The Master Theorem For a recurrence like $T(n) = aT(\frac{n}{b}) + f(n)$, where $a \geq 1$ and $b > 1$, the Master Theorem states three cases:

1. If $f(n) \in O(n^c)$ where $c < \log_b a$ then $T(n) \in \Theta(n^{\log_b a})$.
2. If for some constant $k \geq 0$, $f(n) \in \Theta(n^c (\log n)^k)$ where $c = \log_b a$, then $T(n) \in \Theta(n^c (\log n)^{k+1})$.
3. If $f(n) \in \Omega(n^c)$ where $c > \log_b a$ **and** $af(\frac{n}{b}) \leq kf(n)$ for some constant $k < 1$ and sufficiently large n, then $T(n) \in \Theta(f(n))$.

1 Debug-and-Conquer

Your friend proposes a divide-and-conquer approach to compute a minimum spanning tree of a connected graph $G = (V, E)$. The helper function

```
{G_1, G_2} = subgraphs(G)
```

takes the graph $G = (V, E)$ and partitions the vertices into two graphs $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$ such that G_1 and G_2 are both connected and $|V_1|$ and $|V_2|$ differ by at most 1.

```
DC_MST(G = (V, E)):
    \\ Assume that G is a connected graph

    \\ Base cases:
    if |V| = 1:                          \\ 1 vertex, no edges
        return NONE
    if |E| = 1:                          \\ 1 edge, 2 vertices
        return E

    {G_1, G_2} = subgraphs(G)
    MST_1 = DC_MST(G_1)
    MST_2 = DC_MST(G_2)
    let e = the minimum-weight edge connecting G_1 and G_2
    return [MST_1, e, MST_2]
```

We'll start by analyzing the runtime of this algorithm. Assume that the `subgraphs` function runs in $O(n + m)$ time, where $n = |V|$ and $m = |E|$.

1. We'll find it helpful to consider the performance of this algorithm in the best and worst cases.

   (a) The best case for this algorithm is a particular (simple and common) type of connected graph. What type of connected graph yields the best-case runtime? VERY briefly justify your answer.

   (b) What type of connected graph yields the worst-case runtime? VERY briefly justify your answer.

2. (a) Give and briefly justify a good asymptotic lower bound on the best-case runtime of this algorithm in terms of n.

   (b) Give and briefly justify a good asymptotic upper bound on the worst-case runtime of this algorithm in terms of n.

3. Does this algorithm always generate a minimum spanning tree? Justify your answer.