Computing Science 204 (B1)

Worst Case Analysis of Heap Sort

Friday, August 4

The pseudocode for the heap sort algorithm given below is taken from the second edition of Baase's book, as is the worst case analysis.

```
procedure HeapSort(var L : Array; n : integer);

var   i, heapsize : Index;
      max : Key;

begin (* heap construction *)
    for i := n div 2 to 1 by -1 do
        FixHeap(i,L[i],n);
    end; (* for *)
    (* repeatedly remove the key at the root *)
    (* and rearrange the heap               *)
    for heapsize := n to 2 by -1 do
        max := L[1];
        FixHeap(1,L[heapsize],heapsize-1);
        L[heapsize] := max;
    end; (* for *)
end; (* HeapSort *)

procedure FixHeap(root : Index; K : key; bound : Index);

var   vacant, largerChild : Index;
      done : boolean;

      while (2*vacant <= bound) and (not done) do
          largerChild := 2*vacant;
          if (2*vacant < bound) and (L(2*vacant+1] > L[2*vacant])
          then
              largerChild := 2*vacant + 1;
          if K < L[largerChild] then
              L[vacant] := L[largerChild];
              vacant := largerChild;
          else
              done := true;
          end; (* else *)
      end; (* while *)
      L[vacant] := K;
end; (*FixHeap *)
```

Note that `FixHeap` does $2d$ comparisons of keys in the worst case on a heap of depth d. This follows immediately from the facts that two comparisons of keys are done in each iteration of the while loop (except when the vacant node is the sole node that may have only one child), and that the maximum iterations of the while loop is the depth of the heap.

Now consider the heap construction loop in `HeapSort`, and let

$$d = \lfloor \log_2 n \rfloor,$$

the depth of the heap with all n nodes, then the number of comparisons done by `FixHeap` in the worst case is twice the depth of the subtree it works on, which is d minus the level of the subtree's root.

Thus, the number of comparisons done by the heap construction is at most

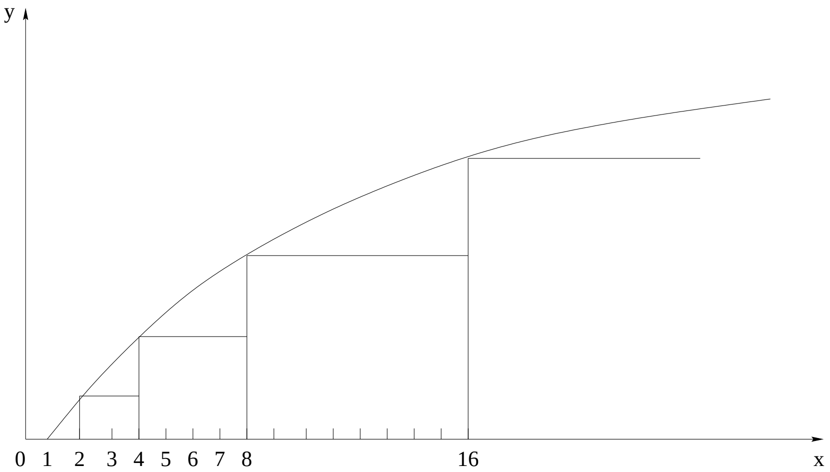
$$\sum_{\ell=0}^{d-1} 2(d-\ell)(\#\text{ of nodes at level } \ell) = 2\sum_{\ell=0}^{d-1}(d-\ell)\cdot 2^{\ell} = 2^{d+2} - 2d - 4,$$

and since $d = \lfloor \log_2 n \rfloor$, then the number of comparisons done by the heap construction loop is in $\Theta(n)$, that is, the heap is constructed in *linear time*.

Now consider the second loop in `HeapSort`, the number of comparisons done by `FixHeap` on a heap with k nodes is at most $2\lfloor \log_2 k \rfloor$, so the total for all the deletions is at most

$$2\sum_{k=1}^{n-1} \lfloor \log_2 k \rfloor.$$

In order to get an estimate on this sum, we look at the graph of the function $y = \log_2 x$ as below.



Making a simple change of variable in the summation, we get

$$2\sum_{k=1}^{n-1} \lfloor \log_2 k \rfloor = 2\sum_{j=1}^{d-1} j\cdot 2^j + d\cdot\left(n - 2^d\right),$$

where the summation term includes the areas of all the complete rectangles with height j and width 2^j, and the last term is the area of the last incomplete rectangle with height d and width $n - 2^d$. Therefore,

$$2\sum_{k=1}^{n-1} \lfloor \log_2 k \rfloor = 2\left(d\cdot 2^{d-1} - 2^d + 1\right) + d\cdot\left(n - 2^d\right) = n\cdot d - 2^{d+1} + 2$$

which is in $\Theta(n \log_2 n)$, and we have the following

Theorem. Heapsort does at most $2(n-1)\lfloor \log_2 n \rfloor$ comparisons of keys in the worst case.

proof. The heap construction takes at most $2^{d+2} - 2d - 4$ comparisons, and the deletions take at most $2\left(n\cdot d - 2^{d+1} + 2\right)$ comparisons. The total is

$$2n\cdot d - 2d = 2(n-1)\lfloor \log_2 n \rfloor,$$

which is in $\Theta(n \log_2 n)$.