Team members: Chen Wang(cwang784), Haodan Tan(htan74)

Division of labor:

1) Chen Wang:

- Read the file and store the wordlist.
  void readFile()
- Write the function to judge if there is an edge between two words.
  int judgeEdgeBetweenWords()
- Construct the structure of Node and Graph and execute the constructing functions.
  struct Node; struct Graph; struct Node* createNode(); struct Graph* createAGraph(); void addEdge();
  void printGraph();

2) Haodan Tan:

- Construct the MinHeap and execute the contructing functions.
  struct MinHeapNode; struct MinHeap; void swapMinHeapNode; void minHeapify; int isEmpty();
  struct MinHeapNode* extractMin(); void shortDis(); int isInMinHeap();

- Conduct the Dijkstra algorithm.
  void dijkstra();

- Print the Path.
  void printArr(); void printPath();

Brief Explanation: Graph and Connection

1) Construct the structure Node to store the index of the word and its next connected node. Also, construct the structure Graph to store the total number of the nodes, and the double Node pointer to store the address of adjacent nodes.
2) Create a new node and a new graph by dynamic allocation.
3) Write the conditional function to judge if there is an edge between two node. The first condition is that there is only one of four characters is different. The second is that two continuous characters can be swaged to be the same.
4) Traverse all 500 nodes to add edge between two nodes if the conditions are satisfied.
5) Print the graph in order to see the result to be sure it is correct

Future Improvement:

We could package the edge addition part into a function, so the main function would seem simpler.

Brief Explanation: Shortest Path (Dijkstra Method)

Main Idea: The Idea is to traverse all node of graph and use a Min Heap to store the nodes which shortest path is not finalized yet. Min Heap is used as a priority queue to get the minimum distance node from set of not yet included nodes.

1) Construct the MinHeapNode structure to represent a min heap node and construct the Minheap to represent a min heap.
2) Conduct the function to create a new min Heap Node and a min heap by dynamic allocation.
3) Write a standard function to heapify at the given index. Also, write the function to extract minimum node from heap. Also, write the function to decreasing distance value of a given vertex v. Write the function to check if a given vertex is in minheap or not.

4) Execute the Dijkstra function by combining all structure and functions to calculate the distances of shortest paths from source node to all vertices. We could use conditional constrains to control the result that we want.

Future Improvement:

Define better data structure to reduce the extra functions.