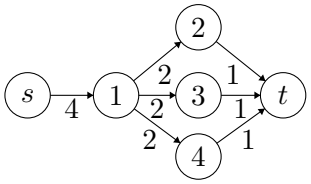
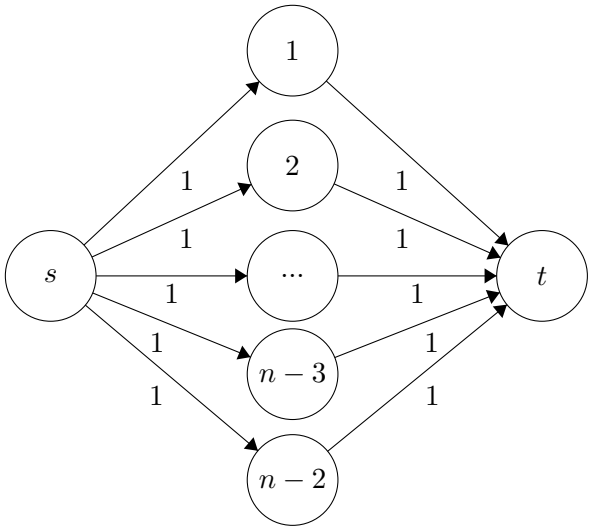
1.
   - **False**, $\Sigma_{i \in S} i$ is the sum of all elements in S but $\Sigma i : 2ii$ is sum of some elements in i, in fact $\Sigma_{i \in S} i = 55$ but $\Sigma_{i:2i \in S} i = 1 + 2 + 3 = 6$
   - **False**, $log_2(2^5 n) = log_2(2^5) + log_2(n) = 5log_2(2) + log_2(n) = 5 + log_2(n)$
   - **True**, $\forall u, v \in V$, there exist a path between the two vertices, hence the graph is connected.
   - **True**, Multiplying all edges by a constant C doesn't affect the min cut, hence the new max flow is just old-maxflow*c, c is this case is 2.
   - **False:** Counter example:



   Clearly the maxflow here is 3, and we have only one min cut. but if we increase the flow of all edges by 1, then the maxflow will be 5 not 6.

2. Here is an Example:



   Claim: every cut has capacity n-2.
   Proof: take any cut, we get a two sets A and B where s is in A and t in B. if node i is in A then it has exactly one edge going to B (since it's connected to t and to no other node in B). so if we have x nodes in A, then we have x edges coming out of A + ((n-2) - x) edges coming out of s, a total of (n-2) edges coming out of A to B giving a total capacity of (n-2). Same for B. Similar reason for B.

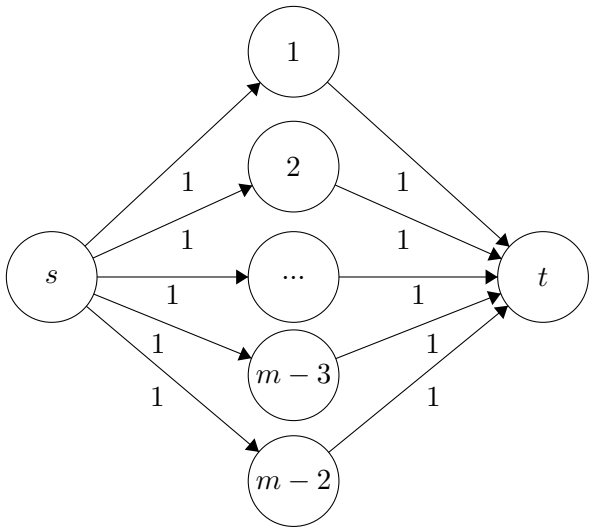
3. The second algorithm is similar to the first one. The paths found in Algorithm2 can be used in Algorithm1. For algorithm1, To find a path with a bottleneck ¿ Δ, we can just check the fattest path. Hence algorithm2 can have at most augmentation as algorithm 1, which is $2m\lceil log_2 K \rceil$

4. (a) Here is an algorithm:
While there is a s-t path with positive flow:
—— Find a s-t path with positive flow
—— Find δ the bottleneck (min edge flow in the path)
—— Subtract δ from every edge's flow.

Claim: There is at most m augmentation.

Proof: There is m edges and at least one edge's flow become 0 at every iteration. Hence there is at most m augmentation.

(b) We use the same example from question 2:



for this graph, there is m nodes (s,t,1,2,..n-2) and 2m edges, the algorithm will require m iterations (at least). hence $\omega(m)$

5. (a) Suppose the edge e is increased by 1. We do one other FF iteration on the new graph. Search if there exists a path in the residual graph that can be increased by 1 (that's O(m)). if there is such a path, we adjust the flow, if not we're done.

(b) Suppose the edge e = (u,v) is the one decreased by 1. Suppose M is the old max flow. if edge e didn't use all its capacity before, if it did not, we're not, but if we did, we have to consider two cases:

- Search for a path o the residual graph that includes e (O(m)), if we can increase it by 1 we adjust the flow, if not we move to next case.
- first we find a path from s to u on the residual graph that we can increase by 1 and same from v to t. (O(m)) then we update the flow.