633260 2
When is an adjacency matrix a good choice for implementing a graph?

a. When the graph is undirected.

*b. When the graph is nearly complete.

c. When a graph has few edges.

d. When the graph contains no cycles.

e. When the graph is connected.

f. "
g. "
h. "
i. "
j. "

General Feedback:

The adjacency matrix will compactly and efficiently store edges when it contains little wasted space. In a complete graph, each vertex shares an edge with each other vertex, meaning all elements in the adjacency matrix will be used.

634147 2
What is true about the pivot in quicksort?

a. Before partitioning, it is always the smallest element in the list

b. After partitioning, the pivot will always be in the centre of the list

*c. After partitioning, the pivot will never move again

d. A random choice of pivot is always the optimal choice, regardless of input

e.
f. "
g. "
h. "
i. "

General Feedback:

As the pivot will be to the right of all smaller elements and to the left of all larger elements, it is in the same position it will be when the array is sorted.

633259 2
When is an adjacency list a good choice for implementing a graph?

a. When the graph is undirected.

b. When the graph is nearly complete.

*c. When a graph has few edges.

d. When the graph contains no cycles.

e. When the graph is connected.

f. "
g. "
h. "
i. "
j. "

General Feedback:

With adjacency lists are used, each vertex stores its own list of vertices it's connected to. When a graph contains few edges, these lists will be very short and will likely consume less memory than the adjacency matrix.