Algorithms Homework Assignment 3

Andrew Osborne

February 12, 2019

Conventions When I refer to $\mathbb{N}$, I speak of

$$\mathbb{N} = \{1, 2, 3, \ldots\}$$

Problem 6-1

Part a.

The answer here is quite clear. Consider the array

```
A = [6,5,4,8,9,21]
```

When using our typical $\Theta(n)$ routine for building a heap, we acquire

```
A = [21,9,6,8,5,4]
```

and when we use the `BUILD-MAX-HEAP'` as suggested in the text, we acquire instead

```
A = [21,8,9,5,6,4]
```

and these two are clearly unequal

Part b

Our code as presented in the text is

```
BUILD-MAX-HEAP'(A)
A.heap-size = 1
for i = 2 to A.length
   MAX-HEAP-INSERT(A,A[i])
```

In the worst case, `MAX-HEAP-INSERT(A,A[i])` runs in less than or equal to $\lceil lg(i) \rceil$ time. Then, the total runtime of this algorithm on array of size n is

$$\sum_{k=2}^{n} lg(k)$$

for which we can create upper and lower bounds via integration.

$$\sum_{k=2}^{n} lg(k) \leq \int_{2}^{n+1} lg(x)\, dx = x\, lg(x) - ln(x)|_2^{n+1}$$

which is

$$(n+1)lg(n+1) + ln(n+1) + c = O(n\,lg(n))$$

So therefore our total runtime is $O(n\ lg\,n)$. On the other hand, we may discern a lower bound with the integral

$$\sum_{k=2}^{n} lg(k) \leq \int_{1}^{n} lg(x)\,dx = x\,lg(x) - ln(x)|_{1}^{n} = \Omega(n\ lg\,n)$$

Then, in the worst case, our runtime is $\Theta(n\,lg\,n)$. Note that we need not specify a worst–case scenario for this algorithm because we know the worst–case runtime of `MAX_HEAP_INSERT` which is the worst case of the algorithm in question. Although, the worst–case scenario for this algorithm would be a sorted array.

Problem 7.2-2

We seek the running time of quicksort on an array of elements which are identicle. First, we must determine the runtime of `PARTITION(A,p,r)` when called on such an array. Recall the pseudocode for `PARTITION(A,p,r)` .

```
PARTITION(A,p,r)
1   i = p - 1
2   ref = A[r]
3   for j = p to r-1
4     if A[j] <= ref
5       i = i + 1
6       exchange A[j] with A[i]
7   exchange A[r] with A[i+1]
8   return i+1
```

Notice that since all elements of our array are equal, the test condition on line 4 is always met, and then the runtime of `PARTITION` on an array of identicle elements is $\Theta(n)$. Moreover, in this case, `PARTITION(A,p,r)` will always return r. Then, calling `QUICKSORT` on an array, A, of length n, we will first call `q=PARTITION(A,1,n)` , and q will hold the value n. We will then call `PARTITION(A,0,r-1)` and `PARTITION(A,r-1,r)` but the right sub–array is a constant time call because a single element array is always sorted. We will continue in this way until we have progressed through each element of our array. Then the total time of `QUICKSORT(A,0,n)` is easily expressed as

$$\sum_{i=2}^{n} c\,i < c\sum_{i=1}^{n} i = \frac{c}{2}n(n+1) = \Theta(n^2)$$

This is worst–case behavior, which makes sense because we know that the worst case behavior of `QUICKSORT` occurs when attempting to sort a list which is already sorted, and an element containing many copies of a single number is sorted.

Problem 7.2-3

We seek to show that `QUICKSORT(A,p,r)` runs in quadratic runtime when A is sorted in decreasing order (with distinct elements). We actually only need show that `PARTITION(A,p,r)` will always return p, which yields a worst–case split, and therefore `QUICKSORT`will have worst–case time cost.

Referencing the code written in **Problem 7.2-2**, if our array is in decreasing order, then the last element in the array is also the least element in the array and therefore the test on line 4 will never be satisfied when called in `PARTITION(A,1,n)` . Then, A[1..n] was in decreasing order, so A[1..n-1] is also in decreasing order, and our array after a single call of `PARTITION` is `A=[A[n],A[1..n-1]]` and subsequent non-constant calls of Partition will be executed on the right sub–array of A. This will continue to create worst–case splits of our array and will have a total time cost of

$$T(n) = \sum_{i=2}^{n} i = \Theta(n^2)$$

References

I used this website to look–up some series identities that were used in solving recursion equations
https://en.wikipedia.org/wiki/List_of_mathematical_series