Un primer algoritmo para resolver el problema en matrices de adyacencia sería, recorrer la matriz por filas, por cada vértice vemos sus adyacentes. Si nos encontramos con un "1" (una adyacencia), recorremos el resto del vector fijando esa posición y creando una arista por cada otro "1" que nos encontremos. En el caso peor, con un grafo completo, por cada vértice de la matriz de adyacencia, hacemos un trabajo de $O(n^2)$. ~~Como~~ ~~ya que~~ tenemos n vértices, el coste total para a ser $O(n^3)$.

Una estrategia más inteligente, es elevar al cuadrado la matriz de adyacencia usando el algoritmo d'Strassen o, al tratarse de matrices de "1's" i "0's", resolverlo con AND's i OR's. Este método es correcto, ya que al multiplicar, realmente estamos comprovando la adyacencia del vértice fila con el vértice columna de la otra matriz a través del vertice elemento que estamos multiplicando:

$$V_F \begin{pmatrix} \overset{V_E}{\downarrow} & & \\ a_1 & \boxed{a_4} & a_7 \\ a_2 & a_5 & a_8 \\ a_3 & a_6 & a_9 \end{pmatrix} \cdot \begin{pmatrix} \overset{V_C}{\downarrow} & & \\ \boxed{a_1} & a_4 & a_7 \\ \boxed{a_2} & a_5 & a_8 \\ \boxed{a_3} & a_6 & a_9 \end{pmatrix} \leftarrow V_E$$

Por tanto, estamos comprovando la codición que nos piden para poner una arista en el grafo resultante. El coste será de $O(n^{\lg 7})$ al tratarse de Strassen.

Finalmente, en el caso de lista de adyacencia, nos damos cuenta de que el grafo cuadrado está formado por aristas con los vecinos de los vecinos de cada vértice que vamos visitando. De esta manera, lo que hará nuestro algoritmo será, para cada vértice de la lista de adyacencia ira uniendo las listas de adyacencia de sus vecinos en el grafo original (evitando repetidos y a sí mismo) para generar su lista de adyacencias en el grafo cuadrado. La comprovació de repetidos se podria hacer mediante un vector de booleanos.

El coste de este algoritmo sería:

$$\circledast \sum_{u \in V} \sum_{v \in N(u)} d_v \leq \sum_{u \in V} \sum_{v \in N(u)} n = \sum_{u \in V} n d_u = n 2m = O(nm)$$

$\circledast$ [El sumatorio_para_cada_vertice (sumatorio_para_cada_arista del vertice (grado del vértice))]
En el peor de los casos todos los vértices son adyacentes a todos.

Este algoritmo es mejor que el planteado con Strassen cuando la matriz es dispersa.