Unity id : amanur
Name - Anusha Gururaja Manur

Homework -1

1. Purpose: Learn about bubblesort and practice how loop invariants are used to prove the correctness of an algorithm. Please re-read Section 2.1 in our textbook and solve problem 2-2 on page 40 of the textbook.

a. In order to show that BUBBLESORT actually sorts, what else do we need to prove?

Ans : Other than proving that bubblesort terminates(1) and is sorted in a decreasing/ increasing order (2), we need to prove that A' which is the output of bubblesort has the same elements - (3) as that of the input A and in sorted order.

b. **State precisely a loop invariant for the for loop in lines 2–4, and prove that this loop invariant holds. Your proof should use the structure of the loop invariant proof presented in this chapter.**

Ans : We can state the loop invariant for the for loop in lines 2–4 as -
The subarray A[i...j] will have the smallest element of the sub array.
Let us consider the 3 properties to prove that the loop invariant holds.

- Initialization - Before we start the execution of the inner loop, since j is initialized to the length of the array, the subarray will have the entire array and will hence contain the smallest element of the array.

- Maintenance - From the start of the inner loop, j=A.length and runs till i+1, we compare A[j] and A[j-1]. If A[j] < A[j-1], we store the lesser element in A[j-1]. If A[j] > A[j-1], the exchange does not happen, but A[j-1] will contain the smaller of the 2 elements. For the next iteration of the inner loop, j is decremented to j' and the next smallest element will be in the subarray A[i..j']. The swapping continues till j=i+1 at which point A[i] will have the smallest element.

- Termination - The last iteration of the loop is at j=i+1 and terminates when j becomes i. At this point, due to continuous swapping between A[j] and A[j-1], the smallest values would have been brought down to A[i..j]=A[i..i] which is A[i].

c. **Using the termination condition of the loop invariant proved in part (b), state a loop invariant for the for loop in lines 1–4 that will allow you to prove inequality (2.3). Your proof should use the structure of the loop invariant proof presented in this chapter.**

Ans: At the start of each iteration of the for loop on line 1, the subarray A[1...i − 1] will have the smallest i-1 elements which are sorted.

- Initialization - We have an array with 0 elements which are sorted in a trivial sense.
- Maintenance -
  - After the first execution of the outer loop, j runs from A.length to i+1 =2 and the inner loop sorts elements and puts the smallest value in A[i]= A[1]. The inner loop does not alter the subarray A[1...i-1] because it alters elements between A[j] and A[i+1].
  - For the next iterations of the outer loop, i gets incremented to i+1 (let's say i') and j runs through j=A.length to i' and puts the smallest element of the subarray in position i'. Therefore, the first i-1=2 elements have been sorted and are in the first 2 positions. (i is already incremented)
  - Therefore, after ith iteration, first i-1 elements will have been sorted to first i-1 positions.

- Termination - At the last iteration of i, i=A.length -1 and j runs from A.length till i+1= A.length and puts the smallest element of the subarray in position A.length. Therefore, the first i-1=A.length elements have been sorted and are in the first A.length positions. (i is already incremented)

d. **What is the worst-case running time of bubblesort? How does it compare to the running time of insertion sort?**

i	Number of comparisons
1	n-1
2	n-2
3	n-3
.	

.	
n	1

Summation = (n-1)+(n-2)+(n-3)+...+1
= n+n+n+...+n - (1+2+3+....+n)
= n^2 - n(n+1)/2
=(2.$n^2 - n^2 - n$)/2
= (n^2 - n)/2
= $\frac{n(n-1)}{2}$
= $\Theta(n^2)$

Insertion sort also has a time complexity of $\Theta(n^{2)}$. Considering the worst case scenario for bubble sort and insertion sort, bubble sort swaps elements during each compare of 2 elements whereas insertion sort compares and swaps only when it finds the right position of the smaller element at the end of each iteration of the outer loop and thus has lesser number of swaps overall. Hence insertion sort is faster than bubble sort during worst case.

2.a. . For n = 1,2,3,4,5 what values for k and l are returned in line 7? How many multiplications ("*") does the algorithm perform for computing these values? How many additions ("+") does the algorithm perform for computing these values?

N	1	2	3	4	5
Return value (k)	8	8	8	8	8
Return value (l)	8	384	46080	10321920	3715891200
#multiplications	6	10	14	18	22
#additions	1	1	1	1	1

b. As a function of *n*, what is the value of *k* returned in line 7? Justify your results.

n	i	k
1	1	8
9	2	17
18	3	32
33	4	53
54	5	80

We try to derive a pattern from the above values.
We see that,
=> $3i^2 + 5 > n$
=> $i^2 > (n - 5)/3$
=> $i > sqrt((n - 5)/3)$

c. As a function of *n*, what is the value of *l* returned in line 7? Justify your results.

i	l = 2*l*i
1	2*1*1 = 2
2	2*(2*1*1)*2
3	2*(2*(2*1*1)*2)*3
4	2*(2*(2*(2*1*1)*2)*3)*4

Which is of the form $2^i * i! = 2^{2n} * (2n)!$

3. $2^n + 2^n$, $\text{sqrt}(2)^{n+20}$, 2^{2n}, 2^{n-20}, $2n^2+20/n$, 3^{n+30}, $n\lg(n!)$

Keeping in mind that $\omega(g(n)$ *implies* $\Omega(g(n))$ and $o(g(n)$ *implies* $O(g(n))$

Solution:

i. Consider $2^n + 2^n$ and $\text{sqrt}(2)^{n+20}$

=> Comparing 2^{n+1} *and* $2^{n/2 + 10}$

=> $\lim_{n\to\infty} \frac{2^{n+1}}{2^{n/2+10}}$ => $\lim_{n\to\infty} \frac{2^{n/2-9}}{}$ -> ∞

=> $2^{n+1} \in \omega(2^{n/2+10})$

Order : $2^{n+1} \in \Omega(2^{n/2+10})$

ii. Consider 2^{n+1} and 2^{2n}

=> Comparing 2^{n+1} *and* 2^{2n}

=> $\lim_{n\to\infty} \frac{2^{2n}}{2^{n+1}}$ => $\lim_{n\to\infty} \frac{2^{n-1}}{}$ -> ∞

=> $2^{2n} \in \omega(2^{n+1})$

Order : $2^{2n} \in \Omega(2^{n+1}) \in \Omega(2^{n/2+10})$

iii. Consider 2^{n+1} and 2^{n-20}

=> Comparing 2^{n+1} *and* 2^{n-20}

=> $\lim_{n\to\infty} \frac{2^{2n+1}}{2^{n-20}}$ => $\lim_{n\to\infty} \frac{2^{21}}{} = 2^{21} > 0$

=> $2^{n+1} \in \Theta(2^{n-20})$

Order : $2^{2n} \in \Omega(2^{n+1}) \in \Theta(2^{n-20})^{*} \in \Omega(2^{n/2 + 10})$

iv. Consider $2n^2 + 20/n$ and $2^{n/2 + 10}$

=> Comparing $2n^2 + 20/n \; and \; 2^{n/2 + 10}$

$$=> \lim_{n\to\infty} \frac{2n^2 + 20/n}{2^{n/2} . 2^{10}}$$

$$=> \lim_{n\to\infty} \frac{4n - 20/n^2}{2^9 . \ln 2 . 2^{n/2}} \quad \text{(Applying L'Hospital's rule)}$$

$$=> \lim_{n\to\infty} \frac{4 + 40/n^3}{2^8 . (\ln 2)^2 . 2^{n/2}} \quad \text{(Applying L'Hospital's rule)}$$

$$=> \lim_{n\to\infty} \frac{-120/n^4}{2^7 . (\ln 2)^3 . 2^{n/2}} \quad \text{(Applying L'Hospital's rule)}$$

$$=> \lim_{n\to\infty} \frac{-120}{n^3 . 2^7 . (\ln 2)^3 . 2^{n/2}} \quad \text{(Dividing)} \; -> 0$$

(The numerator becomes a constant whereas the denominator will have a term in terms of $2^{(n/2)}$ which makes the fraction tend to 0 .

=> $2n^2 + 20/n \in o\,(2^{n/2 + 10})$

Order : $2^{2n} \in \Omega(2^{n+1}) \in \Theta(2^{n-20}) \in \Omega(2^{n/2 + 10}) \in \Omega\,(2n^2 + 20/n)$

v. Consider 2^{n+1} and $3^{n + 30}$

=> Comparing 2^{n+1} and $3^{n + 30}$

$\Rightarrow \lim_{n\to\infty} \frac{2^{n+1}}{3^{n+30}} \quad \Rightarrow \lim_{n\to\infty}[\frac{2}{3^{30}} \quad (2/3)^n] \to 0$ (As $(2/3)^n \to 0$)

$\Rightarrow 2^{n+1} \in o(3^n + 30)$

Now consider 2^{2n} and 3^{n+30}

=> Comparing 2^{2n} and 3^{n+30}

$\Rightarrow \lim_{n\to\infty} \frac{2^{2n}}{3^{n+30}} \quad \Rightarrow \lim_{n\to\infty} \frac{2^{2n}}{3^n 3^{30}}$

$\Rightarrow \lim_{n\to\infty} [\frac{1}{3^{30}} \quad (4/3)^n] \to \infty$ (As $(4/3)^n \to \infty$)

$\Rightarrow 2^{2n} \in \omega(3^n + 30)$

Order : $2^{2n} \in \Omega(3^n + 30) \in \Omega(2^{n+1}) \in \Theta(2^{n-20}) \in \Omega(2^{n/2+10}) \in \Omega(2n^2 + 20/n)$

vi. Consider $2n^2 + 20/n$ and nlogn(n!)

=> Comparing $2n^2 + 20/3$ *and* $nlogn(n!)$

$\Rightarrow \lim_{n\to\infty} \frac{2n^2+20/n}{nlogn(n!)} = \lim_{n\to\infty} \frac{2n+20/n^2}{logn(n!)}$

$\Rightarrow \lim_{n\to\infty} \frac{2n+20/n^2}{log(\sqrt{2.\pi})+(n+1/2)logn-nloge}$ (Stirling's approximation)

$\Rightarrow \lim_{n\to\infty} \frac{2-40/n^3}{\frac{(n+1/2)}{n\,ln2}+logn-loge}$ (Applying L'Hospital's rule)

$\Rightarrow \lim_{n\to\infty} \frac{(n*ln2)(2-40/n^3)}{n+1/2+n*logn*ln2-n*loge*ln2}$ (Dividing)

$\Rightarrow \lim_{n\to\infty} \frac{(ln2)(2n-40/n^2)}{n+1/2+n*logn*ln2-n*loge*ln2}$ (Dividing)

$$=> \lim_{n\to\infty} \frac{(ln2)(2 + 80/n^3\ )}{1+logn*ln2+1-loge*ln2} \text{ -> } 0$$ (Applying L'Hospital's rule)

When we apply limits to the above equation, the denominator evaluates to 0, and 80/n^3 -> 0 as n ->infinity. Therefore numerator will be a constant. Therefore the limit will -> 0.

=> $f(n) \in o\ (g(n))$

Consider $2^{n/2 + 10}$ and nlogn(n!)

=> Comparing $2^{n/2 + 10}$ *and nlogn(n!)*

$$=> \lim_{n\to\infty} \frac{2^{n/2 + 10}}{nlogn(n!)} = \lim_{n\to\infty} \frac{2^{n/2}\cdot 2^{10}}{nlogn(n!)}$$

$$=> \lim_{n\to\infty} \frac{2^{10}.\ 2^{n/2}.}{nlog(\sqrt{2.\pi})+(n^2+n/2)logn-n^2loge}$$ (Stirling's approximation)

$$=> \lim_{n\to\infty} \frac{2^{9}.\ 2^{n/2}\cdot ln2}{log(\sqrt{2.\pi})+\frac{(n^2+n/2)}{nln2}+(1/2 + 2n)logn-2nloge}$$ (Applying L'Hospital's rule)

$$=> \lim_{n\to\infty} \frac{2^{8}.\ 2^{n/2}\cdot (ln2)^2}{\frac{(2n+1/2)}{nln2}+2logn + 1/ln2 - 2loge}$$ (Applying L'Hospital's rule)

$$=> \lim_{n\to\infty} \frac{ln2.\ \ n(2^{8}.\ 2^{n/2}\cdot (ln2)^2)}{2n.ln2(logn-loge)+1/2+3n}$$ (Dividing)

$$=> \lim_{n\to\infty} \frac{ln2\ \ [\ n(2^{7}.\ 2^{n/2}\cdot (ln2)^3)+(2^{8}.\ 2^{n/2}\cdot (ln2)^2)]}{2.ln2(logn-loge)+5}$$ (Applying L'Hospital's rule)

$$=> \lim_{n\to\infty} \frac{ln2\ \ [\ n(2^{7}.\ 2^{n/2}\cdot (ln2)^3)+(2^{8}.\ 2^{n/2}\cdot (ln2)^2)]'}{2.(1/n)}$$ (Applying L'Hospital's rule)

(Applying L'Hospital's rule, ' indicates derivative)

$$=> \lim_{n\to\infty} \frac{ln2\ n\ [\ n(2^{7}.\ 2^{n/2}\cdot (ln2)^3)+(2^{8}.\ 2^{n/2}\cdot (ln2)^2)]'}{2} \text{ –> } \infty$$

$$=> 2^{n/2 + 10} \in \omega\ nlogn(n!)$$

Order: $2^{2n} \in \Omega(3^n + 30) \in \Omega(2^{n+1})\in\Theta(2^{n-20})^* \in \Omega(2^{n/2 + 10}) \in \Omega(nlogn(n!) \in \Omega(2n^2 + 20/n)$

3b. $(n+2)!$, $\lg(n^{1.9})$, $1/n$, $\lg(n)$, $n^2\lg(n^2)$, $n^{1.9}\lg(n^4)$, $(n+1)!$

Solution:

vii. Consider $\log n^{1.9}$ and 1/n

=> Comparing $\log n^{1.9}$ *and* $1/n$

=> $\lim_{n\to\infty} \frac{\log n^{1.9}}{1/n}$ => $\lim_{n\to\infty} \frac{1.9(nlogn)}{}$ -> ∞

=> $\log n^{1.9} \in \omega(1/n)$

Order : $(\log n^{1.9}) \in \Omega(1/n)$

viii. Consider $\log n$ and 1/n

=> Comparing $\log n$ *and* $1/n$

=> $\lim_{n\to\infty} \frac{\log n}{1/n}$ => $\lim_{n\to\infty} \frac{nlogn}{}$ -> ∞

=> $\log n \in \omega(1/n)$

Now consider, $\log n$ and $log\, n^{1.9}$

=> Comparing $\log n$ *and* $\log n^{1.9}$

=> $\lim_{n\to\infty} \frac{\log n^{1.9}}{log\, n}$ => $\lim_{n\to\infty} \frac{1.9 \,.\, \log n}{log\, n}$ = 1.9 > 0

=> $\log n \in \Theta(log\, n^{1.9})$

Order : $\Omega(\log n^{1.9}) \in \Theta(log\ n)^* \in \Omega(1/n)$

ix. Consider $\log n$ and $n^2 \log n^2$

=> Comparing $\log n$ *and* $n^2 \log n^2$

$$=> \lim_{n\to\infty} \frac{n^2 \log n^2}{log\ n} = \lim_{n\to\infty} \frac{2n^2 \log n}{log\ n} = \lim_{n\to\infty} \frac{2n^2}{} \text{ -> } \infty$$

$$=> n^2 \log n^2 \in \omega(\log n)$$

Order : $n^2 \log n^2\ \Omega(\log n^{1.9}) \in \Theta(logn)^* \in \Omega(1/n)$

x. Consider $n^{1.9} \log n^4$ and $n^2 \log n^2$

=> Comparing $\log n$ *and* $n^2 \log n^2$

$$=> \lim_{n\to\infty} \frac{n^2 \log n^2}{n^{1.9} \log n^4} = \lim_{n\to\infty} \frac{2.n^2 \log n}{4n^{1.9} \log n} = \lim_{n\to\infty} \frac{n^{0.1}}{2} \text{ -> } \infty$$

$$=> n^2 \log n^2 \in \omega(n^{1.9} \log n^4)$$

Now consider, $n^{1.9} \log n^4$ and $\log n$

=> Comparing $\log n$ *and* $n^{1.9} \log n^4$

$$=> \lim_{n\to\infty} \frac{n^{1.9} \log n^4}{\log n} = \lim_{n\to\infty} \frac{4.\ n^{1.9} \log n}{\log n} = \lim_{n\to\infty} \frac{4n^{1.9}}{} \text{ -> } \infty$$

$$=> n^{1.9} \log n^4 \in \omega(\log n)$$

Order : $n^2 \log n^2 \in \Omega(n^{1.9} \log n^4) \in \Omega(\log n^{1.9}) \in \Theta(log\ n)^* \in \Omega(1/n)$

xi. Consider $n^2 \log n^2$ and (n+1)!

If $n^2 \log n^2 \in O$(n+1)!, iff there exist c, n_0 > 0 such that $0 \leq n^2 \log n^2 < c$*(n+1)! for all n ≥ n_0

$0 \leq 2n^2 \log n \leq 2n^3$

(n>=1) (n>=1)

=> $2n^3 \leq 2\ n\ !$ (factorials grow faster than exponents as exponents are multiplied by a constant as n-> inf whereas factorials are multiplied by n)

(n>=6)

=> $n! \leq$ (n+1)! ≤ (n+1)n!

(n>=1)

=> $(n+1)! \in O(n^2 \log n^2)$, for c=n+1 and $n_0 \geq 6$

Order : $(n+1)! \in \Omega\ (n^2 \log n^2) \in \Omega\ (n^{1.9} \log n^4) \in \Omega\ (\log n^{1.9}) \in \Theta\ (log\ n)^* \in \Omega\ (1/n)$

xii. Consider (n+2)! And (n+1)!

=> Comparing $(n+2)!\ and\ (n+1)!$

=> $\lim_{n\to\infty} \frac{(n+2)!}{(n+1)!}$ => $\lim_{n\to\infty} \frac{(n+2)}{}$ -> ∞

=> $(n+2)! \in \omega\ (n+1)!$

Order :
$(n+2)! \in \Omega(n+1)! \in \Omega\ (n^2 \log n^2) \in \Omega\ (n^{1.9} \log n^4) \in \Omega\ (\log n^{1.9}) \in \Theta\ (log\ n)^* \in \Omega\ (1/n))$

4.a $n\lg(n)+n \in o(n)$.

Solution:

$n\lg(n)+n \in o(n)$ if for every c > 0 there is an n_0 > 0 for all n ≥ n_0 such that

$0 \le n\lg(n)+n < c*n$

Let us assume $n\lg(n)+n \in o(n)$

Proof :

$$\underset{(n>=1)}{0 \le} n\lg(n)+n \underset{(n>=1)}{<} n^2 + n$$

=> $n^2 + n < c*n$

(Dividing by n throughout)

=> n+1 < c

=> c on the RHS is a constant and hence bound whereas n+1 on the LHS is Unbound.

=> f(n) $\notin o(n)$ by contradiction

4b. $3n^2 - \sin(n) - n \in \Theta(n^2)$.

Solution:

$3n^2 - \sin(n) - n \in O(n^2)$ iff there exist c, n_0 > 0 for all n ≥ n_0 such that

$0 \le 3n^2 - \sin(n) - n \le c*n^2$

Proof :

$$\underset{(n>=1)}{0 \le} 3n^2 - \sin(n) - n \underset{(n>=1)}{\le} 3n^2 + 1 - n$$

$$\Rightarrow 3n^2 - n \underset{(n>=1)}{\le} 3n^2 \quad \text{(dropping constant)}$$

$$\Rightarrow 3n^2 \underset{(n>=1)}{\le} c.\, n^2 \quad \text{(dropping negative term)}$$

=> $3n^2 - \sin(n) - n \in O(n^2)$, for c=3 and $n_0 \ge 1$

$3n^2 - \sin(n) - n \in \Omega(n^2)$ iff there exist c, $n_0 > 0$ for all n ≥ n_0 such that

$3n^2 - \sin(n) - n \geq c * n^2 \geq 0$

Proof :

$$3n^2 - \sin(n) - n \underset{(n>=1)}{\geq} 3n^2 - \sin(n) - n^2$$

$$=> 2n^2 - \sin(n) \underset{(n>=1)}{\geq} 2n^2 - 1 \quad \text{(max value of sinx = 1)}$$

$$=> 2n^2 \underset{(n>=1)}{\geq} c.\, n^2 \geq 0 \text{(drop constant as it is with an unbounded value)}$$

$$=> 3n^2 - \sin(n) - n \in \Omega(n^2), \text{ for c=2 and } n_0 \geq 1$$

=> Since f(n) $\in \Omega$(g(n)) and f(n) $\in$ O(g(n))

=> f(n) $\in \Theta(n^2)$.

4c. $\sum_{i=1}^{3n} i \in O(\text{sqrt}(n))$

Solution:

$\sum_{i=1}^{3n} i \in O(\text{sqrt}(n))$ iff there exist c, $n_0 > 0$ for all n ≥ n_0 such that

$0 \leq \sum_{i=1}^{3n} i \leq c * \text{sqrt}(n)$, $\sum_{i=1}^{3n} i = \frac{3n(3n+1)}{2} = 9n^2/2 + 3n/2$

Proof :

$$0 \leq 9n^2/2 + 3n/2 \leq 9n^2/2 + 3n^2/2 \leq 6n^2$$

(n>=1)   (n>=1)   (n>=1)

=> 6 $n^2 \leq$ c. $\sqrt{n}$
=> 6. n. $\sqrt{n} \leq$ c   (Dividing by n)

=> Since RHS is a constant and bound whereas LHS is a monotone increase and hence unbound

=> f(n) $\notin O(n)$ by contradiction

4d. $n\lg(n)+n^2 \in \omega(n)$ (little-omega)

Solution :

$n\lg(n)+n^2 \in \omega(n)$ if for every c > 0 there is an $n_0 > 0$, for all n $\geq n_0$ such that $n\lg(n)+n^2 > c*n >= 0$

=> $n\lg(n)+n^2 > n + n^2$ (taking logn = 2 (as min value))
(n>=3)

=> $n^2 + n > n^2$
` (n>=1)

=> $n^2 > c.n \geq 0$, for all n > c+3
(n>=1)

=> $n\lg(n)+n^2 > c*n^2 > 0$, for all n > c+3

=> $n\lg(n)+n^2 \in \omega(n)$

5. Purpose: Practice how to design, analyze, and communicate algorithms. Describe an non-recursive Θ (lg n) algorithm which computes (3a)n/2, given a and n. You may assume that a is a positive real number, and n a positive integer, but do not assume that n is a power of 2. Please follow the above instructions for describing your algorithm. Please give both, a textual

description and pseudocode of your algorithm, and make sure that you justify the asymptotic running time of your algorithm.

Pseudocode:

```
Def computer_power(a,n):
        a <- 3a

        If (n mod 2 != 0):
        {
                x <- (n-1)/2
                r <-  sqrt(a)

        }
        else
        {
                x <- n/2
                r <- 1
        }

        while (x>0):
        {
                If ( x mod 2 !=0):
                {
                        r <- r * a
                }
                x <- integer ( x/2)
                a <- a*a
        }
```

Description:

To compute : (3a)^(n/2)

1. Let us initialize a as 3a.
   => a <- 3a

2. Let us consider 2 cases:

a. When n is even :
n can be represented as n=2x
=> $a^{\frac{n}{2}} = a^{\frac{2x}{2}} = a^{x}.1$
=> x= n/2
=> res = 1
(We initialize the result res with a 1 (as a part of $a^{x}.1$) and now need to compute a^{x} .)

b. When n is odd :
n can be represented as n=2x + 1

=> $a^{\frac{n}{2}} = a^{\frac{2x+1}{2}} = a^{x}.a^{\frac{1}{2}}$
=> x= (n-1)/2
=> res = $a^{\frac{1}{2}}$
(We initialize the result res with a $a^{\frac{1}{2}}$ (as a part of $a^{x}.a^{\frac{1}{2}}$) and now need to compute a^{x} .)

3. Now, the problem can be redefined as a power function where we find a^{x}
4. Let us consider 2 cases :
   a. If x is even:
      - Consider an example a^8
      - We can write this as $(a)^{8} = (a^2)^{4} = [(a^2)(.a^2)]^{2} = [(a^2.a^2).(a^2.a^2)]^{1}$
      - This is basically replacing the base with its square while reducing the power by 2 as seen in the last 2 lines..
      - Therefore, $a^{x} = (a^2)^{n/2}$
   b. If x is odd
      - $a^x = a.a^{x-1} = a.(a^2)^{\frac{(n-1)}{2}}$
      - This breaks down the problem into the subproblem with an even power.
      - Consider an example of a^7
      - We can write this as $(a)^{7} = a(a^2)^{3} = a\{a^2[(a^2)(.a^2)]\}^{1}$
      - Intuitively, we split the odd power as x.x^(even power) and apply the algorithm to find even power as given above.

5. Therefore, to computer a^{n} **:**

a. $a^{\underline{n}} = x(x^2)^{\frac{(n-1)}{2}}$, if n is odd

b. $a^{\underline{n}} = (x^2)^{\frac{(n)}{2}}$, if n is even

Example :

a=2, n=7

1. To compute (3.2)^(7/2)
2. a <- 3.2
   => a=6
3. n = 7 which is odd
   => r= sqrt(a)
   => x= (n-1)/2 = (7-1)/2 = 3
4. We enter the loop.

   a. x = 3 > 0
      x = 3 is odd and hence
      => r=sqrt(a)*2
      => x=int(3/2) = 1
      => a = 2*2 = 4

   b. x = 1 > 0
      x = 1 is odd and hence
      => r=[sqrt(a)*2]*4
      => x=int(1/2) = 0
      => a = 4*4 = 16

   c. x= 0 and while loop exits
   d. We have our computed value in r = sqrt(a)*2*4 =value of 8*sqrt(a)

Proof of correctness:

Loop invariant -

At any given point of time a^n can be obtained at that instant by raising a to power of x and multiplying the result by r.

Before we enter the loop, the values are initialized as a<-3a and x and r values depending on n being odd or even and do not alter the loop.

The loop computer a^x

Initialization -
Initially, r=1 or sqrt(a) and a=a, n=x.

Therefore, r* (a^x) is proved with the initial values, trivially.

Maintenance -
We can see that the power x is divided by 2 for every iteration, while the base is being squared. Therefore, we can say that the loop invariant holds.

Let us consider the i'th iteration of the loop.

- If x at i'th iteration is even, the if condition is skipped and x = x/2 and a = a^2 and r=1. If we compute the value at this iteration, we get $1.a^{\frac{2.x}{2}} = 1.\ a^x$
- If x at i'th iteration is odd, the if condition is skipped and x = x/2 and a = a^2 and r=sqrt(a). If we compute the value at this iteration, we get $sqrt(a).a^{\frac{2.x}{2}} = sqrt(a).\ a^x$

At i'th iteration
$a^n = r * (a_i)^x$

For (i+1)th iteration
$r * (a_{i+1})^{x_{i+1}} = r * (a_i)^{2 * x/2} = r * (a_i)^x$

Termination -
The loop terminates when x <= 0.
The values just before termination would be -
r= a^n
x=0
a=a'
=> value = $r * (a')^x = (a)^n.(a')^0 = (a)^n$

Running time analysis

We see that the condition determining how long the loop runs depends on the values of x. We observe that x is halved during every iteration of the loop. The values of x are x, x/2, x/4, …. , x/x.

The loop runs till $\frac{x}{2^k} < 1$
=> x< 2^k

=> $\log x < klog2$ (binary log)
=> k = log n (binary log)

Therefore the asymptotic running time complexity is $\Theta(logn)$ (binary log)