Problem 1

Algorithm:

1. Find a min-cut A that has max-flow of lets say a (According to <https://www.geeksforgeeks.org/minimum-cut-in-a-directed-graph/>, you can find the min-cut by the following:
2. Run the Ford-Fulkerson algorithm and consider the final residual graph
3. Find set of vertices that are reachable from the source in the residual graph
4. All edges which are from reachable vertex to non-reachable vertex are minimum cut edges)
5. Since we can remove at most k edges, remove up to k edges that are leaving/entering A (the min-cut we defined in part 1)
6. The capacity of the min-cut A is now a-k or 0(if k was greater than the existing edges), after the removal of k edges, meaning that the max-flow is also a-k or 0.
7. Return the set of k edges that decreased the max-flow into a-k or 0.

Proof:

The question requires us to find the set of k edges to remove from the directed graph so that we can decrease the maximum flow the most. All capacity has the weight of 1 means that every edge in the directed graph has the weight of 1, meaning that if we can remove k edges, we can decrease the maximum flow by at most k. After finding the minimum cut A that has max-flow of a, if we remove up to k edges, the min-cut now has a-k or 0 (if the value of k is greater than a – ex) maximum flow is 10 but k has the value of 15), which also gives the maximum flow of a-k or 0. Since we reduced the maximum flow by k or removed all the edges of the min-cut A, those edges can be returned.

Runtime:

In order to find the min-cut A, we have to run the Ford-Fulkerson Algorithm, which is O(m\*f), according to <https://stackoverflow.com/questions/33565995/time-complexity-of-the-ford-fulkerson-method-in-a-flow-network-with-unit-capacit>, where m is the number of edges in the directed graph and f is the number of maximal flow. Removing the edges and putting them into a set to return the edges simply takes constant time, c, giving O(m\*f + c), which is O(m\*f). The graph has E edges, so O(E\*f) is the runtime.

Problem 2

(a)

According to the lecture slides, if we choose an edge randomly, the probability that that edge is in the minimum cut is 2/n, where n is the number of nodes.

If we follow the algorithm given in class to reduce a node each step by merging two nodes, the number of nodes at every reduction step decreases by 1, until there is only two nodes left. Thus, if we calculate the probability that the final cut is minimum cut, we get

P (final cut between two nodes is the minimum cut)

= (1- 2/n) \* (1-2/(n-1)) \* (1-2/((n-2)) \* … \* (1- 2/4) \* (1 – 2/3)

= ((n-2)/n) \* ((n-3)/(n-1)) \* ((n-4)/(n-2)) \* … \* 2/4 \* 1/3

= 2/(n\*(n-1))

We get that the probability equals 2/(n\*(n-1)).

Since we are interested in the maximum number of min-cuts, we have to find the expected number, which is 1/p.

We found that p = 2/(n\*(n-1)) or 2/(n2- n), so expected number is (n\*(n-1))/2.

Therefore, the maximum number of min-cuts is (n\*(n-1))/2 for n vertices.

(b)

The undirected graph above has 5 vertices(nodes), with 5\*4/2 = 10 maximum number of min-cuts. For simplicity, we assume that all edges have the length of 1 By following the algorithm given in class, we reduce a node by merging two nodes:

If we merge nodes e and d, we get the graph above.

If we merge nodes a and b, we get the graph above.

If we merge nodes e,d with node c, we get the graph above and from this diagram, we get the min-cut to be 2.

Now, after finding that min-cut is 2, we have to find 10 sets of 2 edges that make a cut through the diagram, which are:

1. edge(a,b) and edge(a,c)
2. edge(a,c) and edge(c,d)
3. edge(c,d) and edge(d,e)
4. edge(d,e) and edge(e,b)
5. edge(e,b) and edge(a,b)
6. edge(a,c) and edge(d,e)
7. edge(a,c) and edge(e,b)
8. edge(c,d) and edge(e,b)
9. edge(c,d) and edge(a,b)
10. edge(d,e) and edge(a,b)

If I disconnect those combinations of two edges above, the undirected graph is now disconnected and therefore the undirected graph has total of 10 minimum cuts.

Problem 3

(a)

If there are 100 quotations, there is a probability of 1/100 quotation for each candy. In other words, since one candy can only contain one quotation and there are 100 quotations, each quotation has the probability of 1/100 of coming out in one candy. If we collect all the quotations with the probability higher than 2/3, we have to dig into each of probability that a certain quotation, lets say x, is chosen from n candies. In other words, we have to check how many candies to buy to get the quotation x with the probability higher than 2/3 or 0.6666667:

For each quotation x:

N \* P(x) > 2/3, where N is the number of candies we need to buy and P(x) is the probability of quotation x being chosen.

Since each quotation has the probability of 1/100, P(x) = 1/100 and we get the value N > 66.67. This means that each distinct quotation requires more than 66.67 candies, which is at least 67 candies. Since there are 100 distinct quotations and each quotation requires 67 candies in order to meet the condition that every quotation is chosen at the probability higher than 2/3, we need to multiply 100quotations with 67candies per quotations, giving 6700 candies.

Therefore, if there are 100 quotations, we need at least 6700 candies in order that the probability > 2/3 we get every quotation.

If there are not 100 quotations but n different quotations, we can simply plot the value 1/n to P(x) for every quotation, giving N > 2/3\*n, where N is the number of candies we need for each quotation x and n is the total number of quotations that we need to collect. Since we need to find N for every quotation, we need to multiply n again, giving 2/3\*n2 as the total number of candies required to ensure that we collect all n quotations with the probability higher than 2/3 is 2/3\*n^2. The rate of growth is dependent on the n quotations (the number of quotations).

(b)

The expected number of candies is 5.187377 \* 100 = 518.7377 candies.

When you buy one candy, you are expected to get a new quotation since you did not open any candy prior to that candy.

When you open the second candy, the P(new quotation) = (100-1)/100 = 99/100 since you already have one quotation that you do not want in the second candy. Expected number is calculated by 1/p, so E(x) = 100/99.

Similarly, the third quotation’s expected value is E(x) = 100/98.

We need to add all the expected values to get the final expected value of collecting 100 quotations.

Following this pattern, we get 100/100 + 100/99 + 100/98 + 100/97 + … + 100/3 + 100/2 + 100/1 = 100 \* (1/100 + 1/99 + 1/98 + … + 1/2 + 1/1) = 518.7377.

Therefore, the expected number of candies to open to collect all 100 quotations is 518.7377 candies.