Course: Analysis of Algorithms
Code: CS33104
Branch: MCA -3rd Semester

Lecture 11 – Flow in Networks

Faculty & Coordinator : Dr. J Sathish Kumar (JSK)

Department of Computer Science and Engineering

Motilal Nehru National Institute of Technology Allahabad,
Prayagraj-211004

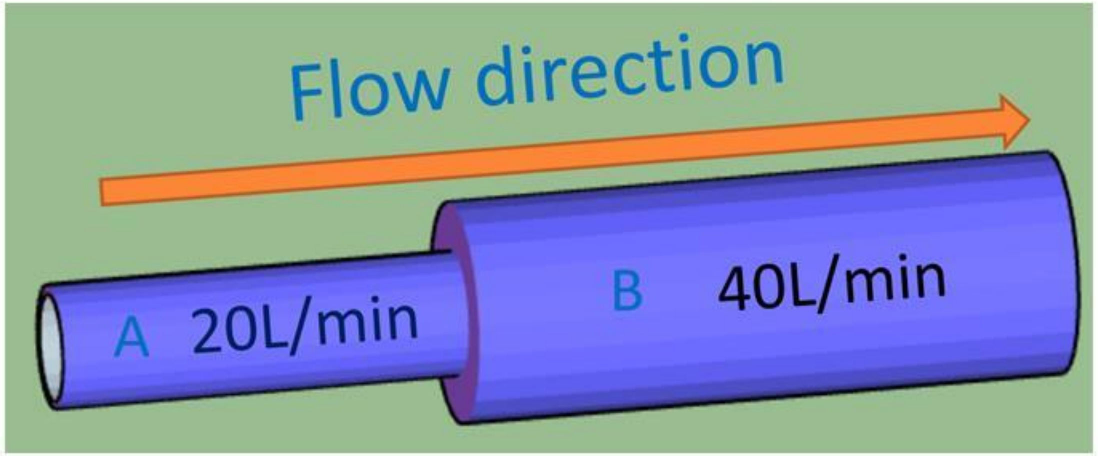
Flow in Networks

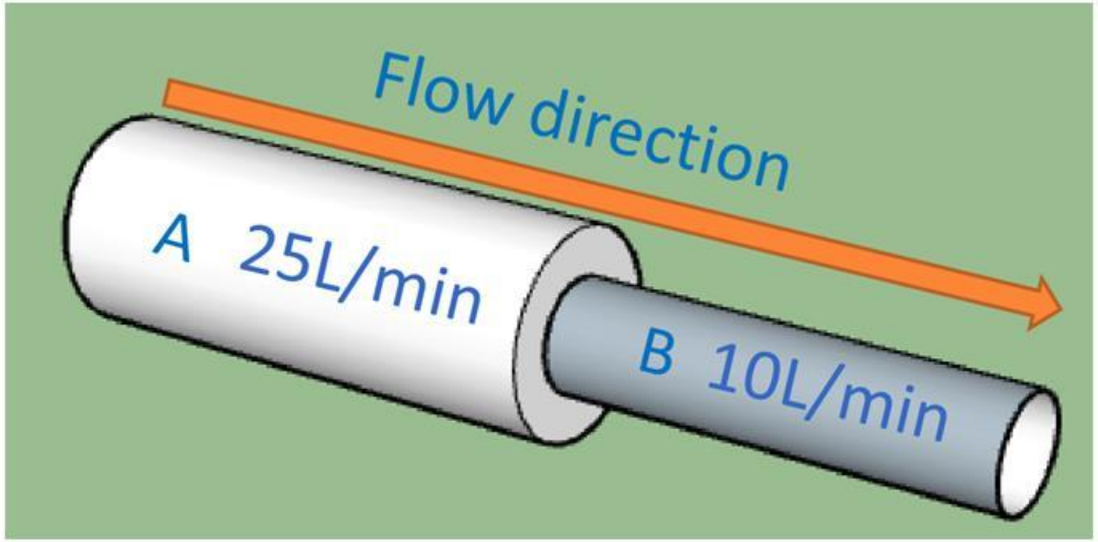
- Directed graph can be interpret as a "flow network" and use it to answer questions about material flows.
  - Imagine a material coursing through a system from a source, where the material is produced, to a sink, where it is consumed.
  - The source produces the material at some steady rate, and the sink consumes the material at the same rate.
  - The "flow" of the material at any point in the system is intuitively the rate at which the material moves.
- Flow networks can model many problems, including
  - liquids flowing through pipes,
  - parts through assembly lines,
  - current through electrical networks, and
  - Information through communication networks.

Flow in Networks

- Each directed edge in a flow network as a conduit for the material.
- Each conduit has a stated capacity, given as a maximum rate at which the material can flow through the conduit, such as 200 gallons of liquid per hour through a pipe or 20 amperes of electrical current through a wire.
- Vertices are conduit junctions, and other than the source and sink, material flows through the vertices without collecting in them.
- In other words, the rate at which material enters a vertex must equal the rate at which it leaves the vertex.
- We call this property “flow conservation,” and it is equivalent to Kirchhoff’s current law when the material is electrical current.

Flow in Networks





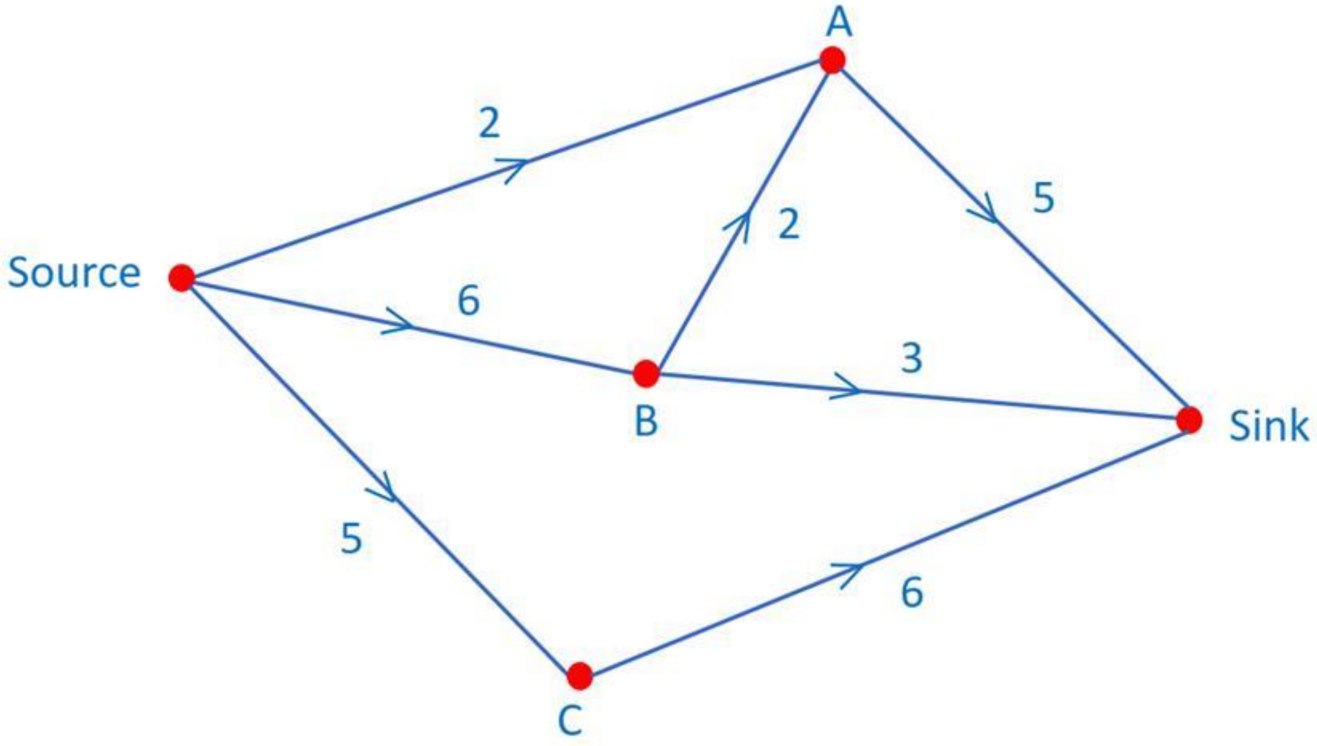
Maximum-flow problem

- In the maximum-flow problem, we wish to compute the greatest rate at which we can ship material from the source to the sink without violating any capacity constraints.
- It is one of the simplest problems concerning flow networks and, this problem can be solved by efficient algorithms.
- Moreover, we can adapt the basic techniques used in maximum-flow algorithms to solve other network-flow problems.
- Classical method of Ford and Fulkerson for finding maximum flows.
  - An application of this method, finding a maximum matching in an undirected bipartite graph

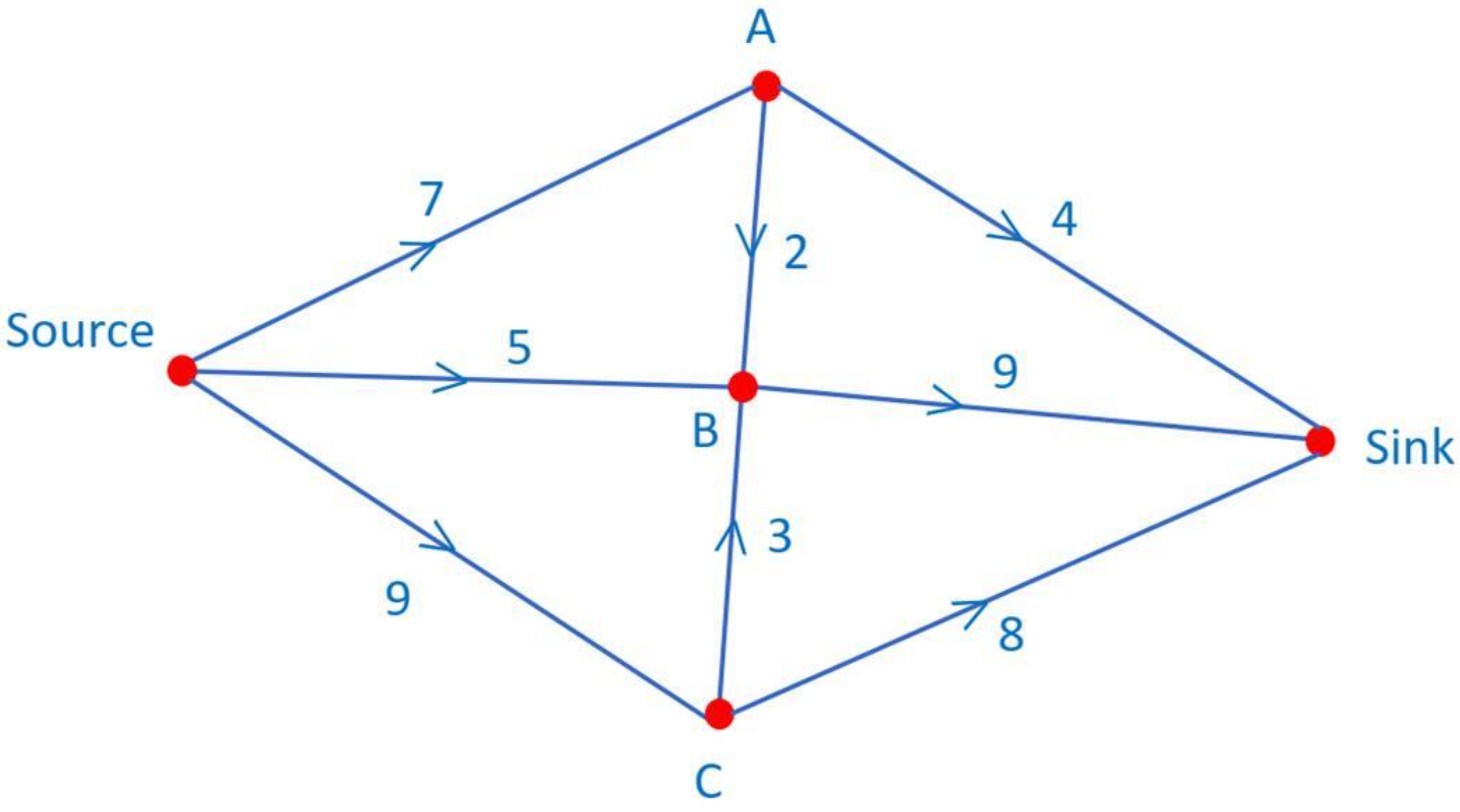
Flow networks and flows

- A ***flow network*** G =(V, E) is a directed graph in which each edge (u, v) ∈ E such that
  - Flowgraph: Directed graph with distinguished vertices s (source) and t (sink)
  - Capacities on the edges, c(u,v) >= 0
- Problem, assign flows f(u,v) to the edges such that:
  - 0 <= f(u,v) <= c(u,v)
  - Flow is conserved at vertices other than s and t
    - Flow conservation: flow going into a vertex equals the flow going out
  - The flow leaving the source is a large as possible
  - Inflow is equal to outflow except source and sink

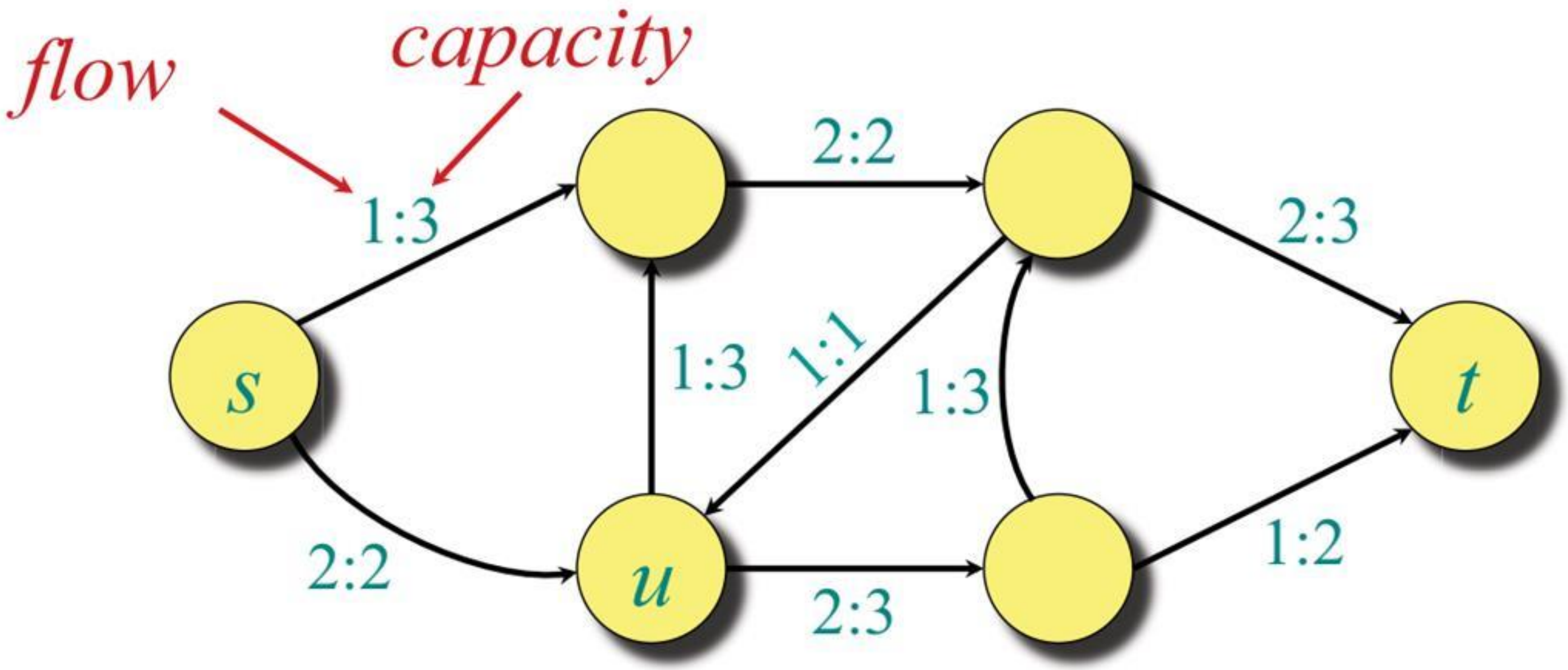
Flow networks and flows



Class Exercise #1



Flow networks and flows

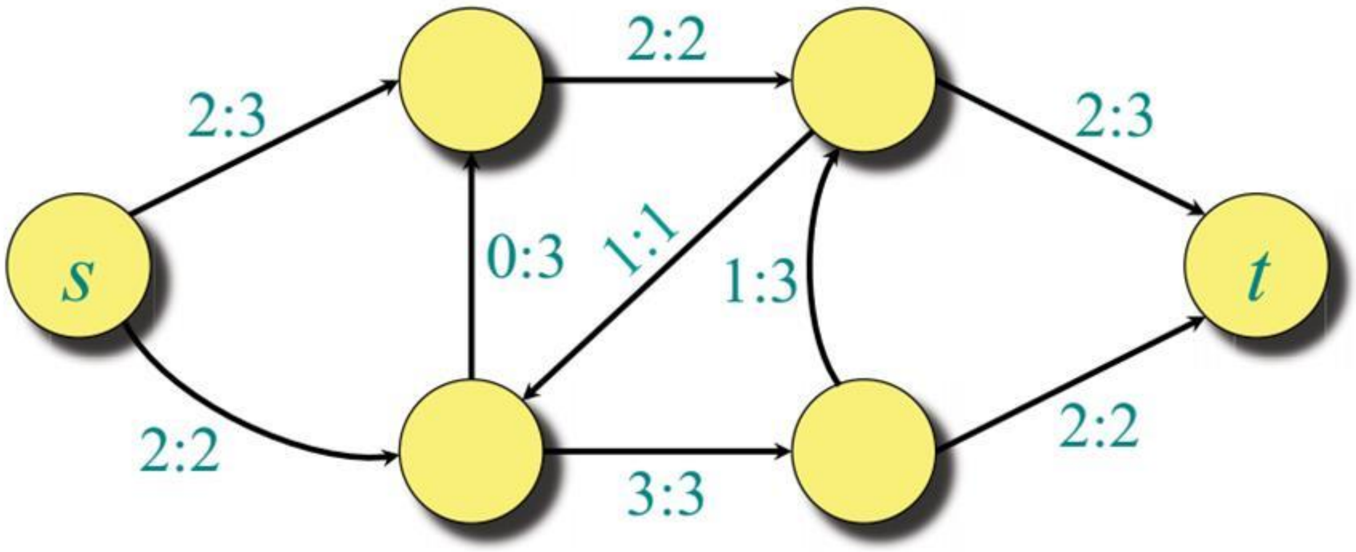


Flow conservation (like Kirchoff's law):

- Flow into *u* is 2 + 1 = 3.
- Flow out of *u* is 1 + 2 = 3.

The maximum-flow problem

- **Maximum-flow problem:** Given a flow network *G*, find a flow of maximum value on *G*.



The value of the maximum flow is 4

Flow in Networks

- Ford-Fulkerson method - 1956
  - $O(E f^*)$
- Edmond Karp method – Early 1970's
  - $O(E^2 V)$
- Dinic's method - 1970
  - $O(E V^2)$
- King, Rao , Tarjan method – 2011
  - $O(E\log_{E/V\log V} V)$
- Orlin method-2013
  - $O(VE)$

Ford-Fulkerson method

FORD-FULKERSON-METHOD(G, s, t)

initialize flow f to 0
while there exists an augmenting path p in the residual network G_f
    augment flow f along p
return f

Residual networks

- Given a flow network G and a flow f , the residual network G_f consists of edges with capacities that represent how we can change the flow on edges of G.
- An edge of the flow network can admit an amount of additional flow equal to the edge's capacity minus the flow on that edge.
- If that value is positive, we place that edge into G_f with a "residual capacity" of $c_f(u,v) = c(u,v)-f(u,v)$.
- The only edges of G that are in G_f are those that can admit more flow; those edges(u,v) whose flow equals their capacity have $c_f(u,v)=0$, and they are not in G_f .

Residual networks

- As an algorithm manipulates the flow, with the goal of increasing the total flow, it might need to decrease the flow on a particular edge.
- In order to represent a possible decrease of a positive flow f(u, v) on an edge in G, we place an edge (v, u) into G_f with residual capacity $c_f(v, u) = f(u,v)$ that is, an edge that can admit flow in the opposite direction to (u, v), at most canceling out the flow on (u, v).
- Sending flow back along an edge is equivalent to *decreasing* the flow on the edge, which is a necessary operation in many algorithms.

Residual networks

More formally, suppose that we have a flow network $G = (V, E)$ with source s and sink t. Let f be a flow in G, and consider a pair of vertices $u, v \in V$. We define the ***residual capacity*** $c_f(u, v)$ by

$$c_f(u,v) = \begin{cases} c(u,v) - f(u,v) & \text{if } (u,v) \in E , \\ f(v,u) & \text{if } (v,u) \in E , \\ 0 & \text{otherwise} . \end{cases}$$

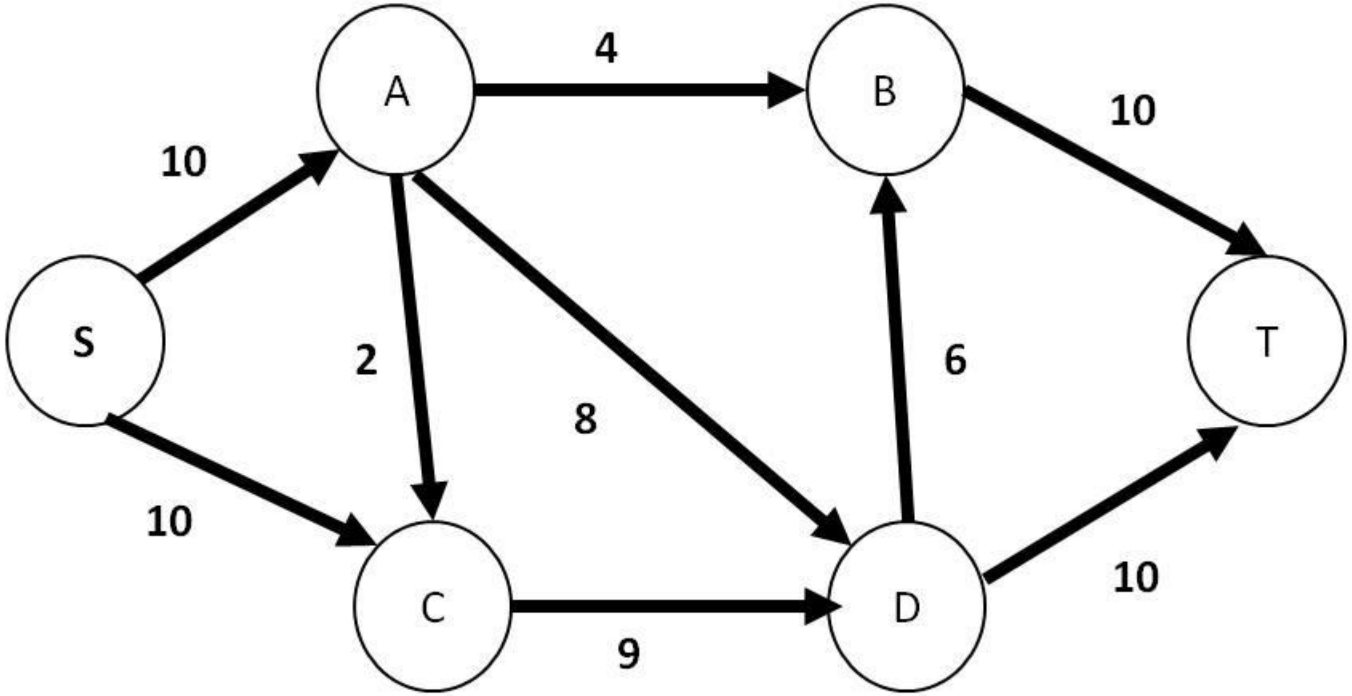
Residual networks

- Pushing flow on the reverse edge in the residual network is also known as ***cancellation***.
- For example, if we send 5 crates of hockey pucks from u to v and send 2 crates from v to u, we could equivalently (from the perspective of the final result) just send 3 creates from u to v and none from v to u.
- Cancellation of this type is crucial for any maximum-flow algorithm.

Augmenting paths

- Given a flow network $G = (V, E)$ and a flow f , an ***augmenting path*** p is a simple path from s to t in the residual network G_f .
- By the definition of the residual network, we may increase the flow on an edge (u, v) of an augmenting path by up to $c_f(u, v)$ without violating the capacity constraint on whichever of (u, v) and (v, u) is in the original flow network G.
- Consider
  - Non Full Forward Edge
  - Non Zero Backward Edge

Example



FORD-FULKERSON ALGORITHM

FORD-FULKERSON(G, s, t)

for each edge $(u, v) \in G.E$

$(u, v).f = 0$

while there exists a path p from s to t in the residual network G_f

$c_f(p) = \min\{c_f(u, v) : (u, v) \text{ is in } p\}$

for each edge (u, v) in p

if $(u, v) \in E$

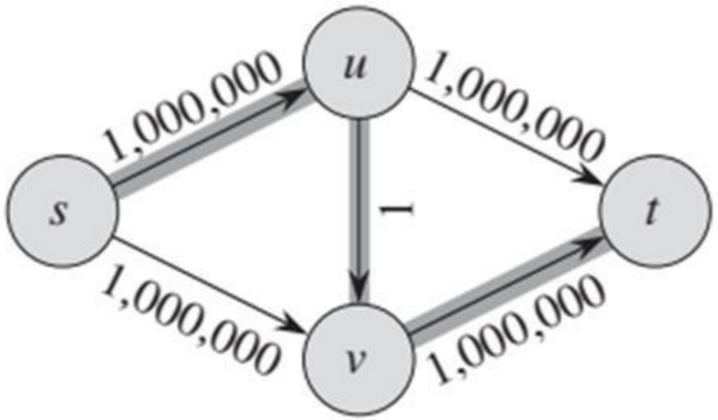
$(u, v).f = (u, v).f + c_f(p)$

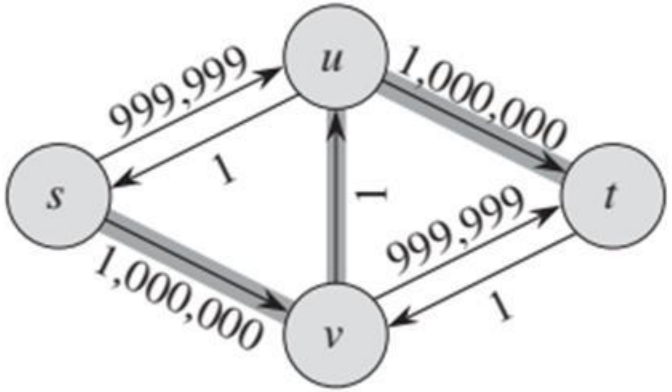
else $(v, u).f = (v, u).f - c_f(p)$

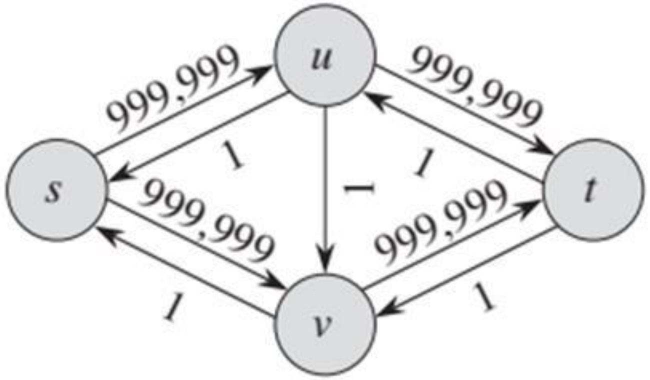
FORD-FULKERSON ALGORITHM

- If f* denotes a maximum flow in the transformed network, then a straightforward implementation of FORD-FULKERSON executes the **while** loop of lines 3–8 at most f* times, since the flow value increases by at least one unit in each iteration.
- The time to find a path in a residual network is therefore O(V + E')=O(E), if we use either depth-first search or breadth-first search.
- Each iteration of the **while** loop thus takes O(E) time, as does the initialization in lines 1–2, making the total running time of the FORD-FULKERSON algorithm O(E f *)

FORD-FULKERSON ALGORITHM





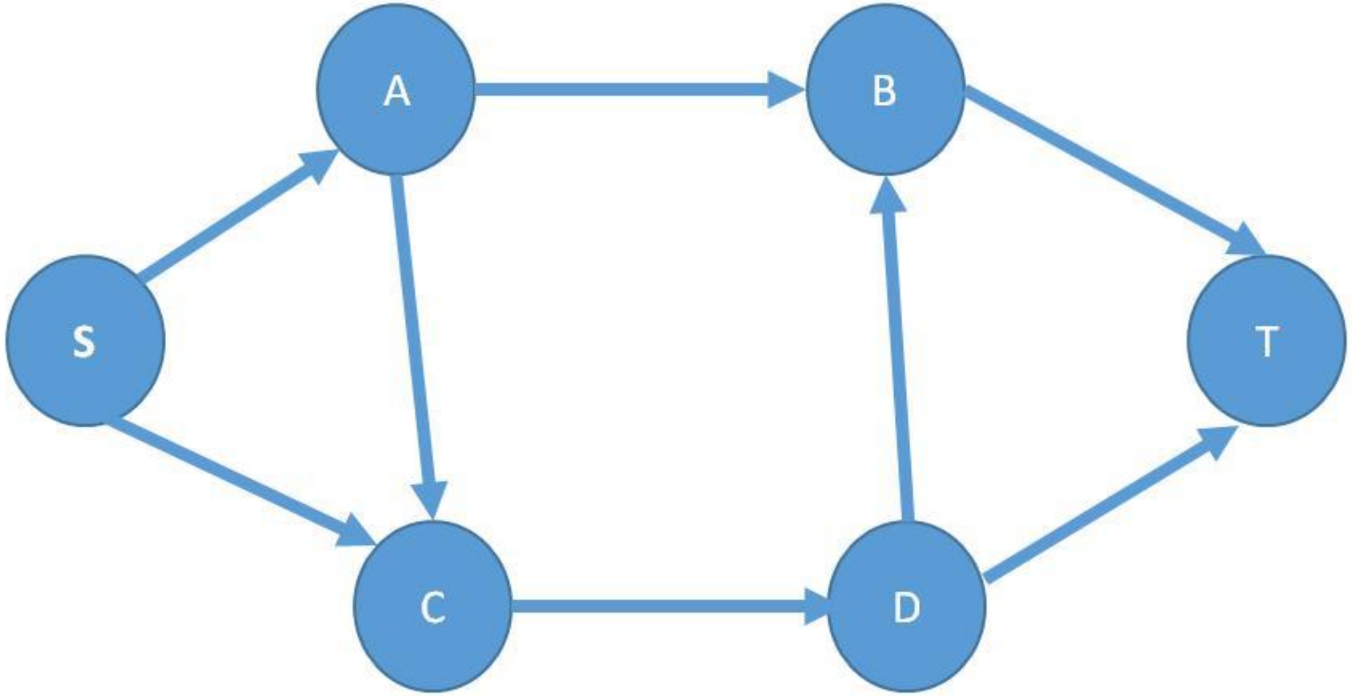


Choosing the augmenting path **s -> u -> v->t** in the odd-numbered iterations and the augmenting path **s -> v -> u->t** in the even-numbered iterations, leads to perform a total of 2,000,000 augmentations, increasing the flow value by only 1 unit in each.

Edmond Karp Method

- Disadvantages of Ford-Fulkerson method
  - Not determined how to choose an augmented path
  - It is observed that choosing an augmented path using DFS makes the situation worst.
  - It is also observed that choosing an augmented path using BFS makes the situation always better.
- Edmonds-Karp algorithm implemented based on Ford-Fulkerson method by applying BFS to choose the augmenting path as a *shortest* path from s to t in the residual network, where each edge has unit distance (weight).

Edmond Karp Method



Edmond Karp Method

```
f = 0;
res_graph = net_graph
while res_graph contains an s − t path P do:
    Suppose P be an s − t path in the residual_graph with of edges.
    P = Breadth-First-Search(C, E, s, t, F)
    Augment maximum_flow using P.
    u = P[v]
    F[u, v] = F[u, v] - m
    Update residual_graph
    F[v, u] = F[v, u] + m
    v = u
end while
return maximum_flow
```

Edmond Karp Method

- Because there are O(|E|) total pairs of vertices that can, for the edge (u, v), become critical O(|V|) times, the total number of iterations that Edmonds-Karp can go through is O(|V|·|E|).
- Total Iterations is O(VE), where V-vertices and E-Edges
- The time to find a path in a residual network is therefore O(V + E')=O(E), if we use breadth-first search.
- Total time complexity = $O(VE^2)$

Dinic’s method

- Dinic’s method includes construction of level graphs and residual graphs and finding of augmenting paths along with blocking flow.
- Level graph is one where value of each node is its shortest distance from source.
- Blocking flow includes finding the new path from the bottleneck node.
- Residual graph and augmenting paths are same as earlier discussed.

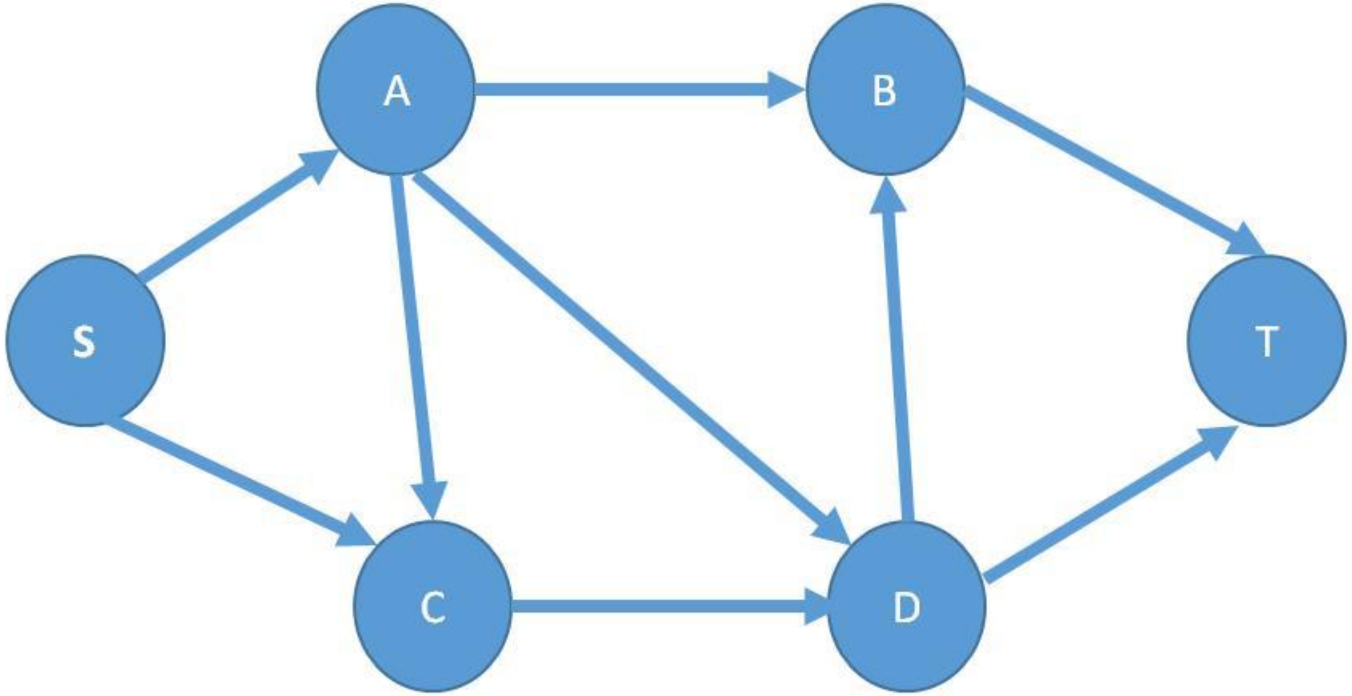
Edmond Karp Method

- https://www.educative.io/answers/what-is-the-edmonds-karp-algorithm
- https://jamieheller.github.io/theory.html

Dinic’s method

```
function: DinicMaxFlow(Graph G,Node S,Node T):
  Initialize flow in all edges to 0, F = 0
  Construct level graph
  while (there exists an augmenting path in level graph):
    find blocking flow f in level graph
    F = F + f
    Update level graph
  return F
```

Dinic's method : Example



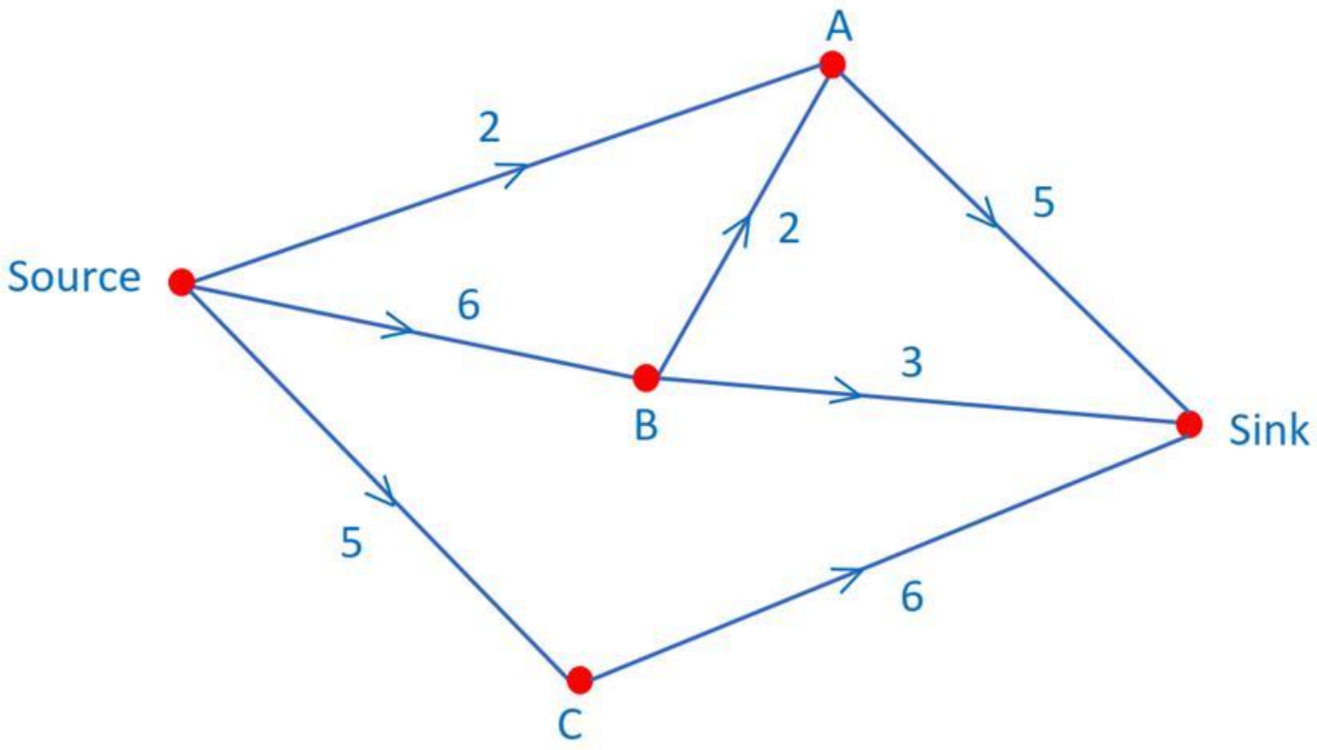
Dinic's method

- The number of layers in each blocking flow increases by at least 1 each time and thus there are at most V-1, blocking flows in the algorithm.
  - the level graph G_L can be constructed by BFS O(E) time.
  - a blocking flow in the level graph G_L can be found in O(VE).
  - Running time O(E+VE)=O(VE).
- Total running time is $O((V-1)* VE))=O(V^2E)$

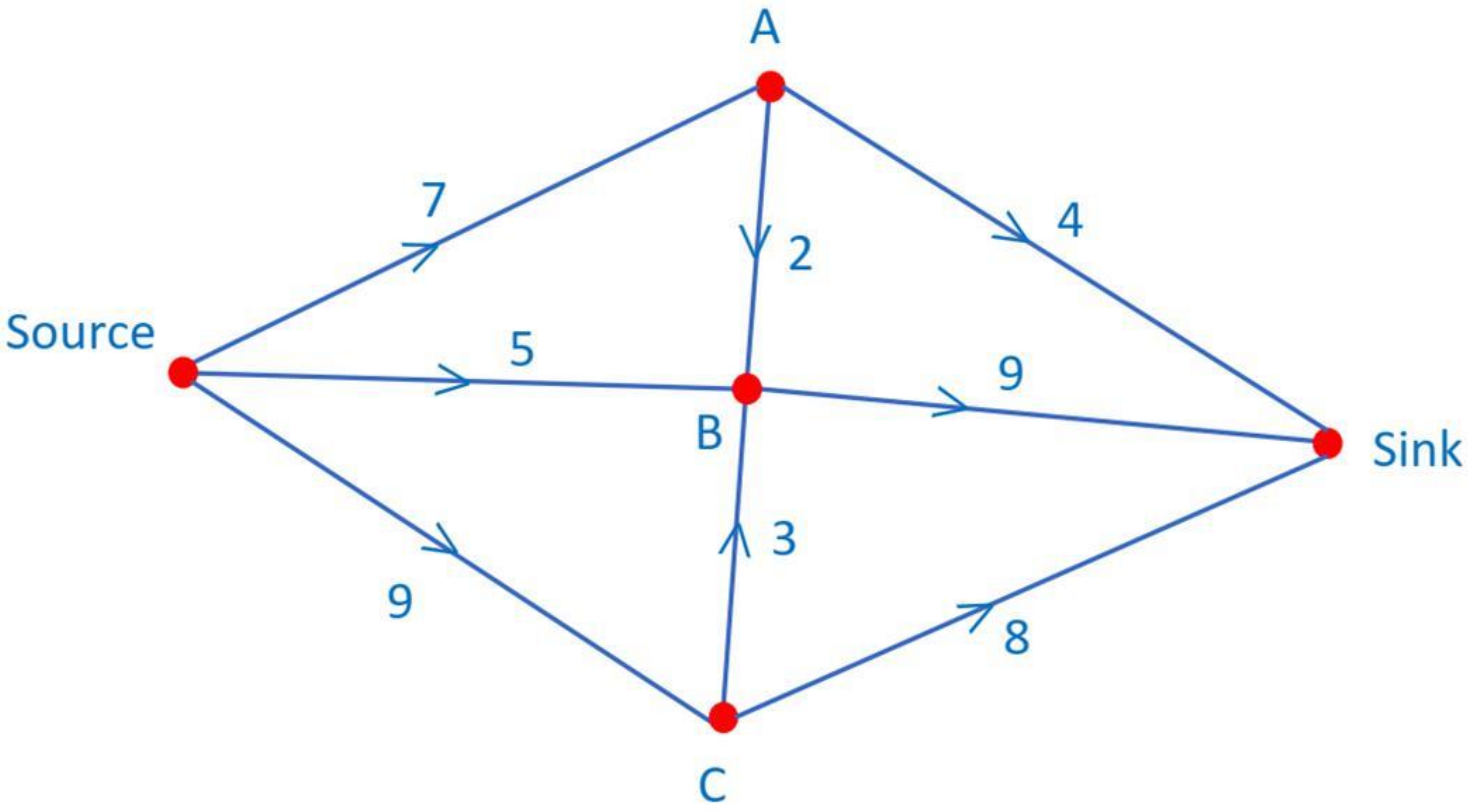
Cuts in a graph

- A **cut**(S, T) of flow network G =(V, E) is a partition of V into S and T = V - S such that s ∈ S and t ∈ T.
- In other words, Cut: Partition of V into disjoint sets S, T with s in S and t in T.
- A ***minimum cut*** of a network is a cut whose capacity is minimum over all cuts of the network.
- There exists a flow which has the same value of the minimum cut
- If we want to find a minimum cut, we begin by looking for a maximum flow.

Flow networks and flows



Class Exercise #1



Latest Research Paper Assignments – Phase 2

- **Application: Healthcare Management using Blockchain**
- Explore papers that are recently published from 2021-2022 and 2023 from IEEE, ACM, Science Direct, Springer and other SCI Journals.
- Prepare the report on the following terms and submit
  - Paper Title
  - Problem Title
  - Complexity of the problem
  - Advantages of the problem
  - Limitations of the problem
  - Brief summary of the overall paper in your own words (approx 50-100)