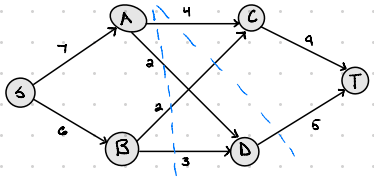
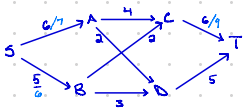
7.17) Network:



a) Max Flow and min cut

1) Find Shortest path

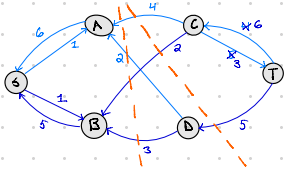
2) Increase Flow along the path



Max Flow: 11 units

min cut: Diagonal through A→C, B→C, D→T

b) Residual Graph G_F (with edge capacities). Mark vertices reachable from S and T.



All nodes are reachable from T

Nodes A and B are reachable from S

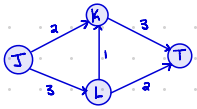
c) Bottleneck Edges

Edge $A \to C$

$B \to C$

($B \to D$, $D \to T$) also a bottleneck, but both are required to increase max flow

d) Simple Example without Bottlenecks



e) Algorithm to Identify Bottlenecks

Edmonds-Karp Algorithm $O(|V||E|^2)$ +

```
Bottlenecks(G):
    f(e) = 0   ∀ e ∈ G
    while ∃ a path from s to t in G^f
        p: Shortest path from p → t (BFS)
        c(p): capacity of p
        augment the flow along p
        recompute G
        recompute G^f
```

$O(|V|+|E|)$ Run BFS on (G^f, s), but only on residual edges (no back edges)

A = nodes reachable from S

$O(|V|+|E|)$ Run BFS on (G_f^R, t), but only with residual edges

B = nodes reachable from t

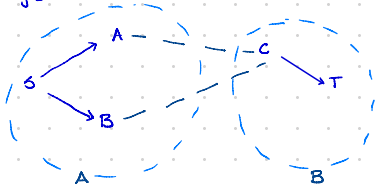
For each edge $(u,v) \in G$:

    if ($u \notin A$ and $v \in B$):

        (u,v) is a bottleneck

Total time $O(|V|+|E|^2)$

Break the graph into only the residual edges:



Any edges that cross between A and B in the original graph are bottlenecks

7.21) Give efficient Algorithm to Find critical edges.

critical edge → edge that exhausts its capacity

↳ Edmund-Karp Algorithm

↳ G^F → traverse from t, Find all edges without a back edge ⇒ Critical

Critical Edges (G, s):

Edmund-Karp $O(|V|+|E|^2)$

```
f(e) = 0   ∀e ∈ G
while ∃ a path from s to t in G^F
    p: Shortest path from p → t (BFS)
    c(p): capacity of p
    augment the flow along p
    recompute G
    recompute G^F
```

Run modified BFS on G^F, starting from T:

```
critical = []
Q = [T]  // Priority Queue
prev = T
while Q not empty:
    u = eject Q
    backedge = False
    for all v ∈ N(u):
        if not visited:
            add v to Q, set visited flag
        if v == prev:
            Add (u,v) to critical
    prev = u
return critical
```

BFS $O(|V|)$

Total Time: $O(|V|+|E|^2)$

p → returns (u, v, w)

Change the forward edges → curr - min

Add the backwards edges → all the same.