Homework 1

Md Kamrul Hasan Khan

February 17, 2017

Answer to the Question Number 1

```
(x<- c(9:16))
```

```
## [1]  9 10 11 12 13 14 15 16
```

```
tail(x,3)
```

```
## [1] 14 15 16
```

```
x[(x%%2 == 0)]
```

```
## [1] 10 12 14 16
```

```
x[-which(x%%2 == 0)]
```

```
## [1]  9 11 13 15
```

Answer to the Question Number 2

```
n = 25
# for loop

sum = 0
for (i in 1:n)
{
  sum = sum + (0.5^i)
}
sum
```

```
## [1] 1
```

```
# while loop

sum = 0
i <- 1
while (i <= n)
{
  sum = sum + (0.5^i)
  i = i+1
}
sum
```

```
## [1] 1
```

```
# without using a loop
```

Sum of the Geometric series:

$$\sum_{i=1}^{n} a^i = a\Big(\frac{1-r^n}{1-r}\Big)$$

```
n = 25
a = 1/2
r = 1/2
```

```
sum <- a*((1-r^n)/(1-r))
sum
```

```
## [1] 1
```

As n increases the sum tends to 1.

If n is large, without using loop program will be more robust to errors.

Answer to the Question Number 3

Question I: x^3 is $O(x^3)$ and $\Theta(x^3)$ but not $\Theta(x^4)$

Answer:

$$\lim_{x\to\infty} \frac{x^3}{x^3} = 1 > 0$$

So, x^3 is $O(x^3)$ and $\Theta(x^3)$.

$$\lim_{n\to\infty} \frac{x^3}{x^4} = \lim_{n\to\infty} \frac{1}{x} = 0$$

So, x^3 is not $\Theta(x^4)$.

Question II: For any real constants a and $b > 0$, we have $(n+a)^b = \Theta(n^b)$

Answer:

$$\lim_{n\to\infty} \frac{(n+a)^b}{n^b} = \lim_{n\to\infty} \left(\frac{n+a}{n}\right)^b = \lim_{n\to\infty} \left(1 + \frac{a}{n}\right)^b = 1 > 0$$

So, $(n+a)^b = \Theta(n^b)$.

Question III: $(log(n))^k = O(n)$ for any k

Answer:

$$\lim_{n\to\infty} \frac{(log(n))^k}{n} = 0$$

So, $(log(n))^k = o(n) = O(n)$

Question IV: $\frac{n}{n+1} = 1 + O(\frac{1}{n})$

Answer: $\frac{n}{n+1} = 1 + O(\frac{1}{n})$

$\Rightarrow \frac{n}{n+1} - 1 = O(\frac{1}{n})$

$\Rightarrow \frac{-1}{n+1} = O(\frac{1}{n})$

Now $\frac{-1}{n+1} = O(\frac{1}{n+1})$, thus $\frac{-1}{n+1} = O(\frac{1}{n})$

Question V: $\sum_{i=0}^{\lceil log_2(n) \rceil} 2^i$ is $\Theta(n)$

Answer: Let $log_2(n) = k$

Now

$$\sum_{i=0}^{k} 2^i = \frac{2^{k+1} - 1}{2 - 1} = 2 * 2^k - 1 = 2 * 2^{log_2(n)} - 1 = 2n - 1$$

So,

$$\lim_{n\to\infty} \frac{2n-1}{n} = \lim_{n\to\infty} (2 - \frac{1}{n}) = 2 > 0$$

Therefore, $\sum_{i=0}^{\lceil log_2(n) \rceil} 2^i$ is $\Theta(n)$

Answer to the Question Number 4(a)

```r
selsort <- function(A)
{
  n = length (A)
  for (i in 1:(n-1))
  {
    index = i
    for (j in (i+1):n)
    {
      if (A[j] < A[index])
        index = j
    }
    temp = A[i]
    A[i] = A[index]
    A[index] = temp
  }
  return (A)
}

x = sample(1:100, 10, replace = TRUE)
x
```

```
##  [1] 55 58 90 16 27 43 86  8 67 74
```

```r
a <- selsort(x)
a
```

```
##  [1]  8 16 27 43 55 58 67 74 86 90
```

Answer to the Question Number 4(b)

```r
n = 100
y = rnorm (n)

# Mergesort code
ptm <- proc.time ()
mergearrays <- function(x,y){
  m = length(x)
  n = length(y)
  if(m==0){
    return(z = y)
    }
  if(n==0){
    return(z = x)
  }
  if (x[1]<=y[1]){
    return(z = c(x[1],mergearrays(x[-1],y)))
  }else{
    return(z = c(y[1],mergearrays(x,y[-1])))
  }
}
mergesort <- function(x){
  n = length(x)
  mid = floor(n/2)
  if(n > 1){
    return(mergearrays(mergesort(x[1:mid]),mergesort(x[(mid+1):n])))
```

```r
  }else{
    return(x)
  }
}
merge<-mergesort(y)
time_mergesort = proc.time () - ptm
summary(time_mergesort)
```

```
##    user  system elapsed
##       0       0       0
```

```r
# Bubble sort code
ptm <- proc.time ()
bubblesort <- function(A)
{
  n = length (A)
  repeat
  {
    swapped = FALSE
    for (i in 1:(n-1))
    {
      if (A[i] > A[i+1])
      {
        temp = A[i]
        A[i] = A[i+1]
        A[i+1] = temp
        swapped = TRUE
      }
    }
    if (swapped == FALSE)
      break
  }
  return (A)
}
bubble<- bubblesort(y)
time_bubblesort = proc.time () - ptm
summary(time_bubblesort)
```

```
##    user  system elapsed
##    0.03    0.00    0.04
```

Answer to the Question Number 4(c)

```r
time_mergesort = rep (0 ,500)
time_bubblesort = rep (0 ,500)
for ( i in 1:500)
{
  ptm <- proc.time ()
  mergesort ( y )
  t1 = proc.time () - ptm
  time_mergesort [i] = t1 [["elapsed"]]
  ptm <- proc.time ()
  bubblesort ( y )
  t2 = proc.time () - ptm
  time_bubblesort [i]= t2 [["elapsed"]]
```

```r
}
summary(time_mergesort)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
## 0.00000 0.00000 0.00000 0.00358 0.00000 0.02000
```

```r
summary(time_bubblesort)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
## 0.00000 0.01000 0.02000 0.01656 0.02000 0.07000
```

Time complexities of mergesort and bubblesort are $O(nlog(n))$ and $O(n^2)$ respectively. So, for large n mergesort will take much much less time than bubblesort, which agrees the summary of the results.