Varibles used

p = number of points

k = no of clusters

For random function:

* random() takes O(k) to assign random points to clusters
* k\_means() takes:
* O(p\*k) in best case, in this case for each point we find distance from point and each centroid and the points are placed in correct cluster in first iteration only.
* O((p\*k)^2) in worst case, in this case for each point we find distance from point and each centroid and the points can change from one cluster to another (max is k).
* assignColor() takes O(p+k)
* printClusters() takes O(k)
* squaredDistError() takes O(p+k)

Hence, for random k means, the overall complexity is

* O((k)+(p\*k)+(p+k)+k+(p+k)) = O(p\*k) in best case
* O((k)+((p\*k)^2)+(p+k)+k+(p+k)) = O((p\*k)^2) in worst case

For pillar function:

* pillar() takes O(k\*p), for each cluster we calculate pillar by traversing all points
* k\_means() takes:
* O(p\*k) in best case, in this case for each point we find distance from point and each centroid and the points are placed in correct cluster in first iteration only
* O((p\*k)^2) in worst case, in this case for each point we find distance from point and each centroid and the points can change from one cluster to another (max is k).
* assignColor() takes O(p+k)
* printClusters() takes O(k)
* squaredDistError() takes O(p+k)

Hence, for pillar k means, the overall complexity is

* O((k\*p)+(p\*k)+(p+k)+k+(p+k)) = O(p\*k) in best case
* O((k\*p)+((p\*k)^2)+(p+k)+k+(p+k)) = O((p\*k)^2) in worst case

Conclusion: pillar() takes more execution time as its pillar() is O(p\*k) and that of random() is O(k), but the overall complexity is the same.