CSE101 Homework 1

Shu-Wei Hsu, Andrew Ho

Spring 2017

Question 1.

(a) *False.* 3^n *grows way faster than* 2^n*. Hence, it is slower. Therefore,* $2(3^n) \in \Omega(3(2^n))$

(b) *True. The highest power in both equation is 12.*

(c) *False.* n^{10} *grows way faster than n. Hence, log(*n^{10}*) grows way faster than log(n). Hence, log(*n^{10}*)* $\in \Omega(log(n))$

(d) *False.* $\sum_{i=1}^{n} i^k = 1^k + 2^k + \cdots + n^k$*. The highest power term is* n^k*. Hence,* $\sum_{i=1}^{n} i^k \in O(n^{k+1})$

(e) *False. n! grows way faster than* 2^n*. Hence, we should have* $n! \in \Omega(2^n)$

Question 2.

(a) *Base case: n=6 From the definition of Fibonacci numbers, we can calculate the following:*

$$F_2 = F_1 + F_0 = 1 + 0 = 1, F_3 = F_2 + F_1 = 1 + 1 = 2, F_4 = F_3 + F_2 = 1 + 2 = 3$$

$$F_5 = F_4 + F_3 = 2 + 3 = 5, F_6 = F_5 + F_4 = 8$$

We know that $2^{0.5*6} = 2^3 = 8$*. Hence, the base case is true.* $F_n \geqslant 2^{0.5n}$ *when* $n = 6$*. Assume* $F_n \geqslant 2^{0.5n}$ *for all* $n \geqslant 6$*. We need to show that this is true for the case* $n + 1$*. By the definition, we have the following:*

$$\begin{aligned} F_{n+1} &= F_n + F_{n-1} && (1)\\ &\geqslant 2^{0.5n} + 2^{(0.5(n-1))} && (2)\\ &= 2^{0.5n} + 2^{0.5n-0.5} && (3)\\ &= 2^{0.5n} + \frac{2^{0.5n}}{2^{0.5}} && (4)\\ &= 2^{0.5n}(1 + 2^{-0.5}) && (5) \end{aligned}$$

Since $(1 + 2^{-0.5}) > 2^{0.5n}$*, we get the following:*

$$\begin{aligned} F_{n+1} &\geqslant 2^{0.5n}(1 + 2^{-0.5}) && (6)\\ &\geqslant 2^{0.5n}(2^{0.5}) && (7)\\ &\geqslant 2^{0.5(n+1)} && (8) \end{aligned}$$

Hence, we have shown that $F_n \geqslant 2^{0.5n}$ *for all* $n \geqslant 6$

(b) *Base Case: n=0 From the definition of Fibonacci numbers, we can calculate the following:* $F_0 = 0$*,* $2^0 = 1$*. Hence, base case is true.* $F_n \leqslant 2^n$ *when n = 0. Assume* $F_n \leqslant 2^n$ *for all* $n \geqslant 0$*. We need to* *show that this is true for the case $n + 1$. By the definition, we have the following:*

$$
\begin{aligned}
F_{n+1} &= F_n + F_{n-1} && (9)\\
&\leqslant 2^n + 2^{n-1} && (10)\\
&= 2^n + \frac{2^n}{2} && (11)\\
&= 2^n(1 + \frac{1}{2}) && (12)\\
&\leqslant 2(2^n) && (13)\\
&\leqslant 2^{(n+1)} && (14)
\end{aligned}
$$

Hence, we have shown that $F_n \leqslant 2^n$ for all $n \geqslant 0$

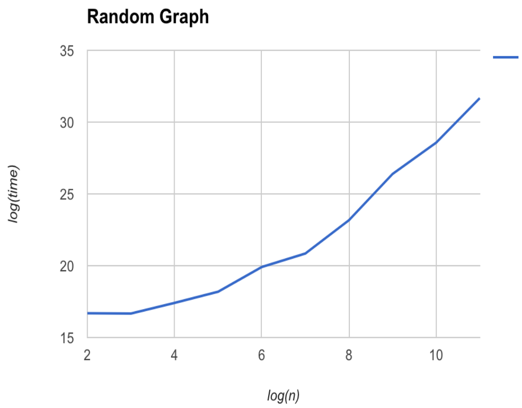
(c) We conclude that F_n grows exponentially an it is between $\Omega(2^{0.5n})$ and $O(2^n)$.

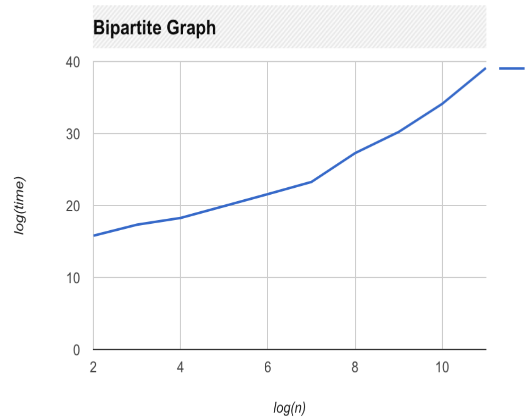
Question 3.

```
public boolean existTriangle2(int [][] arr)
HashSet¡Pair¿ set = new HashSet¡Pair¿();
ArrayList¡Pair¿ edges = new ArrayList¡Pair¿();

int index = 1;
for each row in arr:

    for each column from 0 to index:
    if (arr[i][j] == 1):
    Pair p = new Pair(i,j);
    Pair p1 = new Pair(j,i);
    edges.add(p);
    set.add(p);
    set.add(p1);

index++;
//iterate through edges array
for each edge in array edges: Pair edge = edges.get(i);

    for each vertex:
    int xval = edge.x;
    int yval = edge.y;
    Pair p = new Pair(xval,vertex);
    Pair p1 = new Pair(yval,vertex);
    if set.contains(p) and set.contains(p1):
    return true;

return false;
```

Our algorithm has $O(n^2)$ for worst case running time because we need to iterate the whole n by n matrix. In terms of node and number of edges, it should have a worst case of $O(nm)$ where n is number of nodes and m is number of edges. This is because for each edge, we need to iterate through all vertices and check if a triangle exists.

Question 4. *Below are the running time graphs*





We see that the run time for bipartite graphs is longer than a randomly generated graph. This is because bipartite graphs represent the worse case in which we can never terminate the algorithm early because there are no triangles.