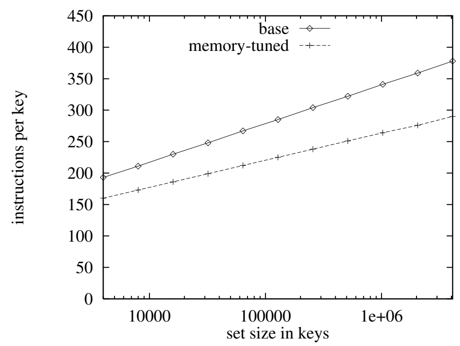
Heapsort Execution Time
2-Heap vs. 4 Heap

- Sorting 8 byte integers, randomly chosen
- DEC alpha 250 with 2MB level 2 cache



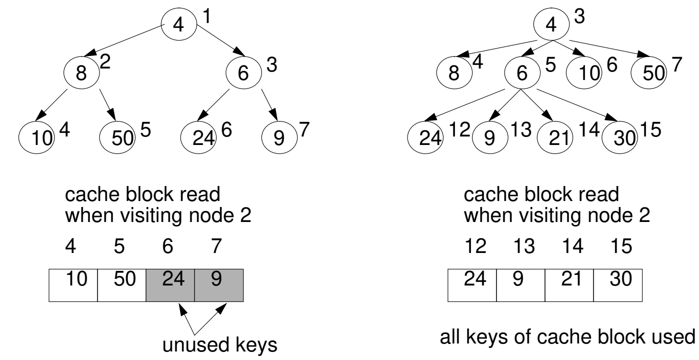
Heapsort Instruction Count
2-Heap vs. 4-heap

- Atom simulator on Dec alpha 250



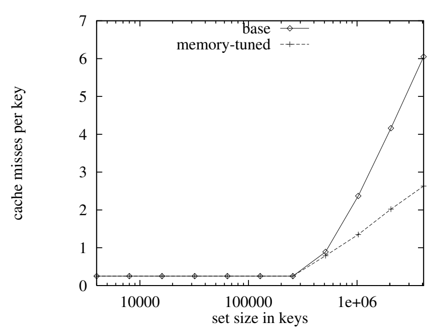
Fully Utilizing Cache Blocks

Alignment needed so array starts at 3



Heapsort Cache Misses
2-Heap vs. 4-Heap

- Atom simulator on DEC alpha 250
- 32 byte cache blocks, 4 integers per block



Heaps on Heaps

- The binary heap

  + simple, elegant, and works great
  + building a binary heap can be done in O(n) time
  + insertion: O(log n) time worst case and constant time on average
  + delete_min: O(log n) time worst case
  + supports additional operations of decrease, increase, delete
  - may be somewhat inflexible in some applications

- The d-heap

  + has all the advantages of binary heaps
  + in some applications where performance is critical a d-heap is a better choice than a binary heap
  - slightly more complicated than binary heap
  - may also be inflexible in some applications