1. Use divide and conquer. If there exist a character with total occurrance $< k$, we can divide the string according to that character, otherwise the whole string is valid. The recursion depth is at most $|\Sigma|$, because at each time we recurse, we will delete at least one character. $O(n|\Sigma|)$.
2. For each $1 \leq m \leq |\Sigma|$, use two pointers to find maximal substrings with at least k repeating characters and exactly m unique characters. $O(n|\Sigma|)$.
3. Fix the right endpoint r of the substring, and let $p_c[r]$ denote the the first occurrence of character c when looking leftwards from r. We can wlog assume the optimal left endpoint is of the form $p_c[r] + 1$ by greedy. Use a balanced tree to maintain $p_c[r]$'s in increasing order, each character c will mark an interval in the balanced tree as invalid, and we want to find the last valid position in the balanced tree. $O(n \log |\Sigma|)$.

References