A. Assume for the sake of contradiction that A is context-free. Let p be a pumping length of A. Let $w = 0^p1^p0^p$. Then $w \in A$ and $|w| \geq p$, so w can be pumped. In the usual decomposition $w = uvxyz$, there are two cases.

- If vxy is a substring of 0^j1^k, then we pump w to uv^2xy^2z. Doing so either increases the 0s at the start of the string, so that they exceed the 0s at the end, or increases the 1s, so that they exceed the 0s at the end.
- If vxy is a substring of 1^k0^j, then we again pump w to uv^2xy^2z. Doing so either increases the 1s, so that they exceed the 0s at the start of the string, or increases the 0s at the end of the string, so that they exceed the 0s at the start.

In both cases, we have $uv^2xy^2z \notin A$, in contradiction of the pumping lemma. So A is not context-free.

B.A. [I'll not draw it, but imagine a graph G of two nodes, with a single edge from the first node to the second.] The adjacency matrix of G is

$$\begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix}.$$

The encoding is $\langle G \rangle = 01n00n$ (n is my newline character).

B.B. Based on my answer to B.A, the input alphabet is $\Sigma = \{0, 1, n\}$. I'll use two tapes. The first tape stores the input. The second tape keeps track of how nodes there are in the graph.

1. Scan the first tape once to wrap the input in turnstiles. While doing so, copy the characters before the first n (which should be the first row of the adjacency matrix) to the second tape, also leaving the second tape wrapped in turnstiles.
2. Scan the first tape once and the second tape repeatedly, comparing each row (substring between ns) on the first tape to the second tape, to make sure that all rows have the same number of symbols. If not, reject.
3. Scan the first tape once and the second tape once, to check that the number of ns also matches the number of symbols per row. If not, reject.
4. (At this point, we know that the input is a valid $\langle G \rangle$.) Scan the first tape once, checking that each row has exactly two 1s. If not, reject. If so, then accept.

C.A. No, the result does not immediately follow from Rice's theorem, because A is not a property of regular languages. To see so, let M be a Turing machine that hangs on every input. Let

N be a Turing machine that rejects every input. Then $L(M) = \emptyset = L(N)$, but $\langle M \rangle \notin A$ and $\langle N \rangle \in A$.

C.B. [The hint suggests that we should mimic our proof that $EMPTY_{TM}$ is undecidable.] Suppose for the sake of contradiction that E decides A. I now describe a decider D for $\overline{HALT_{TM}}$. On input $\langle M, w \rangle$, D does these steps:

1. Build a Turing machine M_w that, on input x, does this:
   (a) If $x = w$, then simulate M on x and output whatever M outputs.
   (b) If $x \neq w$, then enter into an infinite loop (hang).
2. Run E on $\langle M_w \rangle$ and output the opposite of what E outputs.

Both of these steps terminate, so D is a decider. Moreover, the following statements are logically equivalent.

1. D accepts $\langle M, w \rangle$.
2. E rejects $\langle M_w \rangle$.
3. M_w hangs on every input x.
4. M hangs on input w.
5. $\langle M, w \rangle \in \overline{HALT_{TM}}$.

Thus D decides $\overline{HALT_{TM}}$. But we already know that $\overline{HALT_{TM}}$ is undecidable. This contradiction implies that A is undecidable as well.

D. First, H is being run on inputs of the form $\langle N \rangle$, where N is a Turing machine. This observation tells us that A is of the form $A = \{\langle N \rangle : \ldots\}$. To nail down A more specifically, we probably need to think about the other missing piece. There are four cases for what M might do on input w:

1. If M rejects w, then we want R to accept $\langle M, w \rangle$. What we know about N, in this case, is that N accepts x.
2. If M hangs on w, then we want R to accept $\langle M, w \rangle$. What we know is that N accepts x.
3. If M accepts w in more than $|x|$ steps, then we want R to reject or hang on $\langle M, w \rangle$. What we know is that N accepts x.
4. If M accepts w in $|x|$ or fewer steps, then we want R to reject or hang on $\langle M, w \rangle$. What we know is that N hangs on x.

In the first two cases, N accepts all inputs x. In the last two cases, N accepts some inputs x (the ones that are short enough) and hangs on other inputs x (the ones that are too long). So the recognizer H should accept $\langle N \rangle$ such that N always accepts, and H should reject or hang on $\langle N \rangle$ such that N sometimes accepts and sometimes hangs. Here are two possibilities for what A could be:

- $A = \{\langle N \rangle : N \text{ accepts all inputs}\}$.
- $A = \{\langle N \rangle : N \text{ halts on all inputs}\}$.

[The former language is ALL_{TM}, which we already know to be unrecognizable from other arguments. The latter language consists of all deciders. I do not know of any other proof that it is unrecognizable.]