Muhammad Arham Khan
21701848
CS473

Part 1 :

a) I created structures to store the vertices and the edges of the graph and linked them using pointers. and then used global dynamic arrays to store them so they could be accessed across functions. For the heap implementation part, I created an array of vertex pointers that referenced actual pointers and this structure contained the D value of that vertex as well. This way I way able to have an independent Heap implementation alongside the code. Following the algorithm, The code updated the D values of all active elements by parsing their edge array and editing the DValue relevantly.
Since the program has a heap implementation where each Max traversal takes O(logn) time and there are n traversals so in the worst case, considering that the program runs in linear time, the worst case run-time of the algorithm will be O(n^2logn);

b) The implementation was the same as in part (a) but instead of maintaining the heap, I simply traversed through the DValues array to find the Max on each loop in each pass. This was we were able to retrieve the DMax for A and B in linear runtime and much faster.
Since the algorithm has a linear traversal to calculate the Max values of D, the worst case runtime of the algorithm will be O(n^2) which is more efficient than the previous one.

Part 2:

For Erdos:

	Heap	Linear	NetworkX
Initial Cut	4582	4582	4582
Final Cut	1109	1030	929
Time	9.1s	3.5s	1.9s

-NetworkX runs the fastest

*Code was unable to finish for com-DBLP and rgg which are larger matrices

Part 3:

At first, I was aiming to calculate the value of D changes on each pass all over again which was a time-consuming process, then I improved it the account for only the node neighbours which were affected in the pass.

Similarly, I was able to reduce the heapsort update time by reducing the heap sizes on each pass and instead making it an iterative loops of O(n) runtime to update the dValues in the heap.