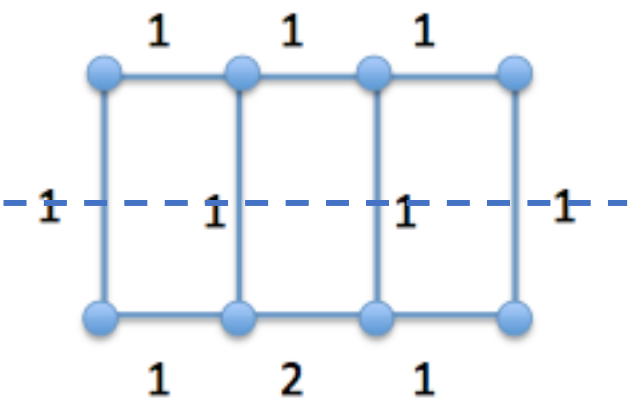
CSCI 109 Introduction to Computer Science HW2

1. $\text{Edges} = \frac{3\text{m}+2(\text{N}-\text{m})}{2} = N + \frac{m}{2}$
   Because each edge contributes 2 to the sum of degrees. Therefore, by calculating the sum of edges, the sum of degrees on the right side should divide by 2.

2. $(k-1)$ edges

3. by using the following way of partitioning, the MST of upper first half is 3, and MST of lower half is 4. And by adding up MST of these two halves, we have 7. Also, we have to add the smallest edge weight between these two halves, which would make the overall MST larger than 7. However, we know, by either of two methods, overall MST is 7. So, the roommate's algorithm is not right.



4. (a).

The initial array.
The first element is skipped, so we begin with 9.

13	9	6	3	15	2	21	4	16	8	11

Compare 13 to 9; 9 < 13, so the two values are swapped.
Number of comparisons: 1

9	**13**	6	3	15	2	21	4	16	8	11

Compare 13 to 6; 6 < 13, so the two values are swapped.
Compare 9 to 6; 6 < 9, so the two values are swapped.
Number of comparisons: 2

6	**9**	**13**	3	15	2	21	4	16	8	11

Compare 13 to 3; 3 < 13, so the two values are swapped.
Compare 9 to 3; 3 < 9, so the two values are swapped.
Compare 6 to 3; 3 < 6, so the two values are swapped.
Number of comparisons: 3

3	**6**	**9**	**13**	**15**	2	21	4	16	8	11

Compare 15 to 13; 13 < 15, so there is no swap.
Number of comparisons: 1

3	**6**	**9**	**13**	**15**	2	21	4	16	8	11

Compare 15 to 2; 2 < 15, so the two values are swapped.
Compare 13 to 2; 2 < 13, so the two values are swapped.
Compare 9 to 2; 2 < 9, so the two values are swapped.
Compare 6 to 2; 2 < 6, so the two values are swapped.
Compare 3 to 2; 2 < 3, so the two values are swapped.
Number of comparisons: 5

2	**3**	**6**	**9**	**13**	**15**	21	4	16	8	11

Compare 21 to 15; 15 < 21, so there is no swap.
Number of comparisons: 1

2	**3**	**6**	**9**	**13**	**15**	21	4	16	8	11

Compare 21 to 4; 4 < 21, so the two values are swapped.
Compare 15 to 4; 4 < 15, so the two values are swapped.
Compare 13 to 4; 4 < 13, so the two values are swapped.
Compare 9 to 4; 4 < 9, so the two values are swapped.
Compare 6 to 4; 4 < 6, so the two values are swapped.
Compare 4 to 3; 3 < 4, so there is no swap.
Number of comparisons: 6

2	**3**	**4**	**6**	**9**	**13**	**15**	21	16	8	11

Compare 21 to 16; 16 < 21, so the two values are swapped.
Compare 16 to 15; 15 < 16, so there is no swap.
Number of comparisons: 2

2	**3**	**4**	**6**	**9**	**13**	**15**	**16**	21	8	11

Compare 21 to 8; 8 < 21, so the two values are swapped.
Compare 16 to 8; 8 < 16, so the two values are swapped.
Compare 15 to 8; 8 < 15, so the two values are swapped.
Compare 13 to 8; 8 < 13, so the two values are swapped.
Compare 9 to 8; 8 < 9, so the two values are swapped.
Compare 8 to 6; 6 < 8, so there is no swap.
Number of comparisons: 6

2	**3**	**4**	**6**	**8**	**9**	**13**	**15**	**16**	21	11

Compare 21 to 11; 11 < 21, so the two values are swapped.
Compare 16 to 11; 11 < 16, so the two values are swapped.
Compare 15 to 11; 11 < 15, so the two values are swapped.
Compare 13 to 11; 11 < 13, so the two values are swapped.
Compare 11 to 9; 9 < 11, so there is no swap.
Number of comparisons: 5

2	**3**	**4**	**6**	**8**	**9**	**11**	**13**	**15**	**16**	21

This is our sorted array, which took **32** comparisons.

2	**3**	**4**	**6**	**8**	**9**	**11**	**13**	**15**	**16**	**21**

(b).

The initial array.

13	9	6	3	15	2	21	4	16	8	11

Compare 9 to 13; 9 < 13, so the lowest value is 9. Therefore, the two values are swapped.
Compare 6 to 9; 6 < 9, so the lowest value is 6. Therefore, the two values are swapped.
Compare 3 to 6; 3 < 6, so the lowest value is 3. Therefore, the two values are swapped.
Compare 15 to 3; 15 > 3, so the lowest value is still 3. Therefore, there is no swap.
Compare 2 to 3; 2 < 3, so the lowest value is 2. Therefore, the two values are swapped.
Compare 21 to 2; 21 > 2, so the lowest value is still 2. Therefore, there is no swap.
Compare 4 to 2; 4 > 2, so the lowest value is still 2. Therefore, there is no swap.
Compare 16 to 2; 16 > 2, so the lowest value is still 2. Therefore, there is no swap.
Compare 8 to 2; 8 > 2, so the lowest value is still 2. Therefore, there is no swap.
Compare 11 to 2; 11 > 2, so the lowest value is still 2. Therefore, there is no swap.
Number of comparisons: 10

2	9	6	3	15	13	21	4	16	8	11

Compare 6 to 9; 6 < 9, so the lowest value is 6. Therefore, the two values are swapped.
Compare 3 to 6; 3 < 6, so the lowest value is 3. Therefore, the two values are swapped.
Compare 15 to 3; 15 > 3, so the lowest value is still 3. Therefore, there is no swap.
Compare 13 to 3; 13 > 3, so the lowest value is still 3. Therefore, there is no swap.
Compare 21 to 3; 21 > 3, so the lowest value is still 3. Therefore, there is no swap.
Compare 4 to 3; 4 > 3, so the lowest value is still 3. Therefore, there is no swap.
Compare 16 to 3; 16 > 3, so the lowest value is still 3. Therefore, there is no swap.
Compare 8 to 3; 8 > 3, so the lowest value is still 3. Therefore, there is no swap.
Compare 11 to 3; 11 > 3, so the lowest value is still 3. Therefore, there is no swap.
Number of comparisons: 9

2	**3**	6	9	15	13	21	4	16	8	11

Compare 9 to 6; 9 > 6, so the lowest value is still 6. Therefore, there is no swap.
Compare 15 to 6; 15 > 6, so the lowest value is still 6. Therefore, there is no swap.
Compare 13 to 6; 13 > 6, so the lowest value is still 6. Therefore, there is no swap.
Compare 21 to 6; 21 > 6, so the lowest value is still 6. Therefore, there is no swap.
Compare 4 to 6; 4 < 6, so the lowest value is 4. Therefore, the two values are swapped.
Compare 16 to 4; 16 > 4, so the lowest value is still 4. Therefore, there is no swap.
Compare 8 to 4; 8 > 4, so the lowest value is still 4. Therefore, there is no swap.
Compare 11 to 4; 11 > 4, so the lowest value is still 4. Therefore, there is no swap.
Number of comparisons: 8

2	**3**	**4**	9	15	13	21	6	16	8	11

Compare 15 to 9; 15 > 9, so the lowest value is still 9. Therefore, there is no swap.
Compare 13 to 9; 13 > 9, so the lowest value is still 9. Therefore, there is no swap.

Compare 21 to 9; 21 > 9, so the lowest value is still 9. Therefore, there is no swap.
Compare 6 to 9; 6 < 9, so the lowest value is 6. Therefore, the two values are swapped.
Compare 16 to 6; 16 > 6, so the lowest value is still 6. Therefore, there is no swap.
Compare 8 to 6; 8 > 6, so the lowest value is still 6. Therefore, there is no swap.
Compare 11 to 6; 11 > 6, so the lowest value is still 6. Therefore, there is no swap.
Number of comparisons: 7

2	**3**	**4**	**6**	15	13	21	9	16	8	11

Compare 13 to 15; 13 < 15, so the lowest value is 13. Therefore, the two values are swapped.
Compare 21 to 13; 21 > 13, so the lowest value is still 13. Therefore, there is no swap.
Compare 9 to 13; 9 < 13, so the lowest value is 9. Therefore, the two values are swapped.
Compare 16 to 9; 16 > 9, so the lowest value is still 9. Therefore, there is no swap.
Compare 8 to 9; 8 < 9, so the lowest value is 8. Therefore, the two values are swapped.
Compare 11 to 8; 11 > 8, so the lowest value is still 8. Therefore, there is no swap.
Number of comparisons: 6

2	**3**	**4**	**6**	**8**	13	21	9	16	15	11

Compare 21 to 13; 21 > 13, so the lowest value is still 13. Therefore, there is no swap.
Compare 9 to 13; 9 < 13, so the lowest value is 9. Therefore, the two values are swapped.
Compare 16 to 9; 16 > 9, so the lowest value is still 9. Therefore, there is no swap.
Compare 15 to 9; 15 > 9, so the lowest value is still 9. Therefore, there is no swap.
Compare 11 to 9; 11 > 9, so the lowest value is still 9. Therefore, there is no swap.
Number of comparisons: 5

2	**3**	**4**	**6**	**8**	**9**	21	13	16	15	11

Compare 13 to 21; 13 < 21, so the lowest value is 13. Therefore, the two values are swapped.
Compare 16 to 13; 16 > 13, so the lowest value is still 13. Therefore, there is no swap.
Compare 15 to 13; 15 > 13, so the lowest value is still 13. Therefore, there is no swap.
Compare 11 to 13; 11 < 13, so the lowest value is 11. Therefore, the two values are swapped.
Number of comparisons: 4

2	**3**	**4**	**6**	**8**	**9**	**11**	13	16	15	21

Compare 16 to 13; 16 > 13, so the lowest value is still 13. Therefore, there is no swap.
Compare 15 to 13; 15 > 13, so the lowest value is still 13. Therefore, there is no swap.
Compare 21 to 13; 21 > 13, so the lowest value is still 13. Therefore, there is no swap.
Number of comparisons: 3

2	**3**	**4**	**6**	**8**	**9**	**11**	**13**	16	15	21

Compare 15 to 16; 15 < 16, so the lowest value is 15. Therefore, the two values are swapped.
Compare 21 to 15; 21 > 15, so the lowest value is still 15. Therefore, there is no swap.
Number of comparisons: 2

2	**3**	**4**	**6**	**8**	**9**	**11**	**13**	**15**	16	21

Compare 21 to 16; 21 > 16, so the lowest value is still 16. Therefore, there is no swap.
Number of comparisons: 1

2	**3**	**4**	**6**	**8**	**9**	**11**	**13**	**15**	**16**	21

This is our sorted array, which took **55** comparisons.

2	**3**	**4**	**6**	**8**	**9**	**11**	**13**	**15**	**16**	**21**

(c). Insertion sort would be better, because if we use selection sort, we would always have 55 comparisons for this list. However, if this list is almost sorted, using insertion sort would only have 12 comparisons, which would be way more efficient than selection sort.

5. **The three expressions are**:

$f(n) = 7n^2 + 3n \log(n) + 5n + 1000$
$g(n) = 7n^4 + 3^n + 1000000$
$h(n) = 7n*(n^2 + \log(n))$

so that the first one has complexity of $O(n^2)$, the second one has complexity of $O(3^n)$, and the third one has complexity of $O(n^3)$. Therefore, the **second** one has the largest complexity and grows the fastest, as it is an exponential function.

6. The **third** list, which is $1, \log(n), n, n*\log(n), n^2, n^3, 2^n, n!$,is correctly ordered from slowest growth to fastest growth. When n approaches larger and larger values, functions on the right side would always grow more rapidly and have much greater values than functions on the left side. To take the example of 50, it is calculated in the following:

- 1 = 1
- log(50) = 3.912023005428146
- n = 50 = 50
- 50 * log(50) = 195.6011502714073
- 50^2 = 2500
- 50^3 = 125000
- 2^{50} = 1125899906842624
- 50! = 30414093201713378043612608166064768844377641568960512000000000000

Therefore, the resulting complexity is order in a numerically ascending order.