HW2

1. Cuckoo Hashing
   Each cell of the table keeps track of an extra boolean variable, called *modified*. *modified* represents whether the cell was modified during the insertion. When inserting a value to a position, we first check the *modified* value of the cell. If it's False, we insert it and keep going. If it's True, we encountered a loop in this insertion. No matter if we insert the value with or without a loop, we need to reset the *modified* value back to False after.
2. Linear Probing
   After inserting 2 elements in the N-size table,
   $total\ number\ of\ possible\ patterns = C(N,2) = \frac{N!}{2!(N-2)!}$
   If we keep cell 0 and 1 empty, that means the rest N-2 cells have the 2 inserted values
   $number\ of\ patterns\ when\ cell\ 1\ and\ 0\ are\ empty = C(N-2,2) = \frac{(N-2)!}{2!(N-4)!}$
   $Pr[cell\ 0\ and\ 1\ are\ empty] = \frac{number\ of\ patterns\ when\ cell\ 1\ and\ 0\ are\ empty}{total\ number\ of\ possible\ patterns} = \frac{(N-2)\cdot(N-3)}{N\cdot(N-1)}$
3. Binary Heap
   (a) The maximum is located at the leaf nodes of the binary heap
   (b) There are at most $\frac{n}{2}$ locations that could contain maximum.
4. Priority Queue
   (a) The worst-time bounds for add, decreasePriority, and delete are all O(n) when we need to search the whole array to find the right location to insert/delete the new value.
   (b) Yes. When the newly added edge weight of Dijkstra's is the minimum among all known edges. Priority queue will have better performance. The following graph is an example if we want to find shortest path between A and E