ADS2 — Week “Network 2” Notes & Full Solutions

Field	Value
Title	ADS2 Network Flow — Week “Network 2”
Date	2025-10-18
Author	ADS2 Notes Copilot
Sources used	weekplan.pdf (pp.1–2); flow2-4x1.pdf (Edmonds–Karp slides); text_book_sections.pdf (Ch.7 excerpts: §§7.3, 7.4, 7.6–7.9); ex_7_14.pdf (KT 7.14); figures: Pasted image.png, Pasted image (2).png, Pasted image (3).png, Pasted image (4).png
Week plan filename	weekplan.pdf

General Methodology and Theory

- **Max flow / min cut** basics, augmenting paths, and **Edmonds–Karp (EK)** (BFS in residual graph → shortest-edge paths; $O(V \cdot E^2)$). **Capacity scaling**: only use residual edges with capacity $\geq \Delta$, halving Δ from the largest power of two $\leq$ max capacity (total $O(E^2 \log C)$).
- **Circulations with demands / lower bounds**: reduce to a single s*→t* max-flow instance; also the right tool for multi-row/column capacity and “exactly-k” constraints.
- **Bipartite matching**: unit capacities give $O(E\sqrt{V})$ (Hopcroft–Karp); with row/column bounds, use a small flow gadget (super-source/sink) or clone-rows trick.
- **Node capacities**: split each v into v_in→v_out with capacity c(v).
- **Certificates**: for each computation give (S,T) cut with crossing capacities that sum to $|f|$.

Notes (quick references)

- EK path = any s→t path in the residual graph found by BFS; bottleneck δ = min residual on its edges. After each augmentation update residual forward/backward edges.
- Scaling rule: start $\Delta = 2^{\lfloor \log_2 C\_max \rfloor}$; while $\Delta \geq 1$, keep augmenting with only edges of residual $\geq \Delta$; then $\Delta \leftarrow \Delta/2$.
- Matching/b-matching model: source → Left side (row-nodes) with capacities (row quotas); edges for allowed placements with cap 1; right side (column-nodes) → sink with column quotas.
- Mixed-graph Euler tour: pre-directed edges fix a **balance** r(v)=in_dir(v)−out_dir(v). If udeg(v) is the number of undirected edges at v, we must choose exactly h(v)=(udeg(v)−r(v))/2 of those to point **into** v. Existence reduces to a bipartite **b-matching** between **undirected edges** and **vertices**.

Coverage Table

Weekplan ID	Canonical ID	Title/Label (verbatim)	Assignment Source	Text Source	Status
1	—	The Edmonds–Karp algorithm and the scaling algorithm (two graphs)	weekplan.pdf p.1	Pasted image.png (both graphs)	Solved
2	KT 7.8	Blood Donations	weekplan.pdf p.1	Pasted image (4).png (table)	Solved
3.1	—	Christmas Trees — model as a graph	weekplan.pdf p.1	Pasted image (2).png (example grid)	Solved
3.2	—	Christmas Trees — algorithm, runtime, correctness	weekplan.pdf p.1	Pasted image (2).png	Solved
4	KT 7.14	Escape	weekplan.pdf p.1	ex_7_14.pdf p. 421	Solved
5	CSES 1696	School Dance	weekplan.pdf p.2	(external task; mapping + I/O)	Solved (method)
6	—	[*] Euler tours in mixed graphs	weekplan.pdf p.2	Pasted image (3).png (example pair)	Solved

---

Solutions

Exercise 1 — EK & Scaling on two graphs

Concept mapping. Compute a max flow and a min cut. Use EK (BFS) and then capacity-scaling. Tie-breaking: lexicographic by node name on adjacency lists.

Left graph (nodes s, L, F, C, B, M, G, A, t).

EK augmentation trace:

Step	Path	δ (delta)	Saturated edges	New residual facts
1	s→C→G→t	2	s→C, G→t	back-edges C→s, t→G get 2
2	s→L→F→A→t	3	F→A	back-edges A→F get 3
3	s→L→F→G→A→t	3	F→G	back-edge G→F gets 3
4	s→L→F→G→C→B→M→t	1	M→t	back-edge t→M gets 1

Final value |f| = **9**. Min-cut certificate: **S = {s, L, F}**; crossing edges and capacities: s→C (2), F→A (3), F→G (4); sum **2+3+4 = 9 = |f|**.

Scaling trace (Δ sequence 4,2,1):

- Δ=4: s→L→F→G→A→t, augment 4.
- Δ=2: s→C→G→t (2); s→L→F→A→t (2).
- Δ=1: s→L→F→A→G→C→B→M→t (1).

Same final **|f|=9** and the cut above.

Right graph (A,B,C,D,E,F,G,H with s,t).

EK augmentation trace:

Step	Path	δ	Saturated edges
1	s→t	2	s→t
2	s→D→E→F→t	3	D→E, E→F
3	s→D→E→G→H→t	1	D→E (now full)
4	s→D→A→B→C→F→t	2	B→C (now 3 left), F→t (now 0 left)
5	s→D→A→B→C→F⇢E→G→H→t	3	A→B (now 0 left), back-edge F→E used

Final value **|f|=11**. Min-cut certificate: **S = {s, D, A, B}**, crossing edges s→t (2), D→E (4), B→C (5); sum **2+4+5 = 11**.

Scaling trace (Δ sequence 4,2,1):

- Δ=4: s→D→E→G→H→t, augment 4 (bottleneck D→E).
- Δ=4 (still): s→D→A→B→C→F→t, augment 5 (bottleneck F→t).
- Δ=2: s→t, augment 2.

✅**Answer:** Left graph max flow **9** with cut (S={s,L,F}); Right graph max flow **11** with cut (S={s,D,A,B}).

Pitfalls. Forgetting backward edges after non-unit δ; treating scaling as "pick the fattest path" (it's a Δ-filtered residual BFS).

Variant drill. Reduce A→t from 6 to 4 on the left graph: EK steps 2–3 would reduce to δ=2 in step 2 and δ=2 in step 3, total |f| becomes **8**; cut remains S={s,L,F} but capacity drops to 2+4+2=8.

Transfer Pattern. Archetype: min s–t cut via EK/scaling. Cues: "augmenting path trace," "compute max flow," small labeled graph. Mapping: vertices as given; capacities on arrows; run EK; certify with (S,T) and sum. Certificate: list crossing edges with capacities.

Exercise 2 — KT 7.8 Blood Donations

Model. Donor types O, A, B, AB; patient types O, A, B, AB. Edges reflect compatibility (O→all; A→A,AB; B→B,AB; AB→AB). Capacities: source→donor = supply; patient→sink = demand.

Computation (integer max-flow). One feasible maximum:

- O→O: 44, O→A: 6.
- A→A: 36.
- B→B: 8.
- AB→AB: 3.

Total treated = **97** patients.

Cut certificate & explanation. A must treat 42 but has only 36 type-A units; the extra **6** must come from type O. This leaves at most **50−6 = 44** O-units for O-patients, but O-demand is 45 ⇒ at least **one O-patient cannot be served**. This exhibits a min-cut of value 97 and proves optimality.

Plain-English rationale. Since type O is the only universal donor, any shortfall in A/B/AB must be compensated by O; here A is short by 6, so O can cover at most 44 of its own 45 patients.

Transfer Pattern. Archetype: circulation with multiple supplies/demands. Cues: compatibility table, "can this meet all demand?". Mapping: donors as sources, patients as sinks; capacities as supplies/demands; run max flow; certificate is a small imbalance argument (O is the bottleneck).

✅**Answer:** Maximum patients served **97**; exactly **1 O-patient** must go untreated.

Exercise 3.1 — Christmas Trees (model)

Graph model. Let rows $R_1 \ldots R_n$ (capacity 2 each) and columns $C_1 \ldots C_m$ (capacity 1 each). For each empty cell (i,j) (no tree), add edge $R_i \to C_j$ of capacity 1. Add source $s \to R_i$ (cap 2) and $C_j \to t$ (cap 1).

Example figure. The provided 4×8 instance yields a graph with row caps (2,2,2,2), column caps all 1, and edges only for non-tree cells.

Certificate form. A placement is a flow of value k; its size is the total flow. A min cut proves optimality.

Exercise 3.2 — Christmas Trees (algorithm, runtime, correctness)

Algorithm (slides-first). Compute max flow in the bipartite network above.

```
Algorithm: place_tables
Input: n, m, forbidden ⊆ [n]×[m]
Output: maximum number of tables

build nodes s, {R_i}, {C_j}, t
add edges s→R_i (cap 2) and C_j→t (cap 1)
```

```
for each cell (i,j) not in forbidden: add R_i→C_j (cap 1)
run max_flow (EK or Dinic / Hopcroft–Karp via row-clones)
return |f|
// Time: O(E·√V) with Hopcroft–Karp; O(V·E²) with EK
```

Correctness.

- **Feasibility:** row/column caps enforce "≤2 per row, ≤1 per column"; unit edges enforce one table per chosen cell.
- **Optimality:** by max-flow/min-cut, any flow value equals the capacity of some cut; thus no placement can exceed |f|.

Runtime. $V \approx n+m+2$; $E \le$ (#empties)+n+m. With Hopcroft–Karp on a cloned-rows matching instance: **$O(E\sqrt{V})$**.

Example value. For the given grid the maximum is **7** (matches the figure's claim); a cut taking R_2's two units and the occupied columns certifies 7.

Pitfalls. Forgetting to remove edges for tree-cells; using greedy per row (can fail).

Variant drill. If each column allowed at most **2** tables, simply set $C_j \to t$ capacity = 2.

Transfer Pattern. Archetype: bipartite b-matching via flow.

Exercise 4 — KT 7.14 Escape

(a) Edge-disjoint escape routes.

- Build s* connecting to each x∈X with cap 1; keep original edges with cap 1; connect each safe node u∈S to t* with cap 1 (or ∞—cap 1 suffices if each safe can receive one route; use ∞ if no per-safe bound).
- Run max flow; **feasible iff |f|=|X|**; the unit s*→x edges ensure one route out of every populated node; edge caps enforce edge-disjointness.

(b) Node-disjoint version.

- **Split every vertex** v into v_in→v_out of capacity 1 (except safe nodes if unlimited, set cap = ∞); replace each original (u,v) by u_out→v_in (cap 1). Keep s*→x_in (cap 1) and s_out of x has cap 1 by the split.
- Run max flow; feasible iff |f|=|X|. Split-edges enforce at most one path through any vertex.

Why it works. Integrality gives |f| vertex/edge-disjoint paths; the converse is immediate by sending 1 along each path. **Node-splitting** is the standard reduction for node capacities.

Complexity. $O(E\sqrt{V})$ with Dinic / HK on unit capacities, or EK in $O(V \cdot E^2)$ is fine.

Variant drill. If each safe node can absorb at most c(u) evacuees, give u_in→u_out capacity c(u) instead of ∞.

Transfer Pattern. Archetype: disjoint paths via flow; cues: "routes do not share edges/nodes". Mapping: unit capacities and super-source/sink; node-split for node constraints.

✅ **Answer:** Build the corresponding unit-capacity flow (with node-splits for (b)); **yes** iff the max flow equals $|X|$.

Exercise 5 — CSES 1696 School Dance (I/O micro-card + method)

- **Inputs.** n boys, m girls, E allowed pairs. Output: maximum set of pairs; list the pairs.
- **Model.** Bipartite matching: s→boys (cap 1), allowed edges (cap 1), girls→t (cap 1).
- **Algorithm.** Hopcroft–Karp: $O(E\sqrt{(n+m)})$. Extract matching edges (boy→girl) from the flow.
- **Pitfalls.** Ensure 1-based indices in output as required by CSES; avoid printing unmatched nodes.

Exercise 6 — [*] Euler tours in mixed graphs

Goal. Decide if we can orient all undirected edges so the final directed graph has an **Euler tour**.

Key facts (directed Euler). A directed graph has an Euler tour iff every vertex has in(v)=out(v) and all non-isolated vertices lie in a single connected component when ignoring directions.

Balances. Let r(v)=in_dir(v)−out_dir(v) from the already-directed edges, and let udeg(v) be the number of incident undirected edges at v.

- If we orient an undirected edge u—v as u→v, then r(u)←r(u)−1 and r(v)←r(v)+1.
- Therefore each vertex must receive exactly **h(v)=(udeg(v)−r(v))/2** incoming orientations from its incident undirected edges. Feasible only if 0≤h(v)≤udeg(v) for all v and **udeg(v)≡r(v) (mod 2)**.

Reduction (hint realized). Build a bipartite graph **(E_u ↔ V)**: left nodes are undirected edges; right nodes are vertices; connect edge-node e={u,v} to u and v. Find a **b-matching** that matches each e exactly once and matches each vertex v exactly **h(v)** times. Implement via flow:

- s→e edges (cap 1) for each e in E_u.
- e→u and e→v edges (cap 1).
- vertex v→t edge (cap h(v)).

Feasible flow of value |E_u| ⇒ choose head of e at the matched endpoint; tail is the other endpoint. Together with the original directed edges, the resulting graph has in=out at each v; connectivity was assumed in the statement when orientations are ignored, hence an Euler tour exists.

Complexity. $O(E\sqrt{V})$ for the b-matching flow; linear to read off orientations.

Pitfalls. Forgetting parity check; demanding strong connectivity beyond what Euler needs; giving some v negative h(v).

Variant drill. If some undirected edges are **forced** to point a specific way, fix them first, update r(·) and udeg(·), then run the same flow.

Transfer Pattern. Archetype: circulation with vertex demands (exact balances) implemented as an edge-to-vertex b-matching.

✅**Answer: Yes** iff (i) for all v, $h(v)=(udeg(v)-r(v))/2$ is an integer in $[0,udeg(v)]$, and (ii) the b-matching flow succeeds (value $|E_u|$). The b-matching also **constructs** a valid orientation.

Puzzle — 99 Cops (<299 questions)

Task. Identify all corrupt cops. Majority are honest. Query form: "Ask X whether Y is corrupt."

Strategy.

1. **Find one honest cop (≤98 questions).** Boyer–Moore majority trick: keep a candidate c. For each other cop y, ask c about y.
2. If c says "y is corrupt", discard y.
3. If c says "y is honest", set c←y. After 98 questions the candidate must be honest (corrupts can only cancel in pairs; honest are a strict majority).
4. **Classify everyone (≤98 questions).** Ask the honest cop H about each of the other 98 cops and trust the answers.

Total ≤196 questions < 299.

Why it works. Pairwise cancellations cannot eliminate the strict majority of honest cops, so the survivor is honest; an honest witness then labels everyone correctly.

Summary

- **Patterns used:** EK & scaling; circulations with demands/lower bounds; b-matching; node-splitting; disjoint paths ↔ flow.
- **Certificates to remember:**
- Min cut (S,T) with crossing capacities = $|f|$.
- For blood: A-shortage consumes O-donor slack → 1 O-patient must remain.
- For trees: source row caps + column caps; value = #tables.
- For mixed Euler: parity & b-matching feasibility.
- **Notational blurb:** $r(v)$=in−out on directed part; $h(v)$ incoming heads required from undirected edges; δ bottleneck on an augmenting path; G_f residual graph; "clone" a row = split capacity into two unit nodes for HK.

Happy studying! 🚯