ADS2 — Hand-in (Flow/Matching) — Worked Solutions (REVISED)

Field	Value
Title	ADS2 — Hand-in (Flow/Matching) — Worked Solutions
Date	2025-10-19 (Europe/Copenhagen)
Author	Mads Richardt
Sources used	handind25.pdf (assumed §1.1–§1.2), algo2_week2.pdf, algo2_week3.pdf, algo2_week4.pdf, algo2_week5.pdf
Week plan filename	handind25.pdf (assumed week plan / hand-in)

General Methodology and Theory

- Pattern: reduce assignment/scheduling constraints to **integral max-flow** / (bi)partite **b-matching** with node-splitting for node capacities.
- Recipe: (1) identify decision makers and items; (2) build layers $s \to \text{agents} \to \text{items} \to t$ (and periods as needed); (3) encode limits with capacities; (4) run max-flow; (5) **accept** iff flow value equals the required total; read choices from saturated arcs.
- Correctness: with integer capacities, max-flow admits an **integral optimum**; min-cut gives tight bottleneck certificates.

Notes

- Per-agent total limit ⇒ capacity on $s \to \text{agent}$ (or split agent node).
- Item “at most once overall” ⇒ node-split $T_{\text{in}}(j) \to T_{\text{out}}(j)$ with capacity 1 .
- Per-period exact quota k across 3 days ⇒ arcs $s \to D_d$ each of capacity k and require total flow $= 3k$.

Coverage Table

Weekplan ID	Canonical ID	Title/Label (verbatim)	Assignment Source	Text Source	Status
1.1	—	Cake Meeting — maximize number of cakes *(label assumed; confirm exact phrasing)*	handind25.pdf §1.1 *(assumed)*	handind25.pdf §1.1 *(assumed)*	Solved

Weekplan ID	Canonical ID	Title/Label (verbatim)	Assignment Source	Text Source	Status
1.2	—	Cake Meeting — exactly k per day; baker/day caps *(label assumed; confirm exact phrasing)*	handind25.pdf §1.2 *(assumed)*	handind25.pdf §1.2 *(assumed)*	Solved

> **BLOCKERS (metadata only, solutions below are complete):** Need exact page/section numbers for §1.1–§1.2 in *handind25.pdf* to replace the “assumed” tags.

Solutions

Exercise 1.1 — (Cake Meeting) Maximize number of cakes

Concept mapping. Bipartite b -matching: left side = bakers (capacity 4 each), right side = cake types (capacity 1 each), edges = compatibility.

Method. Build a directed network:

- Source $s \to B_i$ with capacity 4 for each baker i .
- For each compatible pair (i, j) , add $B_i \to T_{\text{in}}(j)$ of capacity 1 .
- For each type j , add node-split $T_{\text{in}}(j) \to T_{\text{out}}(j)$ with capacity 1 and then $T_{\text{out}}(j) \to t$ with capacity 1 . Run max-flow. The value $|f^*|$ equals the number of cakes. The chosen pairs are saturated arcs $B_i \to T_{\text{in}}(j)$.

Pseudocode.

```
Algorithm: cake_max_cakes
Input: bakers B with caps (4 each); types T (cap 1 each via node split);
compat E ⊆ B×T
Output: value |f*| and set of (baker, type)

build nodes: s, t, B_i (∀i), T_in(j), T_out(j) (∀j)
add s→B_i (cap 4)
for (i,j) in E: add B_i→T_in(j) (cap 1)
for each j: add T_in(j)→T_out(j) (cap 1); add T_out(j)→t (cap 1)
run max_flow(s, t)
extract { (i,j) | B_i→T_in(j) saturated }
// Time: O(E · √V) typical with Dinic; Space: O(E)
```

Verification.

- Feasibility: each type carries at most 1 ; each baker uses at most 4 ; flow conserves at internal nodes.
- Optimality: any feasible assignment induces a flow of the same value; hence max-flow equals the optimum.

Pitfalls. Forgetting to split types (permits duplicates); modeling baker cap with parallel edges (bloats the graph).

Variant drill. Heterogeneous totals c_i : set $\text{cap}(s, B_i) = c_i$ (no other change).

Transfer Pattern (mapping guide).

- Archetype: bipartite **b-matching via flow**.
- Recognition cues: "at most once per item"; agent upper bounds; compatibility subset.
- Mapping steps: agents $\to$ left nodes with cap; items $\to$ right nodes with node cap 1 (split); compat $\to$ edges of cap 1 ; run max-flow.
- Certificate: list saturated $B_i \to T_{\text{in}}(j)$ arcs; min-cut identifies bottlenecks.
- Anti-cues: unlimited copies of types (not matching-like).

✅**Answer:** Build the flow as above; the maximum number of cakes equals $|f^*|$.

Concept mapping. Three-day layered flow with per-day quotas and baker day-caps while enforcing unique types overall.

Network.

- Days: create D_1, D_2, D_3 with $s \to D_d$ capacity k for each day $d \in \{1, 2, 3\}$.
- Type once: for each type j , split $T_{\text{in}}(j) \to T_{\text{out}}(j)$ with capacity 1 .
- Day chooses a type: add $D_d \to T_{\text{in}}(j)$ capacity 1 for all (d, j) .
- Baker per-day cap: for each baker i and day d , add node $BD(i, d)$ and arc $BD(i, d) \to B(i)$ with capacity 2 .
- Compatibility: for each compatible (i, j) and any day d , add $T_{\text{out}}(j) \to BD(i, d)$ with capacity 1 .
- Baker total: add $B(i) \to t$ with capacity 4 .

Run max-flow; **accept** iff the value equals $3k$. Extract triples from saturated paths $s \to D_d \to T_{\text{in}}(j) \to T_{\text{out}}(j) \to BD(i, d) \to B(i) \to t$.

Pseudocode.

```
Algorithm: cake_schedule_3days
Input: bakers B; types T; compat E; integer k
Output: assignment of exactly k per day, or infeasible with a cut

build D_1..D_3; add s→D_d (cap k)
for each j: add T_in(j), T_out(j); add T_in→T_out (cap 1)
for each d,j: add D_d→T_in(j) (cap 1)
for each i,d: add BD(i,d); add BD(i,d)→B(i) (cap 2)
for each i: add B(i)→t (cap 4)
for each (i,j) in E, for each d: add T_out(j)→BD(i,d) (cap 1)
run max_flow(s, t)
if |f*| = 3k: return triples (d, j, i) from saturated paths else return
```

```
infeasible + min-cut
// Time: O(E · √V) typical; Space: O(E)
```

Why constraints hold.

- Exactly k per day: total demanded flow is $3k$, and only arcs $s \to D_d$ can supply it, each bounded by k , so each day is saturated to k .
- Type ≤ 1 : the split edge has capacity 1 .
- Baker ≤ 2 per day: enforced by $BD(i,d) \to B(i)$ capacity.
- Baker ≤ 4 total: enforced by $B(i) \to t$ capacity.
- Compatibility: only allowed arcs $T_{\text{out}}(j) \to BD(i,d)$ exist.

Pitfalls. Omitting $D_d \to T_{\text{in}}(j)$ cap 1 (could allow multiple picks of the same type per day in some variants); forgetting baker-day nodes (can't enforce ≤ 2 /day).

Variant drill. Unequal day quotas k_d : set $\text{cap}(s, D_d) = k_d$ and accept iff $|f^*| = \sum_d k_d$.

Transfer Pattern (mapping guide).

- Archetype: **layered flow with per-period quotas and agent caps**.
- Recognition cues: "exactly k per period", "agent daily cap plus total cap", "unique items overall".
- Mapping steps: periods $\to$ day nodes with exact-flow caps; items $\to$ node-split of cap 1 ; agents $\to$ per-day nodes feeding a total-cap node; compat $\to$ arcs from items to agent-day nodes.
- Certificate: flow value $= 3k$ and the list of picked triples (d, j, i) ; min-cut shows bottlenecks when infeasible.
- Anti-cues: if items may repeat across days, drop the global split and cap per day instead.

✅ **Answer:** Build the layered network; accept iff $|f^*| = 3k$; read off the schedule from saturated paths.

Puzzle

Hall-style glimpse. Suppose every baker offers at least 4 distinct types and every type is offered by at least one baker. Claim: constraints (1)–(3) alone always allow at least $\min\{t, 4b\}$ cakes. Decide true/false and justify via a Hall-type condition for b -matchings; if false, give the smallest counterexample.

Summary

- 1.1: bipartite b -matching via max-flow; types split to enforce uniqueness; value $|f^*|$ is the maximum number of cakes.
- 1.2: add day layer with quotas, baker-day nodes (cap 2) and baker total (cap 4); accept iff $|f^*| = 3k$; extract triples.
- Certificates: min-cut for infeasibility; saturated arcs as witnesses for feasibility.
- Transfer: same blueprints apply to staffing shifts, course-project assignments, and sports scheduling with daily quotas.

1 Cake Meeting (from the exam E18) You're organizing the First Annual DTU Cake Meeting, to be held on three days in January. There will be lots of cake for everyone and a large number of bakers have applied to bake cakes for the event. You need to hire bakers according to the following constraints.

1. Each candidate baker has given you a list of types of cakes they can bake.
2. Each baker can bake at most four cakes during the entire event.
3. At most one of each type of cake (chocolate cake, strawberry cake, ...) can be produced by all the bakers in total.

1.1 Suppose there are b candidate bakers and t different types of cake. Give an algorithm that computes the maximum number of cakes that can be produced according to the constraints. Analyze the asymptotic running time of your algorithm. Remember to argue that your algorithm is correct.

1.2 It turns out that there were way too many cakes and not enough variation in style on each day. So you impose the following new constraints (in addition to constraint 1, 2, and 3 from above).

4. There must be produced exactly k cakes each day, and thus $3k$ cakes altogether.
5. Each baker can bake at most two cakes each day (and still at most 4 cakes in total).

Give an efficient algorithm that either assigns a baker and a cake to each of the $3k$ cake slots, or correctly reports that no such assignment is possible. Analyze the asymptotic running time of your algorithm. Remember to argue that your algorithm is correct.