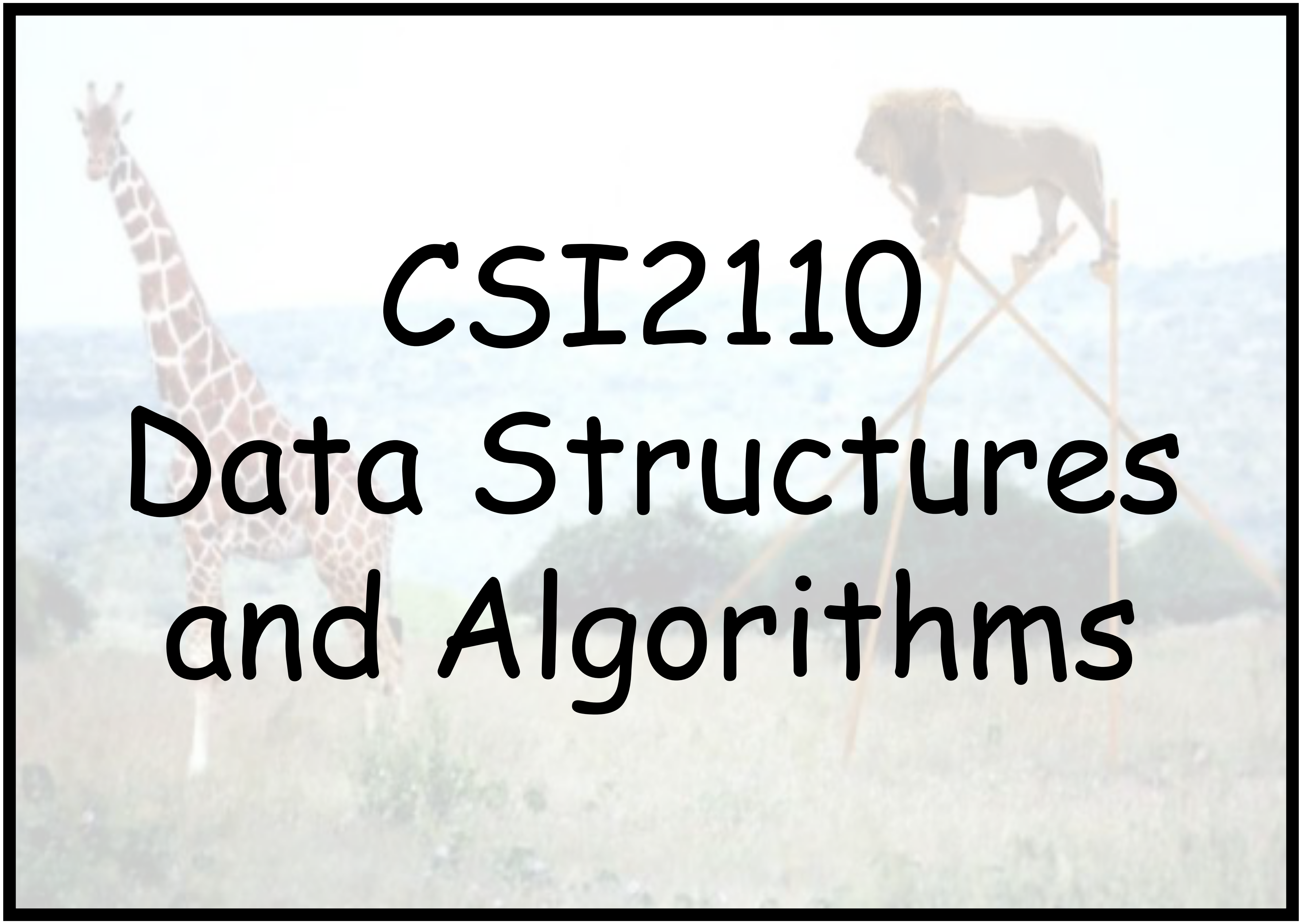

CSI2110 Data Structures and Algorithms

Ex1.

Give a big-Oh characterization, in terms of *n*, of the running time of the following method. Show your analysis!

```
public void Ex(int n)
    int a ;
    for (int i = 0 ; i < n*n ; i++)
            for (int j = 0; j <= i; j++)
                    a = i
}
```

Give a big-Oh characterization, in terms of *n*, of the running time of the following method. Show your analysis!

```
public void Ex(int n)
      int a ;
      for (int i = 0 ; i < n*n ; i++)
               for (int j = 0; j <= i; j++)
                        a = i
}
```

the inner loop j goes from 1 to i

the outer loop i goes from 1 to $n*n=n^2$

the total for the two loops will be $1+2+3 + \ldots + n^2$

which equals $n^2(n^2+1)/2= (n^4+ n^2)/2 < (n^4+ n^4)/2 = n^4 = O(n^4)$

The Ex method runs in $O(n^4)$ time.

Ex2.

Prove that 2^{n+1} is $O(2^n)$ using definition of O.

Prove that 2^{n+1} is $O(2^n)$ using definition of O.

By the definition of big-Oh, we need to find a real constant $c > 0$ and an integer constant $n_0 \geq 1$ such that $2^{n+1} \leq c(2^n)$ for $n \geq n_0$. One possible solution is choosing $c = 2$ and $n_0 = 1$.

Ex3.

In big-Oh notation, what is the running time of the following code fragment:

```
public void EX5(int n)
{
int a=0;
for (int i=0; i< n; i++)
    for(int j=0; j<=i; j++)
        for( int k=0; k<=n; k++)
            a+=i;
}
```

Show your analysis!

```
public void EX5(int n){

int a=0;

for (int i=0; i< n; i++)

for(int j=0; j<=i; j++)

for( int k=0; k<=n; k++)

a+=i; }
```

Solution:

Initializing the integer variable “a” at the beginning of the code fragment takes *O* (1) time.

There are three nested loops in this code fragment. The inner most loop, controlled by counter k adds O(n) operations every time it is executed. But how often is it executed?

Ex4.

Describe how to implement the stack ADT using two queues. That is: write pseudocode algorithms which implement the push() and pop() methods of the stack using the methods of the queue. What are the running times of your push () and pop () algorithms?

Describe how to implement the stack ADT using two queues. That is: write pseudocode algorithms which implement the push() and pop() methods of the stack using the methods of the queue. What are the running times of your push () and pop () algorithms?

Assume that we have two queues named ***Q1*** and ***Q2***.
Q1 will have store the stack elements assuming the front element is the top of the stack. ***Q2*** will be used as temporary storage when pushing new element.

When we want to pop an element, we can just perform dequeue on ***Q1*** (Thus we are taking the top of the stack). While when we want to push an element ***o***, we should dequeue all the elements from ***Q1*** and enqueue them in ***Q2***, then enqueue the element ***o*** in ***Q1*** and then dequeue all the elements from ***Q2*** and enqueue them in ***Q1***. The code follows:

Pseudocode algorithms which implement the push() and pop() methods of the stack using the methods of the queue.

```
Algorithm Pop()
If Q1.isEmpty()  then
        ERROR
Return Q1.dequeue()


Algorithm push(object o)
While ! Q1.isEmpty
      Q2.enqueue(Q1.dequeue())

Q1.enqueue(o)

While ! Q2.isEmpty
      Q1.enqueue(Q2.dequeue())
```

Pop will be O(1)
Push will be O(n)