# Intro to Algorithms: Homework #10

Due on April 28, 2021

*Prof. Zaki*

**Jared Gridley**

Lab Results:

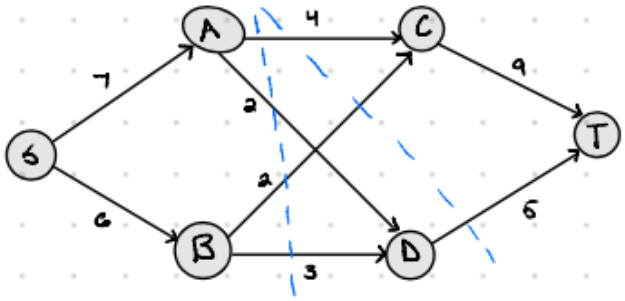
```
Results:

g50:
cut edges ('0', '103', 57), ('0', '1073', 14), ('0', '1083', 39), ('0', '1158', 3), ('0',
'1187', 13), ('0', '1219', 6), ('0', '1262', 55), ('0', '1318', 4), ('0', '1336', 29),
('0', '1343', 18), ('0', '1363', 19), ('0', '1417', 21), ('0', '1433', 18), ('0', '1468',
26), ('0', '1484', 25), ('0', '1495', 31), ('0', '150', 9), ('0', '1512', 38), ('0',
'160', 50), ('0', '1639', 8), ('0', '1688', 9), ('0', '1697', 11), ('0', '173', 7), ('0',
'1754', 16), ('0', '1778', 11), ('0', '1783', 1), ('0', '1799', 24), ('0', '189', 23),
('0', '1901', 7), ('0', '1964', 33), ('0', '197', 40), ('0', '2021', 7), ('0', '2049',
3), ('0', '210', 3), ('0', '2139', 6), ('0', '214', 15), ('0', '2183', 2), ('0', '221',
17), ('0', '2249', 31), ('0', '2399', 18), ('0', '2413', 15), ('0', '2473', 14), ('0',
'2509', 6), ('0', '276', 21), ('0', '329', 3), ('0', '384', 26), ('0', '404', 4), ('0',
'451', 7), ('0', '520', 19), ('0', '529', 2), ('0', '543', 16), ('0', '597', 15), ('0',
'6', 50), ('0', '605', 23), ('0', '647', 4), ('0', '67', 16), ('0', '676', 2), ('0',
'68', 2), ('0', '695', 8), ('0', '715', 40), ('0', '741', 8), ('0', '762', 13), ('0',
'818', 10), ('0', '821', 35), ('0', '86', 58), ('0', '891', 1), ('0', '956', 33), ('0',
'966', 6)
maxflow value 1224

g5:
flow ('0', '24', 59), ('0', '25', 7), ('0', '31', 44), ('0', '32', 3)
maxflow value 113


g10:
flow ('0', '103', 2), ('0', '126', 5), ('0', '136', 12), ('0', '138', 15), ('0', '158',
43), ('0', '159', 35), ('0', '184', 2), ('0', '199', 55), ('0', '20', 63), ('0', '205',
1), ('0', '218', 3), ('0', '244', 16), ('0', '249', 23), ('0', '266', 8), ('0', '276',
1), ('0', '289', 28), ('0', '290', 27), ('0', '297', 19), ('0', '305', 48), ('0', '323',
23), ('0', '343', 8), ('0', '360', 32), ('0', '368', 22), ('0', '381', 10), ('0', '387',
2), ('0', '406', 62), ('0', '415', 71), ('0', '416', 4), ('0', '429', 11), ('0', '43',
4), ('0', '435', 22), ('0', '442', 16), ('0', '444', 28), ('0', '56', 12), ('0', '58',
3), ('0', '60', 19), ('0', '68', 9), ('0', '8', 1), ('0', '88', 14)
maxflow value 779
```

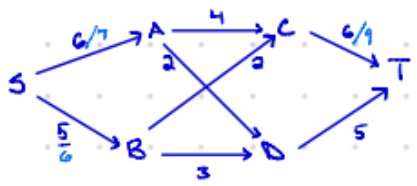
7.17) Network:




a) Max Flow and min cut

1) Find Shortest path

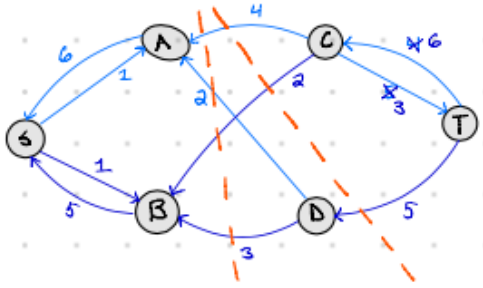
2) Increase flow along the path




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

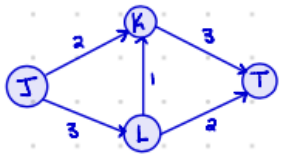
Edge A → C

B → C

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

Figure 1: Page 1

d) Simple Example without Bottlenecks




e) Algorithm to Identify Bottlenecks

Bottlenecks (G):
- $f(e) = 0 \quad \forall_e \in G$
- 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 $G^F$

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

+

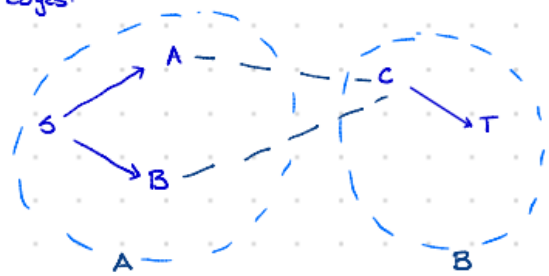
Run BFS on $(G^F, s)$, but only on residual edges (no back edges)
- A = nodes reachable from s

$O(|V|+|E|)$

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 ($u \in A$ and $v \in B$):
  - (u,v) is a bottleneck

Break the graph into only the residual edges:




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

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

Figure 2: Page 2

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

**CriticalEdges(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
```

BFS $O(|V|)$

```
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
```

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

Figure 3: Page 3