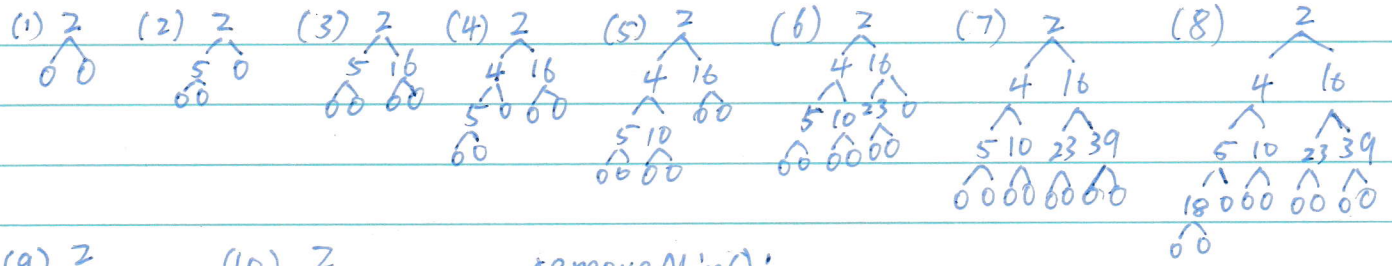
Nelson Dai - V00815253
CSC 225 - Assignment 3
July 1st, 2016

1.a) Since the sequence is already sorted, it would be split into two almost equal subsequences. Let $T(n)$ be the worse case of size n. Then we have: $T(n) = T_n(\lfloor \frac{n}{2} \rfloor) + T_n(\lfloor \frac{n}{2} \rfloor) + cn = 2T_n(\frac{n}{2}) + cn$

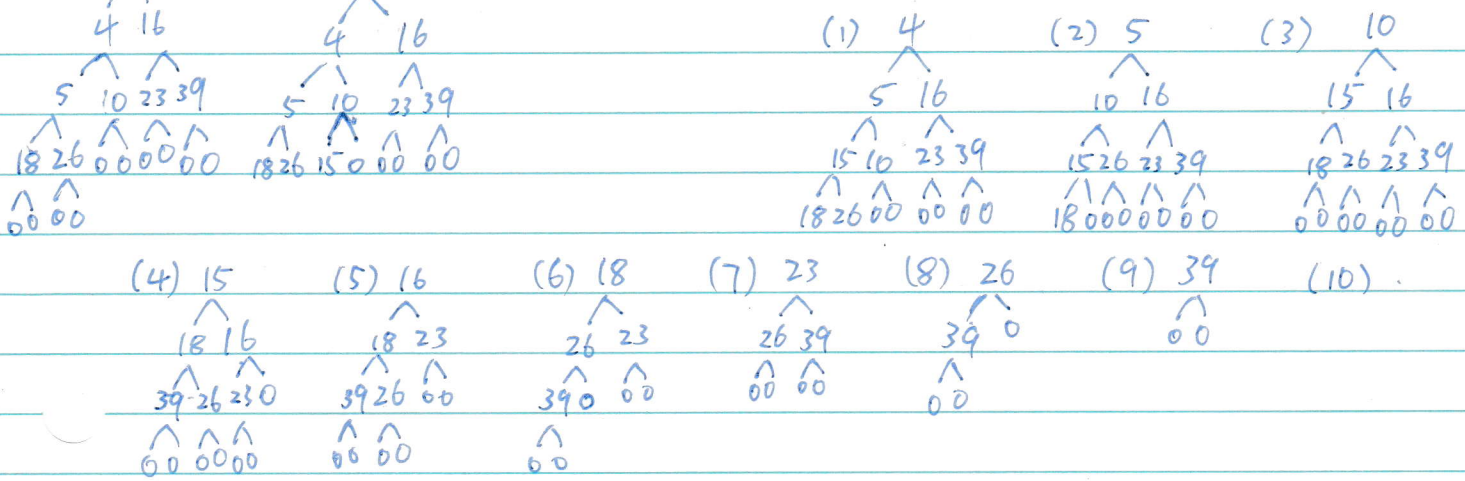
Since we have n items and the sequence is havled k times such that $n < 2^k$, now we have a tree with hight $O(\log n)$. Since we process each item of the sequence at most once at each level, the running time of this deterministic quick-sort algorithm of size n is $O(n \log n)$.

b) In this version we are setting the middle item as our pivot. As we know the worse-case running time of quick-sort is $O(n^2)$ and to reach that situation we want the largest or smallest number as the pivot every time, which means to reach $O(n^2)$ every time the middle item should be the largest or smallest in the squence.

2. insert():



removeMin():



3. PreorderNext(Node v)

```
if v.isInternal() then
return v.leftchild
else
  Node p = p.Parent()
  if v.isleftchild(p) then
   return rightchild(p)
   else
    while not v.isleftchild(p) do
          v=p
          p=p.Parent
         return rightchild(p)
```

inorderNext(node v)

```
if v.isInternal() then
  v = rightchild(v)
 while not v.isExternal do
  v = leftchild(v)
  return v
else
 Node p = v.parent()
 if v.isleftchild(p) then
   return p
 else
  while not v.isleftchild(p) do
       v=p
       p=p.parent()
       return p
```

PostorderNext(node v)

```
if v.isInternal() then
   p = v.parent()
   if v.isRightchild(p) then
    return p
   else
     v= Rightchild(p)
while not v.isExternal do
     v= leftchild(v)
   return v
   else
     p = v.Parent()
   if v.isleftchild(p) then
    return rightchild(p)
   else
     return p
```

The worse-case running time for them are all O(logn) where n is the height of T

4. Prove by induction

Base case: when $n=0$ then $E(T) = I(T) = I(T)+2n = 0$ ✓

I.H. : Assume $n_0 \in N$ then $E(T) = I(T)+2n$ such that $n=k+1$

I.S. : Assume T is full binary tree and T' be the tree after 2 external nodes have been removed from T. p is an internal node of T. d is the depth of node T.

$$\begin{cases} E(T) = E(T') - d(p) + 2d(L) \\ I(T) = I(T') + d(p) \end{cases}$$

$$\therefore E(T) = I(T') + 2n_0 - d(p) + 2d(L)$$
$$= I(T) - d(p) + 2n_0 - d(p) + 2d(L)$$
$$= I(T) + 2n_0 - 2d(p) + 2d(l) + 2$$
$$= I(T) + 2(n_0+1)$$

Conclusion: for all $n_0 \in N$ $E(T) = I(T)+2n$

5. Every number in the squence S from $[0, n^3-1]$ can be represented by a three digit number in the base n

$$(n-1)(n^2+n+1)$$

conversion of each item into base n is $O(1) \to O(n)$ for the sequence

Then we use radix-sort to sort in $O(3n)$ which is $O(n)$