ADS2 Exam Notes – Week 4 Network Flows (Max-Flow/Min-Cut)

Field	Value
Title	ADS2 Exam Notes – Network Flows (Ford–Fulkerson, Min-Cut, Properties, Applications)
Date	2025-10-18 (Europe/Copenhagen)
Author	ADS2 compact notes
Sources used	`network1.pdf` (weekplan, p.1–2), `flow1-4x1.pdf` (slides), `week4_exer_7_1_and_7_2.pdf` (KT 7.1–7.2 figures), `week4_exer_7_4_and_7_5.pdf` (KT 7.4–7.5 statements), figures `Pasted image.png`, `Pasted image (2).png`
Week plan filename	network1.pdf

General Methodology and Theory

- **Flow network** on a directed graph with capacity $c(u,v) \geq 0$. A **flow** f obeys capacity and conservation except at s,t; value $|f| = \sum_v f(s,v) = \sum_v f(v,t)$.
- **Residual graph** G_f with forward capacity $c(u,v) - f(u,v)$ and backward capacity $f(u,v)$.
- **Augmenting path** = any $s \to t$ path in G_f; push by bottleneck $\delta = \min c_f(\cdot)$.
- **Ford–Fulkerson** (FF) repeats augmentation until no path in G_f. With BFS paths (**Edmonds–Karp**) the time is $O(nm^2)$.
- **Cut** (S,T) with $s \in S, t \in T$ has capacity $c(S,T) = \sum_{u \in S, v \in T} c(u,v)$. Always $|f| \leq c(S,T)$; equality holds for max flow and min cut (theorem).

Micro-pseudocode (slides-first):

```
Algorithm: edmonds_karp
Input: G=(V,E), c, s, t
Output: max flow f, value |f|
initialize f≡0
while (parent ← bfs_augmenting_path(G,c,f,s,t)) ≠ ⊥:
  δ ← bottleneck along parent path
  augment f by δ along the path (use back-edges)
return f, |f|
// Time: O(n m^2); Space: O(m)
```

Notes

- When capacities are integers, FF terminates and increases $|f|$ by at least 1 per augmentation.
- To **certify optimality fast** in small instances: after the last augmentation, let S be the vertices reachable from s in G_f; then $c(S,T) = |f|$ is a min cut. This avoids redrawing the entire residual graph on paper.
- Avoid transcribing edge lists; instead record a short **augmentation trace** and a **cut certificate**.

Coverage Table

Weekplan ID	Canonical ID	Title/Label (verbatim)	Assignment Source	Text Source	Status
1	KT 7.1 & 7.2	Max flow / min cut (two small graphs)	network1.pdf §1 p.1	week4_exer_7_1_and_7_2.pdf (figures)	✅solved
2	—	Ford–Fulkerson on given graph	network1.pdf §2 p.1	Pasted image.png (exercise figure)	✅solved
3	KT 7.4	Properties of maximum flows	network1.pdf §3 p.1	week4_exer_7_4_and_7_5.pdf (statement)	✅solved
4	KT 7.5	Properties of minimum cuts	network1.pdf §4 p.1	week4_exer_7_4_and_7_5.pdf (statement)	✅solved
5.1	—	Zombie breakout — Save the capital	network1.pdf §5.1 p.1	weekplan text (p.1)	✅solved
5.2	—	Zombie breakout — Distributing the vaccine	network1.pdf §5.2 p.1	weekplan text (p.1)	✅solved
6	CSES 1694	Download Speed (max-flow)	network1.pdf §6 p.1	CSES statement (external)	✅ algorithm only

No omissions; all weekplan items above have solution stubs completed.

Solutions

Exercise 1 — KT 7.1 (Fig. 7.24) and KT 7.2 (Fig. 7.25)

Assignment Source: network1.pdf §1 p.1. **Text Source:** week4_exer_7_1_and_7_2.pdf (figures in `Pasted image (2).png`).

- **KT 7.1 (all capacities 1):** enumerating valid cuts (S, T) with $s \in S, t \in T$ gives minimum capacity 2. Representative minimum cuts: $S \in \{\{s\}, \{s, v\}, \{s, u, v\}\}$.
  ✅**Answer:** min cut value 2 (any of the above).

- **KT 7.2 (weighted):** by direct cut enumeration the minimum capacity is 4, attained by $S = \{s, v\}$.
  ✅**Answer:** min cut value 4 with certificate $S = \{s, v\}$.

Pitfalls: forgetting that $u \to v$ contributes only when $u \in S, v \in T$.

Transfer Pattern (mapping guide):

- **Archetype name:** Min s–t cut on a small directed graph.
- **Recognition cues:** "find the minimum cut", few nodes/edges, capacities given on arcs, single s, t.
- **Mapping steps:** (1) Fix $s \in S, t \in T$. (2) Enumerate plausible S sets (on very small graphs) or argue via max-flow. (3) For each S, sum only edges crossing $S \to T$. (4) Choose the smallest sum.
- **Certificate form:** exhibit S and list its crossing edges with capacities and their sum.
- **Anti-cues / pitfalls:** counting edges entering S; mixing undirected with directed; forgetting parallel edges' combined effect.

Exercise 2 — Ford–Fulkerson on the provided graph

Assignment Source: network1.pdf §2 p.1. **Text Source:** `Pasted image.png`.

Upper bounds: $\text{out}(s) = 14,\ \text{in}(t) = 10$. We exhibit a flow of value 9 and a matching min cut of capacity 9.

Augmentation trace (one valid sequence):

Step	Path	Bottleneck δ	Saturated edges after step	Residual note
1	$s \to L \to F \to A \to t$	3	partial on $A \to t$	back-edges appear
2	$s \to L \to F \to G \to A \to t$	3	$F \to A$ now full	$G \to A$ carries 3
3	$s \to L \to F \to G \to t$	1	$G \to t$ full	—

Step	Path	Bottleneck δ	Saturated edges after step	Residual note
4	$s \to C \to G \to t$	1	—	feeds via C arm
5	$s \to C \to B \to M \to t$	1	—	one unit via M

Cut certificate: let $S = \{s, L, F\}$, $T = V \setminus S$. Crossing edges $\{s \to C, F \to A, F \to G\}$ sum to $2 + 3 + 4 = 9$.
✅**Answer:** max flow $= 9$; min cut $S = \{s, L, F\}$ with capacity 9 .

Checks: feasibility at each node; value into t equals 9 ; bound $|f| \leq c(S, T)$ tight.

Variant drill: raising $M \to t$ by 1 doesn't help unless its feeder $B \to M$ (and upstream $s \to C$) increase.

Transfer Pattern (mapping guide):

- **Archetype name:** Max-flow by augmenting paths (Ford–Fulkerson / Edmonds–Karp).
- **Recognition cues:** given a specific network with s, t and capacities; asked to "compute a maximum flow" or "show a min cut".
- **Mapping steps:** (1) Work in the residual graph; choose BFS (EK) for determinism. (2) Record an augmentation trace table: step, path, δ , saturations. (3) After no path remains, take $S =$ nodes reachable from s in G_f and sum crossing capacities.
- **Certificate form:** final flow value and a cut (S, T) with listed crossing edges summing to that value.
- **Anti-cues / pitfalls:** forgetting back-edges; violating conservation; writing full edge lists (not needed).

Exercise 3 — KT 7.4 (Properties of maximum flows)

Assignment Source: network1.pdf §3 p.1. **Text Source:** week4_exer_7_4_and_7_5.pdf.

Claim (KT 7.4 paraphrase): "In any network with positive integer capacities, every maximum $s \to t$ flow saturates all edges out of s ."

Verdict: False.

Counterexample (minimal): vertices $\{s, a, b, t\}$ with edges $s \to a\ (1),\ s \to b\ (1),\ a \to t\ (1)$, no arc from $b \to t$. The maximum flow has value 1 (via a) and **does not** saturate $s \to b$.

Why the claim fails: max flow only needs to saturate edges on all minimum cuts; other outgoing edges of s can remain slack if they do not lead to t in the residual sense.

✅**Answer:** False; counterexample above suffices.

Transfer Pattern (mapping guide):

- **Archetype name:** Counterexample builder for universal flow claims.

- **Recognition cues:** statements of the form “every maximum flow must …” or “for all networks …”.
- **Mapping steps:** (1) Identify the asserted property. (2) Build a 3–4 node network that prevents the property while keeping a valid max flow. (3) Prove optimality of your flow (e.g., show a min cut).
- **Certificate form:** explicit small graph and a max-flow/min-cut witness contradicting the claim.
- **Anti-cues / pitfalls:** using zero-capacity edges when the claim assumes positive; failing to prove maximality.

Exercise 4 — KT 7.5 (Add 1 to every capacity)

Assignment Source: network1.pdf §4 p.1. **Text Source:** week4_exer_7_4_and_7_5.pdf.

Claim (KT 7.5 paraphrase): “If (A, B) is a minimum cut for capacities $\{c_e\}$, then after replacing every edge capacity by $1 + c_e$, the same cut (A, B) is still minimum.”

Verdict: False.

Counterexample (explicit): many small constructions work; here is one. Have five unit edges leaving s into a bottleneck node u that then connects to t with capacity 6. Initially $\{s\}$ is the min cut with value 5. After adding 1 to every capacity, that cut has value 10, while the cut enclosing u has value 7, so the minimum cut **changes**.

Reason: adding a constant to each edge changes every cut by “number of crossing edges,” which differs across cuts; order can flip.

✅**Answer:** False; counterexample above.

Transfer Pattern (mapping guide):

- **Archetype name:** Cut-order instability under uniform capacity shifts.
- **Recognition cues:** global operations applied to **all** edges (add k , multiply by α , etc.), question asks whether a specific min cut remains min.
- **Mapping steps:** (1) Construct two near-optimal cuts with different numbers of crossing edges. (2) Apply the uniform change; the cut with fewer crossing edges becomes cheaper.
- **Certificate form:** before/after cut capacities with sums exhibiting the flip.
- **Anti-cues / pitfalls:** choosing cuts with equal crossing-edge counts (no flip possible).

Exercise 5.1 — Zombie breakout: Save the capital

Assignment Source: network1.pdf §5.1 p.1. **Text Source:** weekplan text.

Goal: minimum number of directed roads to remove so no infected city can reach the capital.

Reduction to min-cut: create a **super-source** s^* with arcs $s^* \to x$ to each infected city x (very large capacity). Let the **capital** be the sink t . Give every road capacity 1 and compute a minimum $s^* \to t$ cut. The cut edges are exactly the roads to destroy; their count equals the min-cut value.

Complexity: with $N = X + Y$ cities and R roads, Edmonds–Karp runs in $O(NR^2)$; Dinic faster in practice. Construction is linear in $N + R$.

Correctness sketch: any set of removed roads that separates all infected nodes from t is an $s^* \to t$ cut; the minimum such set is the minimum cut.

✅**Answer:** remove exactly the edges in a minimum $s^* \to t$ cut of the unit-capacity road network.

Transfer Pattern (mapping guide):

- **Archetype name:** Minimum edge cut separating two sets (multi-source to sink).
- **Recognition cues:** "fewest roads to destroy/block so group A cannot reach B", unit costs or per-edge weights, directed map/roads.
- **Mapping steps:** (1) Super-source into all sources in A; choose sink in B. (2) Unit capacity for cardinality; weights for costs. (3) Run min-cut; interpret crossing edges as removals.
- **Certificate form:** cut (S, T) with listed roads and count (or cost).
- **Anti-cues / pitfalls:** undirected roads need both directions or symmetric arcs; forgetting to include all infected sources.

Exercise 5.2 — Zombie breakout: Distributing the vaccine

Assignment Source: network1.pdf §5.2 p.1. **Text Source:** weekplan text.

Goal: send 10 doctors to each uninfected city in one day; infected cities allow at most 50 doctors to **pass through** per day (node capacity 50).

Reduction with node capacities: split each infected city v into $v_{in} \to v_{out}$ with arc capacity 50; redirect every incoming road to v_{in} and every outgoing road from v_{out} . Treat the capital as source with large out-capacity. Add arcs $u \to t^*$ of capacity 10 from each uninfected city u to a **super-sink** t^* . Compute a max flow to t^* and check if the value equals $10Y$.

✅**Answer: Yes** iff the max flow from the capital to t^* equals $10Y$.

Complexity: graph grows by $O(X)$ due to splitting; run time as in 5.1.

Transfer Pattern (mapping guide):

- **Archetype name:** Node-capacity flow with per-destination quotas.
- **Recognition cues:** transit-through limits on specific nodes; fixed demand q per destination; "can we route q to each city?".
- **Mapping steps:** (1) Node-split every capacity-limited node. (2) Add per-destination arcs to super-sink with capacity equal to required quota. (3) Solve a single max flow and compare value to total quota.
- **Certificate form:** feasible flow of value total-quota; or a min-cut showing impossibility.
- **Anti-cues / pitfalls:** confusing supply vs. transit constraints; forgetting to protect non-infected nodes with high capacity.

Exercise 6 — CSES 1694 “Download Speed” (max-flow)

Assignment Source: network1.pdf §6 p.1. **Text Source:** CSES problem statement.

I/O card: given n, m and directed edges with capacities, compute a maximum $s \to t$ flow and print its value.

Algorithm: Edmonds–Karp or Dinic. For contest constraints, **Dinic** is recommended.

Pseudocode (capacity-scaling Dinic, outline):

```
Algorithm: dinic_maxflow
Input: G=(V,E), c, s, t
Output: |f|
|f| ← 0; f ≡ 0
while bfs_layered_graph(G,c,f,s,t):
  work[ ] ← 0
  while (δ ← dfs_blocking_flow(s,t,∞)) > 0:
    |f| ← |f| + δ
return |f|
// Typical: O(m n) general; faster on unit graphs and with scaling.
```

Checks: use 64-bit for sums; guard against parallel edges by summing or storing separately.

Transfer Pattern (mapping guide):

- **Archetype name:** Plain max-flow evaluation.
- **Recognition cues:** input is a graph with capacities and distinguished s, t ; output is a single number $|f|$.
- **Mapping steps:** (1) Build adjacency with residual edges. (2) Run Dinic/Edmonds–Karp. (3) Output $|f|$.
- **Certificate form:** not required by judge; locally verify with min-cut if needed.
- **Anti-cues / pitfalls:** 32-bit overflow on totals; mishandling parallel edges; forgetting to zero-index per statement.

Puzzle — Four Coins (from weekplan p.2)

Problem: after each of your flips (any subset of coins), the hangman may rotate the table by 0/90/180/270°. Win if all four are heads within 20 moves.

Strategy (sketch): use **orbit classes** under rotation and cycle a fixed sequence of subsets that breaks all rotational symmetries. One winning 10-move sequence is to flip, in order: $\{1\}, \{2\}, \{3\}, \{4\}, \{1,2\}, \{1,3\}, \{2,3\}, \{1,4\}, \{2,4\}, \{3,4\}$ where indices refer to positions relative to the table. After each move, the rotation cannot keep you in the same orbit indefinitely; the Hamming weight strictly increases at some step, and within 10 steps you reach all-heads. (Many equivalent sequences exist.)

Summary

- **Algorithms:** FF, Edmonds–Karp, Dinic; residual graphs; cut certificates.
- **Week results:** KT 7.1 ⇒ 2 ; KT 7.2 ⇒ 4 ; Ford–Fulkerson instance ⇒ max flow 9 with min cut $\{s, L, F\}$ (capacity 9); KT 7.4 ⇒ False (not all edges out of s need saturate); KT 7.5 ⇒ False (adding 1 can change the min cut); Zombies 5.1 ⇒ min edge cut from infected to capital; Zombies 5.2 ⇒ node-capacity flow with demand $10Y$; CSES 1694 ⇒ implement Dinic.
- **Transfer patterns:** min-cut enumeration; augmenting-path computation; node-splitting; super-sink quotas; counterexample builder for universal claims.
- **Notation:** $G = (V, E),\ c,\ f,\ G_f,\ |f|,\ c(S, T), \delta$ (delta).