1)
It is difficult to optimize this operation because *a priori* we know nothing about the graph. We can be efficient in that we will iterate over the whole adjaceny list once, however. For each vertex we will have an in counter, cin, and an out counter, cout. As we iterate over a row we increment the out counter for each head vertex we find. In addition, we increment the in counter of the head vertex each time we see it. There are a total of E edges in our graph, and each edge has an associated head and tail vertex. Therefore, we will perform a total of $2E$ arithmetic operations. The complexity of this algorithm is $O(|E|)$.
2)
Lemma: A graph can have at most one sink. If v and w are sinks, then $v.out = w.out = 0$ and $v.in = w.in = n - 1$. There is no edge from v to w, since v has no outgoing edges. Therefore, $w.in \leq n - 2$. This is a contradiction.

Furthermore, by the definition of a sink, a sink vertex v_i is characterized in the adjacency matrix by row $r_i = [0, 0, ..., 0]$ and column $c_i = [1, ..., 1, 0, 1, ..., 1]$ (where the zero occurs in the i^{th} position).

We can find this vertex in $O(|V|)$ by first finding a candidate vertex, and then checking to see if it is indeed a sink. Let's assume that $V = \{1, ..., n\}$. We start with cand = 1. Then we loop over the rows of the matrix and check if A[cand, current iteration] = 1. If it does, then row 'cand' cannot be a sink, and we increment cand by one. Then we check A[cand, current iteration]. We know that A[cand, current iteration - 1] = 1, so we have some hope that the column might be mostly ones, as described above. We continue iterating across the row as described.

This iteration ends in V steps. Then the process of checking if a vertex is a sink requires checking The row and column entries, which will require another $V + V - 1$ bits of information.
Algorithm
i = 1
for j = 1:n
  if $A_{i,j} == 1$;
  i++
  fi
rof
Now check to make sure i is a sink.
3)

$$G^* = (V, E^*)$$
$$V = \{1, 2, 3, ..., n\}$$
$$E^* = \{(i, k) : i \in (1, n-1), k \in (i+1, n), (i, k) \neq (n, n)\}$$

Then, we have edges (1, 2), (1, 3), (1, 4), ..., (1, n), (2, 3), (2, 4), ..., (2, n), ...(n-2, n-1), (n-2, n), (n-1, n), so the total number of edges is:

$$\sum_{i=1}^{n-1} i = \frac{n(n-1)}{2}$$

4) If we let a_i denote the i^{th} row of A, then $< a_i, u >$ gives the out-degree of vertex v_i. If we let a_i^T denote a row of A^T, then $< a_i^T, u >$ gives the in degree of the vertex v_i. Then, $(Au)^T u = (A^T u)^T u$, and both equal the total number of edges in the graph.