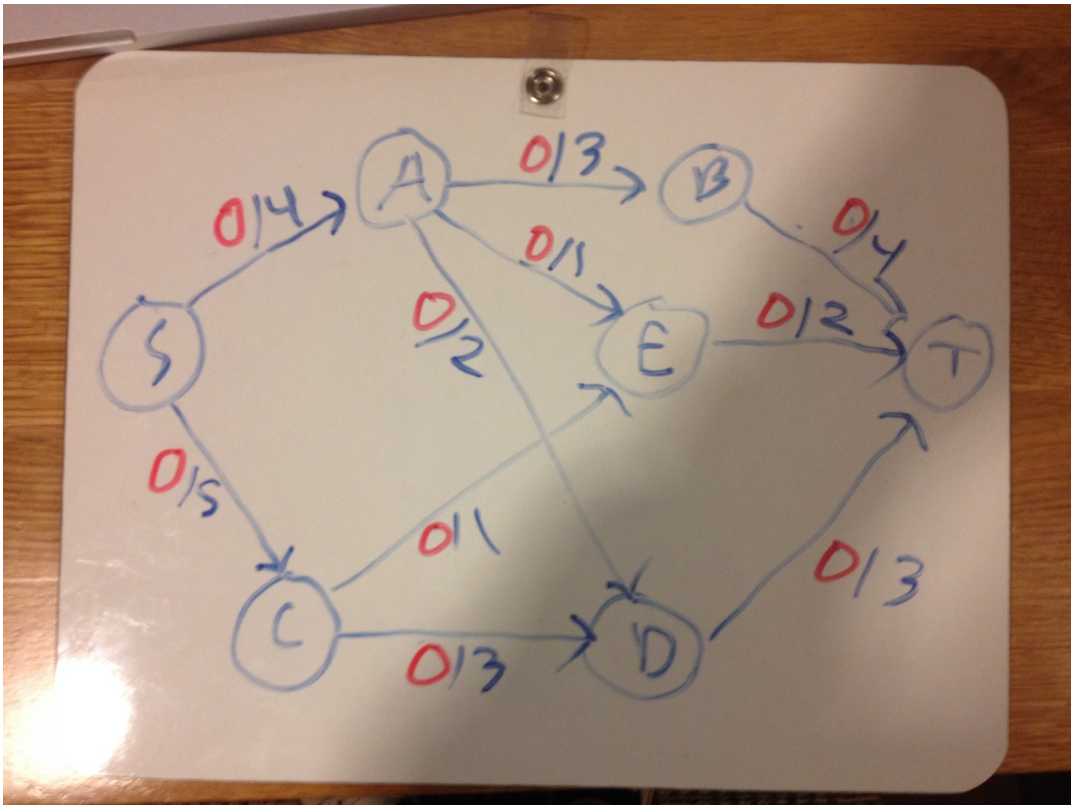
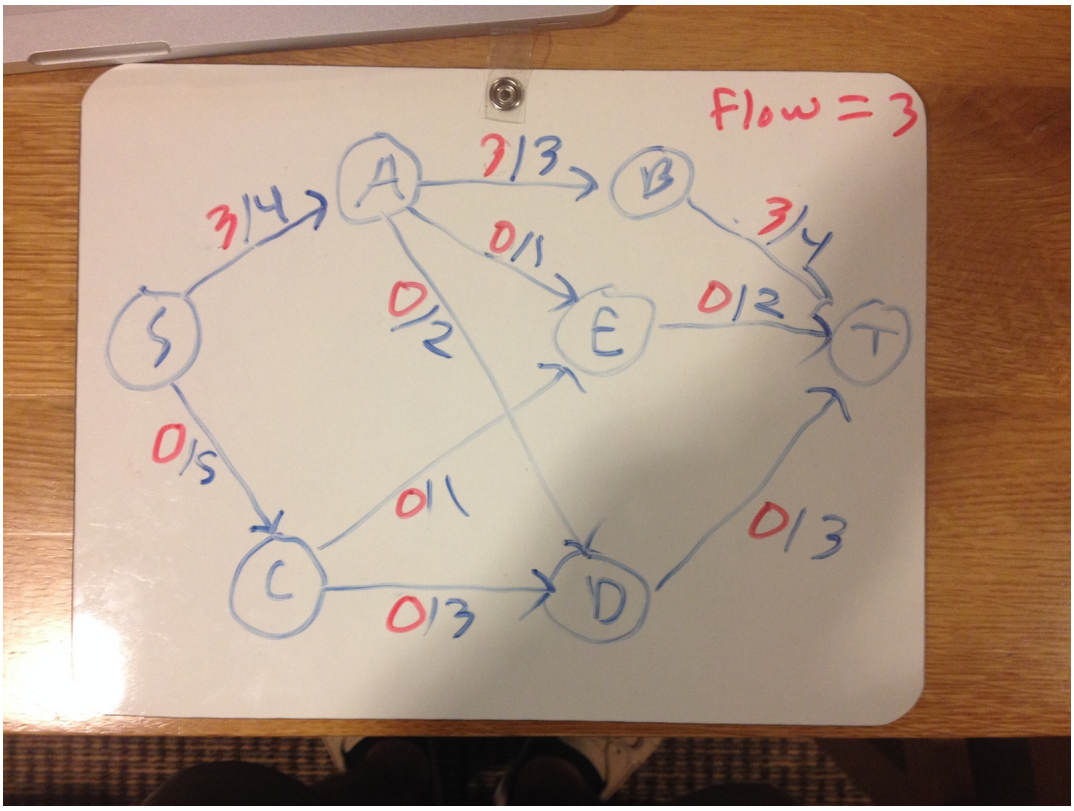
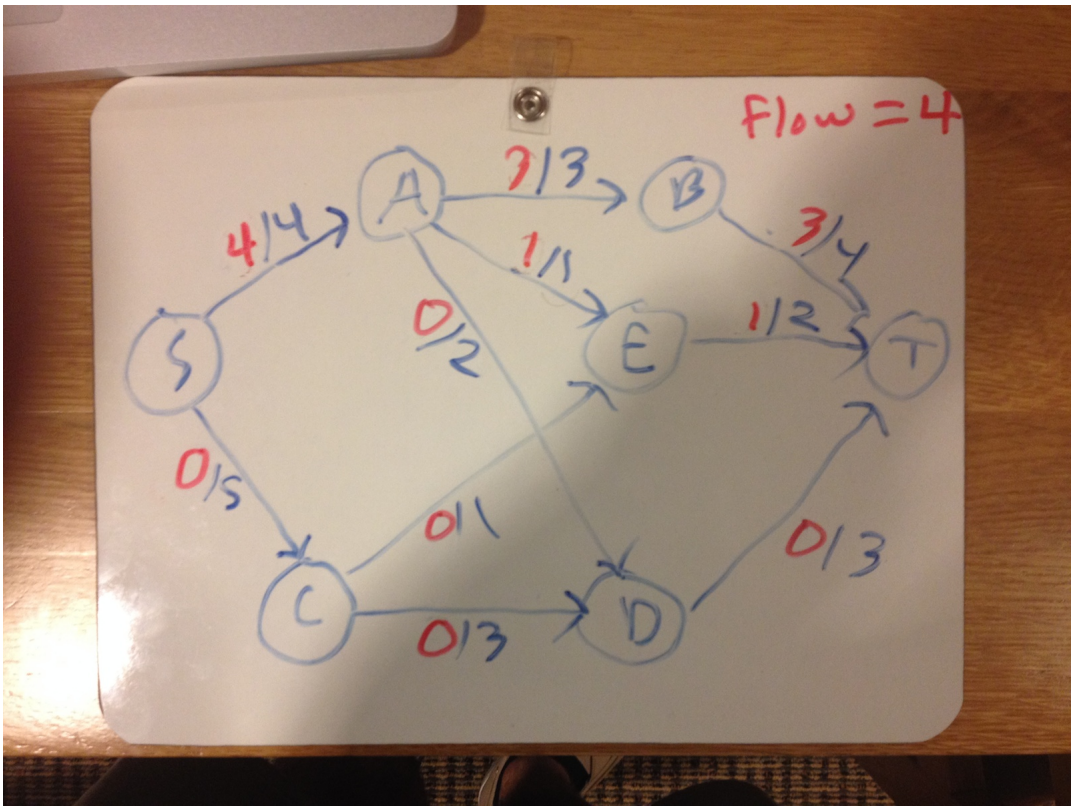
Buffalo Hird
CS124
Assignment 7

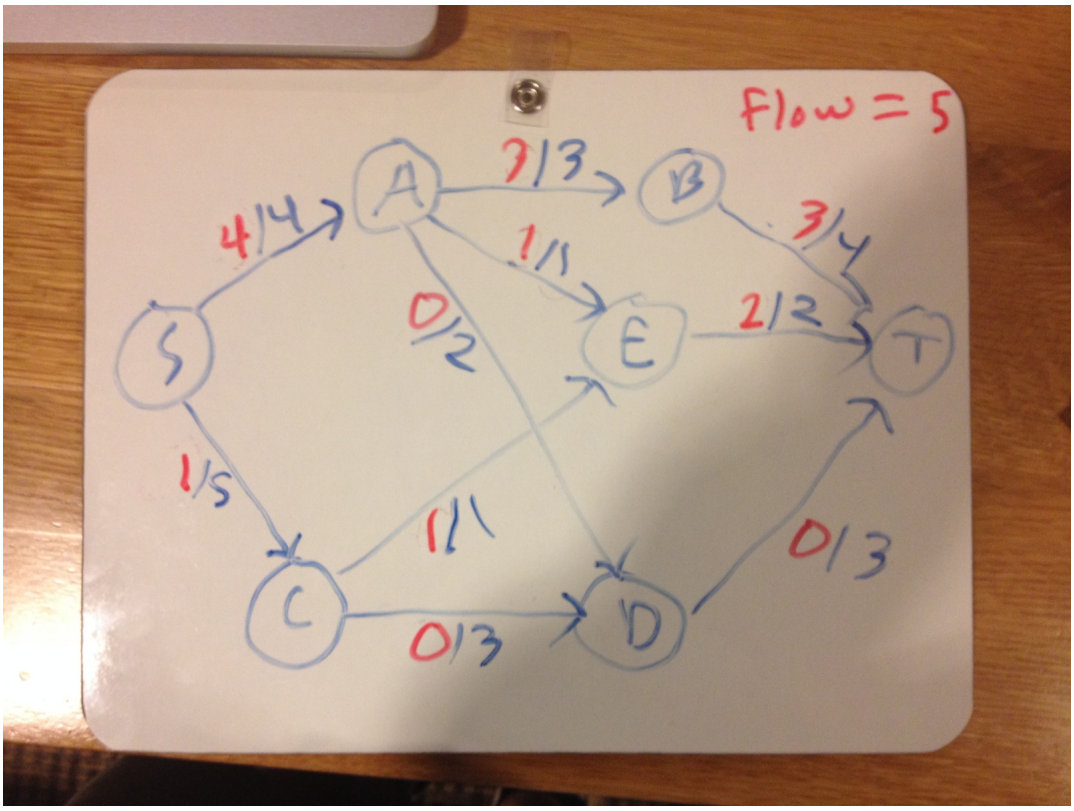
Problem 1:

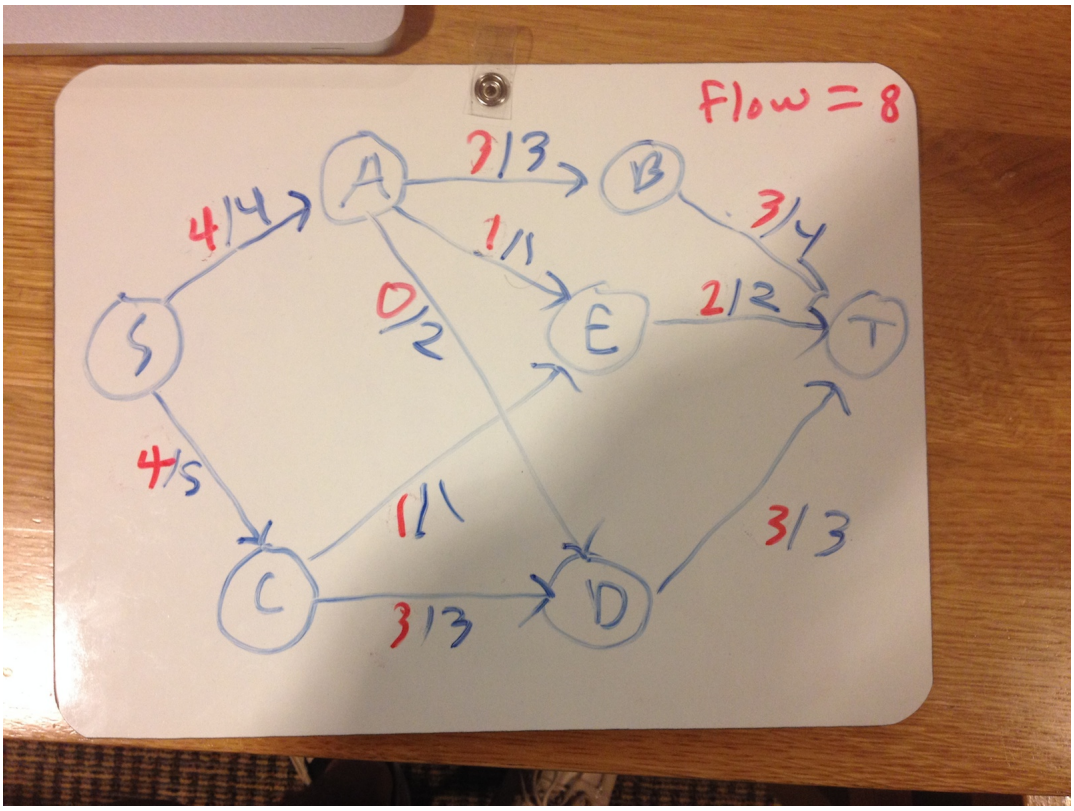
We can use Ford-Fulkerson's algorithm to solve this graph for maximum flow fairly easily. We find a final maximum flow from s to t of 8, which is unsurprising given that the maximum flow that can leave s in theory is 9 and the maximum flow that can enter t in theory is 8. Therefore, given an ideal set of intermediary nodes between s and t we expect a maximum flow of 8. We in this case do find this maximum flow value using the Ford-Fulkerson's algorithm. We show the residual network at the intermediate steps as we build the flow below. Note that for easier whiteboard writing I used the shorthand edge weight for edge $(u, v) = \frac{x}{y}$ such that if this edge from u to v has value $\frac{x}{y}$ then there is an edge (u, v) with value $y - x$ and a backedge (v, u) with value x:




Flow = 3
A
B
S
E
T
C
D
3/4
3/3
3/4
0/1
0/2
0/2
0/5
0/1
0/3
0/3


Flow = 4
A
B
S
E
T
C
D
3/3
4/4
3/4
1/1
0/2
1/2
0/5
0/1
0/3
0/3


Flow = 5
A
B
S
E
T
C
D
4/4
3/3
1/1
3/4
0/2
2/2
1/5
1/1
0/3
0/3



At this point we determine that there are no further augmented paths as we find no backedges by which we can redivert flow to increase the total flow through the system. As a result, we return max-flow = 8 as the solution of our problem. To determine the minimum cut we take the edges which limit our flow at each iteration. We therefore have min-cut of value 8:

$$\{(A, B), (A, E), (C, D), (D, T)\}$$

Problem 2:

2a) There are multiple sources and multiple sinks, and we wish to maximize the flow between all sources and all sinks.

We redesign the problem such that we create a new single master sink and single master source. For the master source, we combine each source into one single source. For the sink, we combine all sinks into one master sink. We note that any edge which leaves (or for a sink entered) a source will be present in the new master source/sink. We note that this is effectively equivalent to running Ford-Fulkerson iteratively on each source for each timestamp. For example, at T=0, run the algorithm on source_0 then source_1, etc. Repeat this for each iteration $T = t$ until we have determined the max flow for this graph. We can

see that this is correct as as we run continued iterations of Ford-Fulkerson we will eventually redistribute flows to and from these sources and sinks until we have found a max flow.

2b) Both the edges and the vertices (except for s and t) have capacities. The flow into and out of a vertex cannot exceed the capacity of the vertex.

We can easily solve this by considering a capacity to be an internal maximum flow. Therefore if we have a vertex with a capacity, we can redraw that vertex as two vertices with a dummy node such that there is node_1 with all the incoming edges of the vertex and node_2 with all the outgoing edges of the vertex. node_1 then has an outgoing edge to node_2 containing the capacity of the vertex. In this way, we have captured this constraint using only edge capacities. We know this must be correct because these internal capacities of the vertices only affect what amount of flow can leave them as they cannot have affected the flow potential before it reaches the vertex. Therefore, as we capture the same flow constraint with this new edge and dummy node, we have the same constraints represented in our graph. This modifed graph can then be solved using max-flow algorithms as studied in class.

Problem 3:

We note that this problem is essentially a variation on the coupon collector problem as seen in section notes 8. We then too note that we showed in class that Karger's algorithm will return a given minimum cut with probability $\frac{1}{n^2}$. We then modify this algorithm to find all minimum cuts.

We note that there are at most $\binom{n}{2}$ min cuts by definition in the worst case where every cut is a min cut (one big cycle). We therefore note that defining I_j to be an indicator variable which is defined as

$$I_j = \begin{cases} 1 & Karger's = cut_j \\ 0 & Karger's = cut_i, i \neq j \end{cases}$$

where cut_j refers to Karger's algorithm returning the jth cut. We can therefore determine the expected values of all of these potential cuts such that we determine the expected number of executions to find a min cut:

$$E(I_0 + ... + I_{\binom{n}{2}})$$

By sum of expectation we have:

$$= E(I_0) + ... + E(I_{\binom{n}{2}})$$

by symetry we then have:

$$= \binom{n}{2} * E(I_j) = \frac{\binom{n}{2}}{\binom{n}{2}} = 1$$

As we expect to get some min cut on ever execution of the algorithm, we can then apply this as a variation of the coupon collector problem where we sample from a set of outcomes until we have had all desired outcomes occur. Viewing each min cut as a coupon we note we want to collect all of up to $\binom{n}{2}$ coupons.

We note that the correctness of the coupon collector problem is well-defined and we showed the runtime in class. From class we note that the expected runtime of the coupon collector problem is the expected value of a sum of geometrics which is $\theta(nlogn)$. However, in this case $n = \binom{n}{2}$ such that we have expected runtime $\theta(\binom{n}{2}log\binom{n}{2})$.

We can use Markov's inequality to bound this algorithm to a Monte Carlo approximation with failure probability of at most $\frac{1}{3}$. Using Markov's inequality we get $\theta(log3 * \binom{n}{2}log\binom{n}{2})$. We note that asymptotically this has not affected our answer and since $n^2 \geq \binom{n}{2}$ we can then run this with a probability $\frac{1}{3}$ of failure in time $O(n^2log(n)$ as desired.

If we desired this for any given $0 \leq P \leq 3$, we could rewrite this to be in $O(n^2log(n)log(\frac{1}{P})$ which of course is asymptotically equivalent

Problem 4:

We consider two cases: 1) where the capacity of an edge is increased by 1 2) where the capacity of an edges is decreased by 1. We consider the first case first:

We note that Ford-Fulkerson always terminates and returns us a residual graph with a minimum cut such that the source and sink are on opposite sides of said cut. We therefore can consider two subcases: i) the edge is on the cut ii) the edge is not on the cut

ii) f it is not in the minimum cut then we are not concerened with this edge as it is not in the bottleneck of the flow problem such that adding additional capacity to this edge will not affect our solution as we will still be bottlenecked by our minimum cut edges.

i) If the edge is in the cut we increment the capacity of this edge and run our breadth-first-search again from the source to the sink such that a new path exists. If this is the case, then we have incremented the max flow of our system such that it is greater by 1 than before this edge was incremeneted. If no new path is found, our graph is still separated by this cut such that no new additional flow was diverted from source to sink.

Runtime analysis: Besides our constant increment of an edge capacity our computation is this additional breadth-first-search to check for a new path. This is done in $O(|V| + |E|)$ such that our algorithm works in time $O(|V| + |E|)$

Proof of Correctness: We consider both cases

ii) the edge does not span the cut

In this case then no path from source to sink could now exist as this cut still exists unbridged.

i) the edge does span the cut

In this case then a path from source to sink could now exist. We run a breadth-first-search between the source and sink to attempt to find this path given it exists. If this path exists we know the new max flow must be an increment by 1 as we have only bridged the cut such that 1 flow unit can be sent through it. If this new path does not exist, however, then we know no succesful flow is sent through the cut such that our max flow has not be incremented.

We now consider case 2) where the capcity of an edge is decreased by 1. We note that Ford-Fulkerson always terminates and returns us a residual graph with a minimum cut such that the source and sink are on opposite sides of said cut. We therefore can consider two subcases: i) the edge is on the cut ii) the edge is not on the cut

i) if the edge is on the cut we decrement this edge and run a breadth-first-search to see if we can find a path from $u \to v$ where our edge is defined as $\{u, v\}$ such that it points out from u into v. If this is the case, then the flow that we have decremented can be diverted elsewhere. If this is not the case, then we have just decremented the max flow of the graph as 1 flow unit had to be removed without anywhere to divert this lost flow potential to.

ii) if the edge is not on the cut then this poses no problem as only the edges are on the cut have max utilization such that decrementing the potential of a non-cut edge is non-problematic as this edge has not reached capacity.

Runtime Analysis: Besides our constant increment of an edge capacity our computation is this additional breadth-first-search to check for a new path. This is done in $O(|V| + |E|)$ such that our algorithm works in time $O(|V| + |E|)$

Proof of Correctness: We consider both cases

i) the edge doesspan the cut

In this case we know that unit of flow has now had its path disconnected. By running a breadth-first-search from $u \to v$ for $edge = \{u, v\}$ we can determine if there is an available path by which we can divert this lost unit of flow such that we maintain the same max flow. If this path is found this is the case because Ford-Fulkerson could use this alternate path instead of the now severed route. If we cannot find this path, however, then our max flow is decremented by 1 as we cannot send this unit of flow because there is no way to travel it from $u \to v$ such that we have it depart from the source to u and from v travel to the sink.

ii) the edge does not span the cut

In this case we know this lost potential unit of flow is irrelevant as only the edges in the cut are at maximum capacity. As a result this decremented edge capacity does not affect available paths as this edge was previously underutilized such that it had ≥ 1 available unutilized flow. Therefore no path could have been severed which previously provided flow to our max flow.