Hello, I am Noel. In my part, I will talk about time complexity of adjacency list and compare it with matrix.

I divide the code into two parts. the first half. Here we use a for loop to initialize HeapNode List for all vertices. So it’s O(V). And then we add all the vertices into our minheap structure, every insert is O(logV) and we loop it for V times to insert all the vertices. So the first half totally is O(V+VlogV). Then we see the second half. The while loop will takes V times. And in every loop, we need to extractMin once which time complexity is O(logV) and we have another for loop in it, which will loop for e times, e is the number of edges that connected to one vertices, so we use small e. and the decreaseKey takes O(logV). So totally, we (logV+e\*logV) sum up V times. We can get roughly (ElogV + VlogV). This big E is all the edges in the whole graph.

Therefore, the overall Adjacency List time complexity is roughly O(V + ElogV + 2VlogV), which is O((E+V)logV).

Now we compare these two times complexity. It seems that list must run faster than matrix, since one is V square another is E+V times logV. But when we compare these two, actually we need to compare the exactly complexity equation here. And since the graph must be a connected graph in this problem. E will be in range of V-1 to 1/2V(V-1). So we replace the E with it min and max value and draw the graph. Red line is list with min edges, green line is list with max edges, and black is matrix. We can find green line always above black line and black line always above red line. If cannot see the green and black clearly, I also use green minus black. We got the function always above 0. So we can get when the num of edges is small, list run faster than matrix. When the num of edges become large, matrix run faster than list. These means there will be an edges number for every vertices number that matrix get better than list.

So now we try to find the critical edge number. Use this equation and transform it to use V represent E. We can draw a graph here. And find in a specific vertices number, what will be the critical edge number is. The graph with more edges than E, using matrix better. The graph with less edges than E, using list better.