

- Trace Dijkstra's sssp algorithm on the digraph with adjacency weights below.

Show the distance array and tree-so-far after each node is added to the tree-so-far.

	A	B	C	D	E	F
A		5	3	1		
B					1	0
C				0	8	
D					1	5
E	2					3
F	0	1				

- For an n -node graph, give the maximum number of times D's sssp calls decreasekey. Justify.
- Give a digraph with 3 nodes, 3 edges (exactly one with negative weight), and a start node so that Dijkstra's sssp gives the right answer.
 - Repeat (i) so that it gives the wrong answer.
- Trace slow buildminheap on this array. Show the array, and the corresponding complete binary tree, after each bubbleup.
[9 2 6 4 7 3 8 1 0]
 - Repeat for fast buildminheap. Show the array, and tree, after each trickledown.
- Give an array with 7 keys on which slow buildminheap performs the maximum possible number of key comparisons.
 - Prove that your answer to (i) is correct.
 - How many comparisons does it perform?
 - v,vi) Repeat (i,ii,iii) for fast buildminheap.
- An array has 100 keys. Give the min and max number of key comparisons that can be performed by slow buildminheap. Leave your answer as an arithmetic expression.
 - Repeat (i) for fast buildminheap.

1. node	A B C D E F	add tree-so-far	bup(5)	2	bup(6)
parent	A - - - - -	A A	4		3
dist	0 - - - - -		9 7		6 8
			1 0		
parent	A A A A - -	D A			
dist	0 5 3 1 - -	D	bup(7)	1	
			2		3
parent	A A A A D D	E A	4 7		6 8
dist	0 5 3 1 2 6	D	9 0		
		E			
parent	A E A A D E	C A	bup(8)	0	
dist	0 4 3 1 2 5	D C	1		3
		E			
parent	A E A A D E	B A	2 7		6 8
dist	0 4 3 1 2 5	D C	9 4		
		E	(ii)	9	
		B	2		6
parent	A E A A D B	F A	4 7		3 8
dist	0 4 3 1 2 4	D C	1 0		
		E			
		B	td(3)	9	
		F	2		6

2. If each pair of nodes is adjacent, then when t nodes are in the tree-so-far, and a node is added, then each of the $n - (t + 1)$ fringe nodes could have its distance decreased. t starts with 0, so the sum is $(n-1) + (n-2) + \dots + 1 = n(n-1)/2$. Also, it is easy to create edge weights so that this number of decrease-keys occurs (exercise).

3. There are many correct answers.

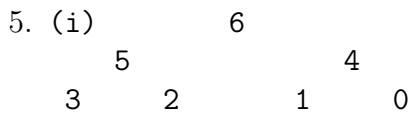
	A B C	A B C
A	1 2	2 1
B	-1	-2

4. (i)		9		
	2		6	
	4 7		3 8	
	1 0			

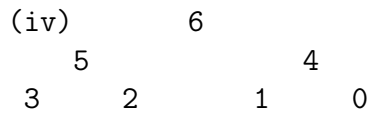
bup(1)	2	bup(2)
9		6
4 7		3 8
1 0		

bup(3)	2	bup(4)
4		6
9 7		3 8
1 0		

	0 7		3 8
1 4			
td(2)	9		
2			3
0 7			6 8
1 4			
td(1)	9		
0			3
1 7			6 8
2 4			
td(0)	0		
1			3
2 7			6 8
9 4			



(ii) Each key bubbled-up is smallest so far, so bubbles up to root, which is the worst case for any bubble up. (iii) $1 + 1 + 2 + 2 + 2 + 2 = 10$



(v) Each key trickled-down is largest so far, so trickles down to leaf, which is the worst case for any trickledown. (vi) $2(1 + 1 + 2) = 8$

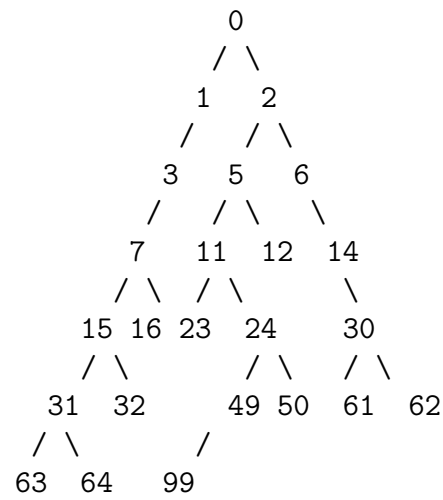
6. A compact binary tree with 100 nodes has 1,2,4,8,16,32,37 nodes at depths 0,1,2,3,4,5,6 respectively.

(i) The minimum number of key comparisons occurs when, in each bubble-up, the 1st comparison shows the parent's key is not larger than the child, so only 1 comparison for each bubbleup, so 99 comparisons.

The maximum number of comparisons occurs when, in each bubble-up, the new key is smaller than all keys on the path to the root. So, summing by level, $2 * 1 + 4 * 2 + 8 * 3 + 16 * 4 + 32 * 5 + 37 * 6 = 480$.

(ii) The minimum occurs when, in each trickle-down, after the min child is found, the comparison shows the node's key is not larger than the min child. So, if the node has only one child, then 1 comparison, otherwise 2 comparisons. Here, the first trickledown is from position 49, which has only 1 child, at position 99. So $1 + 49 * 2$.

The maximum number occurs when, in each trickledown, the new key is larger than all keys on a longest path to a root.



The diagram above shows the path from last position 99 to root position 0, as well as left and right edges of the tree.

So $td(49)$ does 1 comparison,

$td(48) \dots td(25)$ each 2,

$td(24)$ 3,

$td(23) \dots td(12)$ each 4,

$td(11) \dots td(6)$ each 6,

$td(5) \dots td(3)$ each 8,

$td(2), td(1)$ both 10,

$td(0)$ 12.

So $1 * 12 + 2 * 10 + 3 * 8 + 6 * 6 + 12 * 4 + 1 * 3 + 24 * 2 + 1 * 1 = 192$.