ADS2 — DP2 Exam Notes & Worked Solutions

Meta	Value
Title	ADS2 — Dynamic Programming II (Knapsack & Sequence Alignment)
Date	2025-10-18
Author	Generated from **system_prompt** + provided PDFs/images
Sources used	Weekplan **weekplan.pdf** (pp.1–2); Slides **DP2-4x1.pdf** (Knapsack & Sequence Alignment); Kleinberg–Tardos **§6.4** (pp.266–272) and **§6.6** (pp.278–284); Exercise sheet **exercise_6_8.pdf** (KT 6.8); images: *Pasted image.png*, *Pasted image (2).png*, *Pasted image (3).png*
Week plan filename	weekplan.pdf

General Methodology and Theory

- **DP recipe**: define subproblem → recurrence → base cases → evaluation order → table fill → witness recovery.
- **0/1 Knapsack** (slides-first): subproblem $$OPT(i,w)$$ = best value using first $$i$$ items within capacity $$w$$. Recurrence (slides): if $$w_i\le w$$ then $$OPT(i,w)=\max(OPT(i-1,w),\ v_i+OPT(i-1, w-w_i))$$; else $$OPT(i,w)=OPT(i-1,w)$$. Time/space $$O(nW)$$ (pseudo-polynomial).
- **Global sequence alignment** (Needleman–Wunsch, linear gap $$\delta$$): $$D(i,j)=\min{\alpha(x_i,y_j)+D(i-1,j-1),\ \delta+D(i-1,j),\ \delta+D(i,j-1)}$$ with $$D(i,0)=i\delta,\ D(0,j)=j\delta$$. Time/space $$O(mn)$$.
- **Design hygiene**: verify with small tables; state tie-breaks for determinism; trace back to produce a certificate.

Notes

- Slides are primary; textbook variants noted under *Alternative Approach* per exercise.
- Enumeration **must** follow the week plan; images supplement statements only.
- Keep tables concise: include DP values or tracebacks only where it clarifies the witness.

Coverage Table

Weekplan ID	Canonical ID	Title/Label (verbatim)	Assignment Source	Text Source	Status
1	KT §6.4 (Knapsack)	**[w] Knapsack** (items (5,7), (2,6), (3,3), (2,1), W=6)	weekplan.pdf p.1	weekplan.pdf p. 1; slides DP2-4x1 (Knapsack); KT §6.4	Solved
2	KT §6.6 (Alignment)	**[w] Sequence alignment** (APPLE vs PAPE, gap $$ \delta=2$$, matrix on {A,E,L,P})	weekplan.pdf p.1	weekplan.pdf p. 1; slides DP2-4x1 (Alignment); KT §6.6	Solved
3.1	CSES 1158	**Book Shop 3.1** — compute max pages	weekplan.pdf p.2	slides DP2-4x1 (Knapsack pattern)	Solved
3.2	CSES 1158	**Book Shop 3.2** — $$O(x)$$ space	weekplan.pdf p.2	slides DP2-4x1 (1-row DP)	Solved
3.3	CSES 1158	**Book Shop 3.3** — implement (site ref)	weekplan.pdf p.2	problem statement on CSES; pattern here	Solved (pseudocode)
4	—	**Longest palindrome subsequence**	weekplan.pdf p.2	standard LPS DP (course canon)	Solved
5	KT 6.8	**Defending Zion** (EMP scheduling)	weekplan.pdf p.2	exercise_6_8.pdf	Solved

---

Solutions

Exercise 1 — KT §6.4 — Knapsack

Assignment Source: weekplan.pdf p.1. **Text Source**: weekplan.pdf p.1; slides DP2-4x1; KT §6.4.

Instance: items $$i_1!(w{=}5,v{=}7),\ i_2!(2,6),\ i_3!(3,3),\ i_4!(2,1);\ W{=}6$$.

Recurrence (slides): as in *General Methodology*. Base row/col 0.

DP table (values) $$i\times w$$:

i\w	0	1	2	3	4	5	6
0	0	0	0	0	0	0	0
1	0	0	0	0	0	7	7
2	0	0	6	6	6	7	7
3	0	0	6	6	6	9	9
4	0	0	6	6	7	9	9

Witness (traceback): from $(4,6)$ → take i_3 (to $(3,3)$) → take i_2 (to $(2,1)$) → stop. **Picked** $\{i_2,i_3\}$, weight 5, value 9.

Pitfalls: mixing 0/1 with unbounded; forgetting base row/col.

Variant Drill: If $W=7$, best is $\{i_1,i_2\}$ with value 13.

Alternative Approach (KT): identical recurrence; value/weight emphasis differs.

Transfer Pattern: 0/1 knapsack. *Cues*: capacity W, each item at most once, additive value. *Mapping*: nouns → (w_i,v_i), limit → W. *Certificate*: subset with total weight ≤ W achieving $OPT(n,W)$. *Anti-cues*: fractional or unlimited copies.

Pseudocode

```
Algorithm: knapsack_01
Input: n, arrays w[1..n], v[1..n], capacity W
Output: best value and one witness via keep[][]
for c=0..W: dp[0,c]←0
for i=1..n:
  for c=0..W:
    dp[i,c]←dp[i-1,c]
    if w[i]≤c and dp[i-1,c-w[i]]+v[i] > dp[i,c]:
      dp[i,c]←dp[i-1,c-w[i]]+v[i]; keep[i,c]←true
// traceback from (n,W)
// Time: O(nW); Space: O(nW)
```

✅**Answer**: $OPT=9$ with items $\{(2,6),(3,3)\}$.

Exercise 2 — KT §6.6 — Sequence Alignment

Assignment Source: weekplan.pdf p.1. **Text Source**: weekplan.pdf p.1; slides DP2-4x1; KT §6.6.

Instance: $X=\text{APPLE}$ (columns), $Y=\text{PAPE}$ (rows). Gap $\delta=2$. Penalty matrix over $\{A,E,L,P\}$ with 0 on diagonal and off-diagonals as in plan.

Recurrence: as in *General Methodology*. Init $D(i,0)=2i, D(0,j)=2j$. Tie-break: diag < up < left.

DP values (compact):

i\j	0	1	2	3	4	5
0	0	2	4	6	8	10
1	2	1	2	4	6	8
2	4	2	2	3	5	7
3	6	4	2	2	4	6
4	8	6	4	3	4	**4**

Traceback (one optimum):

```
X: A P P L E
Y: P A P - E
```

Cost: $1+1+0+2+0=4$.

Pitfalls: swapping X/Y; forgetting linear gap init.

Variant Drill: Lower δ to 1 → cheaper gaps; recompute to expect cost ≤ 3.

Alternative Approach: shortest path on grid graph with diagonal weights $\alpha(\cdot,\cdot)$ and horizontal/vertical δ.

Transfer Pattern: global alignment (linear gap). *Cues*: substitution matrix + uniform gap. *Mapping*: chars → nodes; operations → DP moves. *Certificate*: paired strings with per-column costs summing to $D(n,m)$. *Anti-cues*: local alignment (resets to 0); affine gaps (three matrices).

Pseudocode

```
Algorithm: needleman_wunsch_linear
Input: strings X[1..m], Y[1..n], penalties α(·,·), gap δ
Output: cost D[n,m] and an alignment via backpointers
for j=0..m: D[0,j]←j·δ
for i=0..n: D[i,0]←i·δ
for i=1..n:
  for j=1..m:
    D[i,j]←min{ α(Y[i],X[j])+D[i-1,j-1], δ+D[i-1,j], δ+D[i,j-1] }
// traceback from (n,m)
// Time: O(mn); Space: O(mn)
```

✅**Answer**: minimum cost **4** with alignment shown.

Exercise 3.1 — CSES 1158 — Book Shop (max pages)

Assignment Source: weekplan.pdf p.2. **Text Source**: slides DP2-4x1 (knapsack pattern); CSES statement.

Mapping: price h_i → weight; pages s_i → value; budget x → capacity. 0/1 knapsack.

Recurrence: $DP(i,c)=\max(DP(i-1,c),\ s_i+DP(i-1,c-h_i))$ for $h_i\le c$, else $DP(i,c)=DP(i-1,c)$. Base row 0.

Pseudocode

```
Algorithm: book_shop_max_pages
Input: n, price h[1..n], pages s[1..n], budget x
Output: maximum pages
for c=0..x: dp[0,c]←0
for i=1..n:
  for c=0..x:
    dp[i,c]←dp[i-1,c]
    if h[i]≤c and dp[i-1,c-h[i]]+s[i] > dp[i,c]:
      dp[i,c]←dp[i-1,c-h[i]]+s[i]
return dp[n,x]
// Time: O(nx); Space: O(nx)
```

Pitfalls: confusing price vs pages; exceeding budget.

✅**Answer**: Reduces exactly to 0/1 knapsack; table yields optimum pages within budget.

Exercise 3.2 — CSES 1158 — Book Shop in $O(x)$ space

Idea: 1-row DP scanning capacities **descending** to preserve 0/1 semantics.

Pseudocode

```
Algorithm: book_shop_one_row
Input: n, h[1..n], s[1..n], budget x
Output: maximum pages
for c=0..x: dp[c]←0
for i=1..n:
  for c=x down to h[i]:
    dp[c]←max(dp[c], s[i]+dp[c-h[i]])
return dp[x]
// Time: O(nx); Space: O(x)
```

Why descending? Prevents reusing the same book multiple times.

✅**Answer**: Achieves $$O(x)$$ space with identical optimal value.

Exercise 3.3 — CSES 1158 — Implementation note

I/O micro-card: read $$n, x$$; arrays $$h[1..n], s[1..n]$$; print $$\text{book_shop_one_row}(...)$$ result.

Testing tips: include edge cases $$x{=}0$$, a single book equal to $$x$$, and many books with identical prices.

✅**Answer**: Use the one-row DP above; outputs the maximum total pages.

Exercise 4 — Longest Palindrome Subsequence (LPS)

Assignment Source: weekplan.pdf p.2.

Subproblem: $$L(i,j)$$ = length of LPS in substring $$s[i..j]$$.

Recurrence: - If $$i>j$$: 0; if $$i=j$$: 1. - If $$s[i]=s[j]$$: $$L(i,j)=2+L(i+1,j-1)$$. - Else: $$L(i,j)=\max(L(i+1,j), L(i,j-1))$$.

Pseudocode

```
Algorithm: lps_length
Input: string s[1..n]
Output: length of an LPS
for i=1..n: dp[i,i]←1
for len=2..n:
  for i=1..n-len+1:
    j←i+len-1
    if s[i]=s[j]: dp[i,j]←2 + (dp[i+1,j-1] if i+1≤j-1 else 0)
    else: dp[i,j]←max(dp[i+1,j], dp[i,j-1])
return dp[1,n]
// Time: O(n^2); Space: O(n^2) → can be reduced to O(n) for length only on
diagonals
```

Correctness sketch: last pair either matches (use both ends) or not (drop one end). Standard interval-DP.

Witness: keep a parent direction to reconstruct one palindrome.

✅**Answer**: Recurrence and algorithm as above; length in $$O(n^2)$$.

Exercise 5 — KT 6.8 — Defending Zion (EMP scheduling)

Assignment Source: weekplan.pdf p.2. **Text Source**: exercise_6_8.pdf.

Model: arrivals $x_1..x_n$; recharge function $f(j)$ (power after j seconds since last use). Using at time t after j seconds since previous use destroys $\min(x_t, f(j))$.

Counterexample to greedy (part a): Let $x=[0,10,10,0]$ and $f=[\_,1,3,8]$ (i.e., $f(1){=}1,f(2){=}3,f(3){=}8$). Greedy "fire at $t{=}4$ with smallest j s.t. $f(j)\ge x_4$" fires at $t{=}4$ with $j{=}1$ (kills 0), then recurses on $[0,10,10]$, missing the optimal plan: fire at $t{=}3$ (kills 8) and **don't** fire at 4 → total 8.

DP (part b) — *slides-style formulation over last-fire time*: - Let $G[t]$ = best robots destroyed up to time t **with last activation exactly at t**. - Let $F[t]$ = best up to t with last activation at $\le t$ (overall optimum prefix). - Base: $G[0]=0,\ F[0]=0$. - Transition for $t\ge1$: $G[t]=\max_{0\le k < t}\big(G[k] + \min(x_t,\ f(t-k)) \big)$ (previous last fire at time k; if $k=0$, it's the first fire with $j=t$). Then $F[t]=\max(F[t-1],\ G[t])$.

Pseudocode

```
Algorithm: emp_max_destroyed
Input: n, arrivals x[1..n], recharge f[1..n]
Output: max robots destroyed
G[0]←0; F[0]←0
for t=1..n:
  best←0
  for k=0..t-1:
    j←t-k
    best←max(best, G[k] + min(x[t], f[j]))
  G[t]←best
  F[t]←max(F[t-1], G[t])
return F[n]
// Time: O(n^2); Space: O(n)
```

Correctness: last activation time partitions the schedule; the gap length j is determined; subproblems do not overlap in time.

Variant Drill: if f is nondecreasing and concave, consider Knuth-/divide-&-conquer-style optimizations; otherwise stick to $O(n^2)$.

Transfer Pattern: segmented scheduling with state = time since last use. *Cues*: recharge curve, "since last" wording. *Mapping*: choose activation times; reward per activation depends on gap. *Certificate*: list of activation timestamps and per-use kills summing to optimum.

✅**Answer**: DP above returns the maximum robots destroyed; greedy fails.

Puzzle

Blind Man's Deck (10 face-up among 52): take **any 10 cards** as pile A; let the rest be pile B. Flip pile A. Now both piles have the **same** number of face-up cards. *Reason*: if pile A initially had k face-up, B had $10-k$; flipping A makes exactly $10-k$ face-up in A.

Summary

- **Knapsack**: 0/1 DP with $OPT(i,w)$; pseudo-polynomial $O(nW)$; 1-row $O(W)$ space when only value is needed.
- **Alignment**: Needleman–Wunsch with linear gaps; initialize borders with cumulative gaps; traceback yields a certified alignment.
- **Book Shop**: direct 0/1 mapping (price→weight, pages→value); descending 1-row DP avoids reuse.
- **LPS**: interval DP over substrings; elegant two-case recurrence.
- **Zion (KT 6.8)**: schedule by DP on last-fire time; greedy counterexample; $O(n^2)$ time, $O(n)$ space.
- **Notation**: $W, v_i, w_i,\ \delta,\ \alpha(\cdot,\cdot),\ D(i,j),\ L(i,j)$. Always show a witness (subset, alignment, or schedule) to certify optimality.