633259
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.

633257
You've got an algorithm that is $O(N^2)$. On the first run, you feed it a collection of size M. On the second run, you feed it a collection of size M / 2. Assuming each run has worst-case performance, how much time does the second run take?

*a. 1/4 of the first

b. 1/2 of the first

c. 2/3 of the first

d. 3/4 of the first

e. 1/8 of the first

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

General Feedback:

The first run took time proportional to M^2. The second run took $(M/2)^2$, or $M^2/4$. The second run is 1/4 of the first.

633245
Two algorithms accomplish the same task on a collection of N items. Algorithm A performs N^2 operations. Algorithm B performs 10N operations. Under what conditions does algorithm A offer better performance?

*a. N < 10

b. N < 100

c. N > 10

d. For all N.

e. For no N.

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

General Feedback:

The two algorithms offer equal performance when N = 10. For N greater than 10, $N^2 > 10N$.