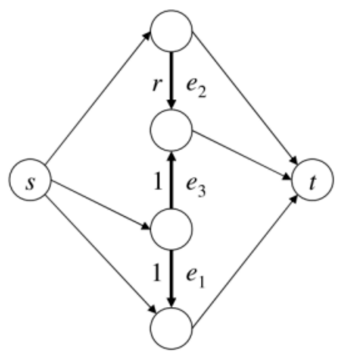
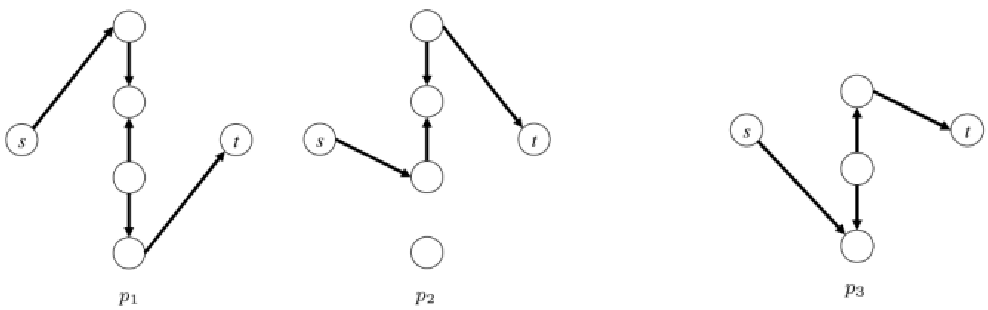
Name: HAVI BOHRA
Roll No.: 210429

Solution (1):
Consider the flow network:



And following paths in the network:



Scenario:
Define a sequence defined by $a_{n+2} = a_n - a_{n+1}$, with $a_0 = 1$ and $a_1 = r$, where $r = (\sqrt{5}-1)/2$
Edges e1, e2, and e3 have capacities 1, r, and 1, respectively, while other capacities are 2.

When following procedure is applied to this network, the Ford-Fulkerson algorithm never terminate.

Procedure:
- Choose a path from the source s to the sink t through e3. Flow value = 1; e3 becomes full. Residual capacities of e1, e2, and e3 are represented as (a0, a1, 0).
- Now, repeat the following steps, where current residual capacities are (an, an+1, 0):
  - Choose path p1, updating the residual capacities to (an+2, 0, an+1).
  - Choose path p2, updating the residual capacities to (an+2, an+1, 0).
  - Choose path p1: updating the residual capacities to (0, an+3, an+2).
  - Choose path p3: updating the residual capacities to (an+2, an+3, 0).

Analysis:
- Each repetition of those 4 steps will lead to increase in flow by amount:
  $2a_{n+1} + 2a_{n+2} = 2a_n$
- Hence, above procedure never terminates. And the flow converges to $1 + 2\sum_{n=1}^{\infty} a_n = 3$. This value does not match the maximum flow 2*2 + 1= 5.

Name: HAVI BOHRA
Roll No.: 210429

Solution (2):
Idea:
Use an array to store the elements and maintain a *'max'* variable to efficiently track the current maximum element.
Operations (Implementations): (Assume |S|=n)

1. **Insert(S, x):**
   a. Append x to the array.
   b. Update the *'max'* variable if x > *max.*

   The amortized cost of insertion is O(1) due to the efficient resizing properties of dynamic arrays.
2. **Delete-Larger-Half(S):**
   a. Employ the Quickselect algorithm (taught in ESO207) to identify the median element in O(n) average-case time.
   b. Partition the array around the median
   c. Remove elements greater than or equal to the median in O(n) time
   d. Update the *'max'* variable
3. **Report-Max(S):**
   a. Return value stored in *'max'* variable.
4. To output the elements of S, it can just iterate over the array and output each element in total of O(n) time.

Amortized Analysis:
Use the following potential function method to analyse the amortized cost of operations:

Φ = 2 · number of elements in the array

Operation	Actual Cost	ΔΦ	Amortized Cost
Insert	c	2*c	3*c
Delete-Larger-Half	n*c	-n*c	0
Report-Max	c	0	c

Hence, Amortized cost of each insert operation is O(1) ⇒ Actual cost of **m** operations is O(**m**).