1. Recall that BFS computes shortest paths in $O(E)$ time on an unweighted graph, while Dijkstra takes $O(E+V \log V)$ for weighted graphs with nonnegative edge weights. In this problem, we consider how to speed this up for “small” edge weights, where $1 \leq w(u \rightarrow v) < C$ for some integer $C$.

(a) First, suppose all edge weights $w(u \rightarrow v)$ are in $\{1, 2\}$. Give an $O(E)$ algorithm to find the shortest path distances from a source $s$.

(b) Dijkstra’s algorithm normally visits vertices in order of increasing $c(u)$, and relaxes every edge out of the vertices it visits. Consider a variant of Dijkstra’s algorithm that instead visits vertices in order of increasing $\lfloor c(u) \rfloor$, with ties broken arbitrarily.

Suppose that $1 \leq w(u \rightarrow v)$ for all edges $u \rightarrow v$. Show that, on any shortest path $s = u_1 \rightarrow u_2 \rightarrow u_3 \rightarrow \cdots \rightarrow u_k$, this “rounded” variant of Dijkstra will visit $u_{k-1}$ before $u_k$.

Conclude that, by our lemma in class (namely: after the edges of a shortest $s \rightarrow t$ path have been relaxed in order, $c(t) = c'(t)$), this “rounded” variant of Dijkstra will correctly compute shortest path distances on graphs with $w(u \rightarrow v) \geq 1$.

(c) Now suppose that $1 \leq w(u \rightarrow v) < 2$ for all edges $u \rightarrow v$. Give an $O(E)$ algorithm to find the shortest path distances from $s$.

**Hint:** Run the “rounded” variant of Dijkstra, but rather than store vertices in a heap, keep a separate queue for each value of $\lfloor c(u) \rfloor$.

(d) Extend the above algorithm to $O(EC)$ time when $1 \leq w(u \rightarrow v) < C$. 
