Building a Heap in Linear Time

Idea:

• A sequence of insert operations builds the heap from the top down.

• Hence most of the inserts have to traverse a path $O(\log(n))$ deep (over half the vertices are in the last two levels of a full tree, or just the last level of a complete tree).

• Instead, if all elements are to be inserted at the same time, a full binary tree of the elements in an arbitrary order can be formed, and the tree can be rearranged from the bottom up to be a heap.

• Visit the vertices in reverse level order (working from $H[n-1]$ down to $H[0]$), percolating each vertex down to make that subtree into a heap (since the two subtrees must already be heaps).
Linear time heap construction algorithm:

procedure BUILDHEAP:
  Store $n$ items into $H[0] \ldots H[n-1]$ in any order.
  for $i := (n-1)$ downto 0 do PERCDOWN($i$)
end

Idea: Because the heap is built from bottom up, the work is weighted towards the vertices of low height, and when it is added up, it can be shown that the total time is linear.
Time to construct a heap:

- The time to visit all vertices is proportional to $n$ plus the number of exchanges performed by PERCDOWNs.
- For $n=1$ no exchanges are made, so assume $n>1$ and let $N$ be the number of vertices on level 1.
- Then, since the portion of the tree from level 1 up is a complete binary tree with $N$ leaves, in the entire tree, the level of the root is $\log_2(N)+1$ and, for $i \geq 1$, there are $N/2^{i-1}$ vertices on level $i$.
- Since PERCDOWN from a vertex at level $i \geq 1$ makes at most $i$ exchanges, the total number of exchanges for all PERCDOWN operations is:
And since this derivation contains an inequality, it follows that at most $n+1$ exchanges are performed.