Q1

[4, 3, 5, 6, 1]

13 - - 12 12 9
14 - - - 15 12
15 - - - 14 14
16 - - - - 13
17 - - - - 16
18 - - - 18 15
19 - - - - -
20 - - - - -
21 - - - - 19

[4, 3, 5, 6, 2]

13 - - 12 12 10
14 - - - 15 13
15 - - - 14 14
16 - - - - 14
17 - - - - 17
18 - - - 18 16
19 - - - - -
20 - - - - -
21 - - - - 20

[4, 3, 5, 6, 3]

13 - - 12 12 11
14 - - - 15 14
15 - - - 14 14
16 - - - - 15
17 - - - - 18
18 - - - 18 17
19 - - - - -
20 - - - - -
21 - - - - 21

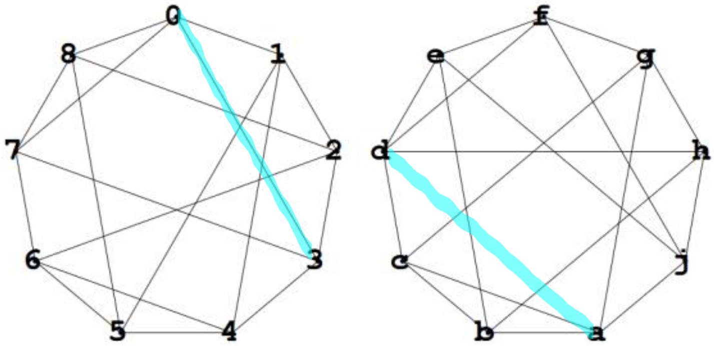
Q2

Superpolynomial if, for any positive integer k, limn→∞(n^k)/f(n) = 0

Or an algorithm is defined to take superpolynomial time if T(n) is not bounded above by any polynomial. Using little omega notation, it is ω(nc) time for all constants c, where n is the input parameter, typically the number of bits in the input.

3. Graphs $G = (V, E)$ (left) and $H = (W, F)$ (right) and bijection $f : V \leftrightarrow W$ are shown below.

j	0	1	2	3	4	5	6	7	8
f(j)	a	b	c	d	e	f	g	h	j

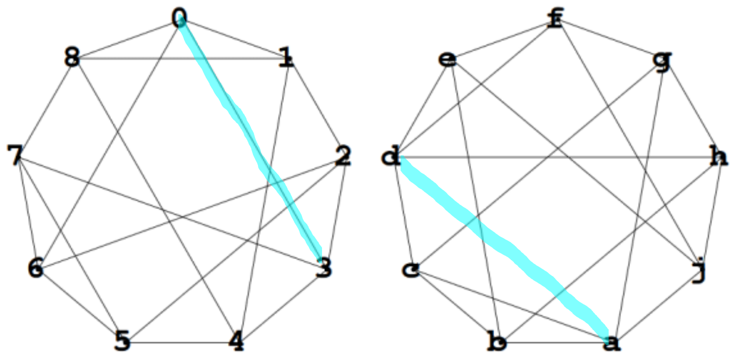


(a) Is f an isomorphism between G and H? (answer yes or no) No.

(b) Justify your answer.

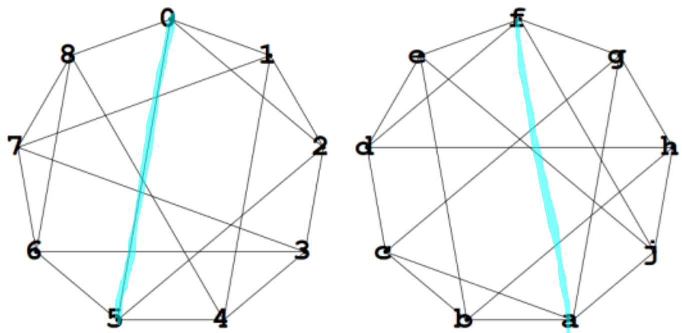
Take the edge $(0,3) \in E$. The result of applying f to it would be $(f(0), f(3)) = (a,d)$. But $(a,d) \notin F$. $\Rightarrow$ f is not an isomorphism.

j	0	1	2	3	4	5	6	7	8
f(j)	a	b	c	d	e	f	g	h	j



Take the edge $(0,3) \in E$. The result of applying f to it would be $(f(0), f(3)) = (a,d)$. But $(a,d) \notin F$. $\Rightarrow$ f is not an isomorphism.

j	0	1	2	3	4	5	6	7	8
f(j)	a	b	c	d	e	f	g	h	j



Take the edge $(0,5) \in E$. The result of applying f to it would be $(f(0), f(5)) = (a,f)$. But $(a,f) \notin F$. $\Rightarrow$ f is not an isomorphism.

4. On a 0-1 knapsack instance with n items, explain why the brute force algorithm performs $n2^{n-1} - 2^n$ additions.

The brute force algorithm for knapsack needs to compute the sum for all the subsets of items. For each subset of size t, there are $t-1$ additions needed.

The sum of all sizes of subsets of $\{1, \dots, n\}$ is $n2^{n-1} = f(n)$.

proof. By induction. This is true for $n=1$ [the subsets are $\{\}, \{1\}$, $f(1) = 0+1 = 1 \times 2^{1-1} = 1$ ✓]

Now assume this is true for $k=n$, that is, $f(n) = n2^{n-1}$.

The set $\{1, \dots, n, n+1\}$ has 2^n subsets that contain $n+1$, and 2^n that do not. Therefore:

$$f(n+1) = \underbrace{f(n)}_{\text{subsets that don't contain } n+1} + \underbrace{f(n) + 2^n}_{\text{subsets that contain } n+1,\ \text{each gets bigger by } 1}$$

$$= n2^{n-1} + n2^{n-1} + 2^n$$

$$= n(2^{n-1} + 2^{n-1}) + 2^n = n2^n + 2^n$$

$$= (n+1)2^n = (n+1)2^{(n+1)-1} \checkmark$$

Now that we know the total sum of sizes is $n2^{n-1}$, and the number of additions required for each one of size t is $t-1$, then the total number of additions is $n2^{n-1} - \underbrace{2^n}_{\text{one for each subset.}}$.

5. *Set cover* is this problem:

   Instance. A set U, a set $S = \{S_1, S_2, \ldots, S_t\}$ of subsets of U, and an integer k.

   Query. Is there a subset C of S of size k whose union equals U?

   a) In the set cover instance below, $U = \{0, 1, \ldots, 8\}$ and $S = \{$S0, S1, …, S10$\}$. Does this instance have a cover of size 6? (yes/no) Yes

   b) Justify your answer (use the space below at the right).

```
    0 1 2 3 4 5 6 7 8
S0  - - - - - - - * *
S1  - - * - - - - - -
S2  - * * - - - * - *
S3  * - - * - * - * -
S4  - - - * - - - - -
S5  * - - - - - - - -
S6  - - - - * * * - -
S7  * - - - - * - - *
S8  - - - - - - - - -
S9  - - - - * - - - *
S10 - * - - - * * - -
```

Sets S2,S3,S6 alone will cover U.

6. Prove/disprove: the set cover problem is in the class NP.

(i) Based on defintion above, set cover is a decision problem.
(ii) there is a polynomial time verifier, that for every element u in the universe U checks whether or not that element in included in one of - the sets S, this would take time |U| times size of the input which will be a polynomial time because size of the input is polynomial.