(2) First consider the case $k=1$, where there is only one negative edge. $f_1 = (u_1, v_1)$, remove such an edge (u_1, v_1), the remaining graph G' is just a graph with all positive edges which we can run Dijkstra's on, then run Dijkstra's on G' from S and from V_1, then we know $Dist_{V_1}$ and $Dist_S$, if $Dist_{V_1} u_1 + f_1 < 0$ and $Dist_S u_1 \neq \infty$, then we know there is a negative length cycle reachable from S. If there is no such negative length cycle, we will compare ~~to~~ the $Dist_S V_1$ and $Dist_S u_1 + f$ ~~and~~ take the smaller value, set the distance to V_1 to that smaller value and remove the (u_1, V_1) edge and do the Dijkstra's starting with both S and V_1 visited.

Complexity: we run Dijkstra's Algo. 3 times and comparsions are just $O(1)$ so it's still $O(n\lg n + m)$ and since $k=1$ it's also $O(kn\lg n + km)$

Correctness: the correctness of detecting negative length cycle reachable from S is from the fact if there is one, then it has to use the (u_1, v_1) edge then the shortest possible cycle ~~with~~ will be the shortest dist from V_1 to u_1 and the negative (u_1, v_1) edge, then if $Dist_S u_1 \neq \infty$, then it means the cycle is also reachable from S, thus by checking $Dist_{V_1} u + f < 0$ and $Dist_S u_1 \neq \infty$ will give out the correct result.

Then, if there does not have a negative cycle, then the distances are defined. The correctness of the distance checking is that first by checking $Dist_s V_1$ and $Dist_s u_1 + f$, we are checking all possible ways to get V_1, $Dist_s V_1$ represent the distance required from s to V_1 using every edge except (u_1, V_1) and $Dist_s u_1 + f$ is the distance ~~us~~ to V_1 using (u_1, V_1), we exhasted all possible ways and the smaller one will be the shortest distance for ~~us~~ s to V_1, then by adding V_1 into visited and remove u_1, V_1, the remaining Graph G' again is just positive edge graph and thus the Dijkstra's Algo runs will be correct.

Then we proved case $k=1$, we use it as base case.

Induction Hypothesis for # of negative edge $h \leq 1, 2, 3 \ldots k-1$, we have ~~and~~ ~~to~~ an algorithm Dijk-h to check it in $O(hn\lg n + hm)$ time.

Consider the case when $h=k$, then we remove one of the negative length edge $f_i = (u_i, V_i)$, then the remaing graph only has $k-1$ negative edge, then run Dijk-(k-1) from s and V_i, similarly like what we did in base case, ~~so~~ compare $Dist_{V_i} u_i + f_i < 0$ and $Dist_s u_i$, if it's true, or Dijk-(k-1) returns a negative length cycle ~~from~~ reachable from s, we have negative length cycle reachable from s, else we can then ~~com~~ compare $Dist_s u_i + f_i$ and $Dist_s V_i$, set V_i visited and $Dist_s V_i$ the smaller of the two, remove (u_i, V_i), run Dijk-(k-1) from s again.

The complexity: we run Dijk-(k-1) 3 times and comparisons are $O(1)$ times, then it's $O((k-1)n\lg n + (k-1)m)$ which is also $O(kn\lg n + km)$ since $k > k-1$. it could be ~~tele~~ found using Dijk-(k-1)

The correctness: After remove (u_i, v_i), it's either the remaining graph has a negative length cycle or not, if not the the only possible negative cycle is by using edge (u_i, v_i) then similar check the shortest path from v_i to u_i add the $f_i(u_i, v_i)$ will be the shortest cycle, if that's ~~greater than zero~~ negative, then we have a negative cycle. Then for the distance, by removing (u_i, v_i) and compute $dist_s$, the distance computed are very close to shortest, the only thing could make them shorter is by taking (u_i, v_i). More specific, by removing (u_i, v_i) we only directly modified the distance to v_i which iteratively affected the distance to all remaining nodes. Therefore by comparing $dist_s v_i$ and $dist_s u_i + f$, we similarly exhausted the possible ways to get to v_i and the smaller one will exactly be the shortest distance from s to v_i, we set v_i visited, it will also iteratively "shorten" the distance of the remaining nodes by calling Dijk-(k-1) ~~which will be the distance from s to~~ since Dijstra's Algo will compute distance iteration by iteration.