1 Exercise 1

1.

Character	Frequency	Codeword
A	6	110
B	4	1110
C	9	10
D	4	1111
E	19	0

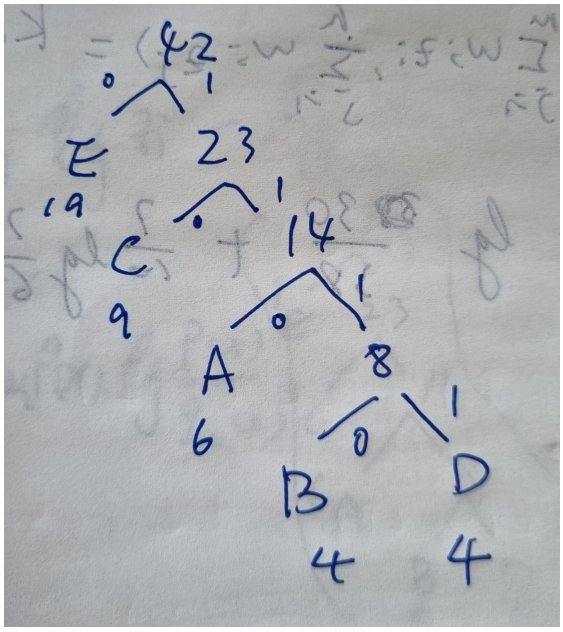


Figure 1: Hoffman coding

Thus, 011010101100 represents EACCAE.

2
2.1 20, 70, 30, 0
2.2 4, 1, 1, N
2.2 20, 70, 30, 0
2.4 4, 1, 1, N

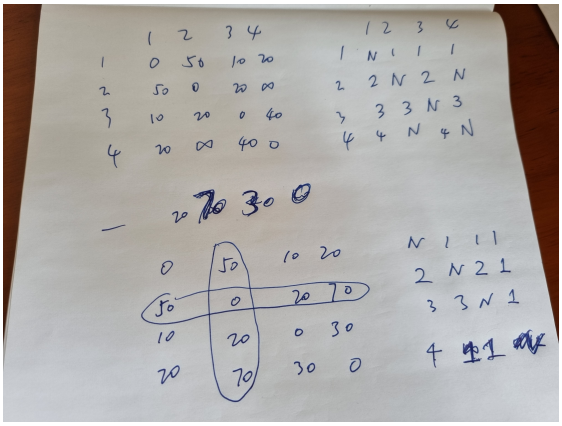


Figure 2: Floyd-Warshall

3

3.1

☐ **a** Linear time: $O(n)$. ☐ **b** Quadratic time: $O(n^2)$.
☐ **c** Logarithmic time: $O(\log n)$. ☒ **d)** Exponential time: $O(a^n)$.

3.2

☐ **a** Linear time: $O(n)$. ☐ **b** Quadratic time: $O(n^2)$.
☐ **c** Logarithmic time: $O(\log n)$. ☒ **d)** Exponential time: $O(a^n)$.

3.3

☒ **a)** Linear time: $O(n)$. ☐ **b** Quadratic time: $O(n^2)$.
☐ **c** Logarithmic time: $O(\log n)$. ☐ **c** Exponential time: $O(a^n)$.

4.

4.1 *PUSH*: 1, *POP*: 1, *EXPAND*: n

4.2

Operation	Actual cost	Amorized cost
$PUSH(S, x)$	1	3
$POP(S)$	1	0
$EXPAND(S)$	n	0

4.3

- Each stack operations takes at most cost 3.

- A sequence of n stack operations is at most $3n$, thus the time complexity of a sequence of n stack operations is $O(n)$.

Exercise 2, The S-Part, 50 points in total

1. Satisfiability of Propositional Logic (*15 points*)

1.1 Consider the following pairs of clauses. Is it possible to apply the resolution rule? If yes, indicate what would be the resulting clause (if there is more than one option, you may choose any of them).

		Yes	No	Result (if yes)
$\{A, \neg B\}$	$\{A, \neg B, C\}$	☐	☒	
$\{A, \neg B\}$	$\{A, B, C\}$	☒	☐	$\{A, C\}$
$\{\neg A, \neg B\}$	$\{A, B, C\}$	☒	☐	$\{\neg A, A, C\}$ or $\{\neg B, B, C\}$ (they simplify to true)
$\{A, C\}$	$\{A, \neg C\}$	☒	☐	$\{A\}$
$\{A, B, C\}$	$\{A, B, \neg D\}$	☐	☒	

1.2 Consider the set of clauses:

$$\Delta = \{\{A, \neg B, C\}, \{A, B, \neg C\}, \{\neg A, \neg B\}, \{B, C\}, \{\neg A, \neg C\}\}$$

Use DPLL to determine whether Δ is satisfiable or unsatisfiable. Give a complete trace of the algorithm, showing the simplified formula for each recursive call of the DPLL function. If you need to choose a literal to assign a value to, always choose the one alphabetically first (i.e. A before B before C, etc.) in its non-negated form (so K rather than $\neg K$), and assume that the splitting rule first attempts the value True (T) and then the value False (F). **Draw the trace of the algorithm on a separate piece of paper.**

Is Δ satisfiable? ☒ **Yes** ☐ **No** If Yes, provide the satisfying assignment found by DPLL:

A=F, B=T, C=T

1.3 Would Clause Learning help in this example? ☐ **Yes** ☒ **No**
Justify your answer by explaining how it would help (if yes), or arguing why it does not help (if no).

No, clause learning would learn the clause $\{\neg A\}$, but the algorithm backtracks only once either way.

Solution for 1.2:

$$\Delta = \{\{A, \neg B, C\}, \{A, B, \neg C\}, \{\neg A, \neg B\}, \{B, C\}, \{\neg A, \neg C\}\}$$

Splitting Rule $A \mapsto \top$:

$$\Delta = \{\{\neg B\}, \{B, C\}, \{\neg C\}\}$$

UP Rule $B \mapsto \bot$

$$\Delta = \{\{C\}, \{\neg C\}\}$$

UP Rule $C \mapsto \top$
Unsat, backtrack

Splitting Rule $A \mapsto \bot$:

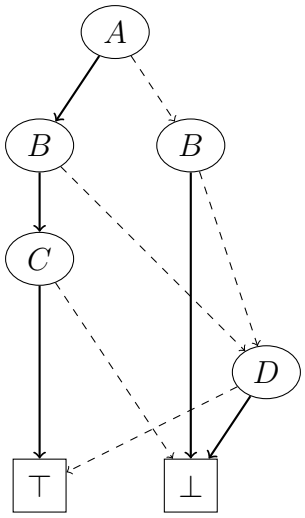
$$\Delta = \{\{\neg B, C\}, \{B, \neg C\}, \{B, C\}\}$$

Splitting Rule $B \mapsto \top$:

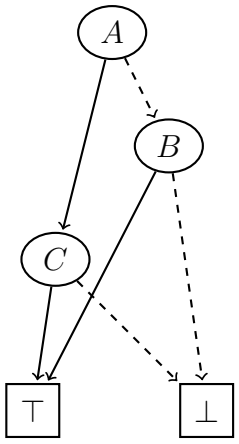
$$\Delta = \{\{C\}\}$$

UP Rule $C \mapsto \top$
SAT

2 Binary Decision Diagrams (*12 points*)
2.1 Using variable ordering $\langle A, B, C, D\rangle$ draw a reduced ordered BDD that represents the following function $(A \wedge B \wedge C) \vee (\neg B \wedge \neg D)$.



2.2 Considering the following BDD:



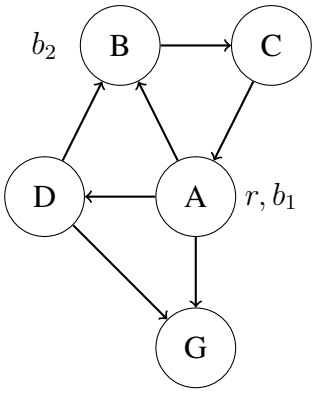
How many satisfying assignments does the BDD represent?

4 (8 is also correct if the assignment below mentions D)

Provide an example of a satisfying assignment represented by the BDD:

A=T, B=F, C=T

3 Planning (*13 Points*)
Consider the following planning task. It is about a storage facility with five rooms (A, B, C, D, and G), a robot r and two boxes (b_1, b_2). Initially, the robot and b_1 are in room A and b_2 is in room B. The goal is to deliver both boxes to room G. The layout of the storage facility is depicted in the following figure:



The robot can *move* between the rooms; note that all connections are unidirectional, i.e., the robot can take each connection only in the indicated direction. Additionally, the robot can *grab* a box if the robot and the box are in the same room (in parts (a) and (b) the robot can carry any number of boxes at the same time). If the robot is holding a box, then it can *drop* the box in its current location. The task is formalized in STRIPS as follows.

- The facts are $P = \{at(r, y) \mid y \in \{A, B, C, D, G\}\} \cup \{at(x, y) \mid x \in \{b_1, b_2\}$ and $y \in \{A, B, C, D, G, r\}\}$
- The initial state is $I = \{at(r, A), at(b_1, A), at(b_2, B)\}$
- The goal is $G = \{at(b_1, G), at(b_2, G)\}$

There are three types of actions (specified as their precondition, add and delete list), all having a cost of 1.

1. $move(x, y) : (\{at(r, x)\}, \{at(r, y)\}, \{at(r, x)\})$ for all $(x, y) \in \{A, B, C, D, G\}$ such that $x \to y$ in the figure above.
2. $grab(x, y) : (\{at(r, y), at(x, y)\}, \{at(x, r)\}, \{at(x, y)\})$ for all $x \in \{b_1, b_2\}$ and $y \in \{A, B, C, D, G\}$
3. $drop(x, y) : (\{at(r, y), at(x, r)\}, \{at(x, y)\}, \{at(x, r)\})$ for all $x \in \{b_1, b_2\}$ and $y \in \{A, B, C, D, G\}$

3.1 Give a shortest plan for the initial state in case that one exists.

grab(b_1, A), move(A, B), grab(b_2, B), move(B, C), move(C, A), move(A, G), drop(b_1, G), drop(b_2, G)

3.2 Give a shortest delete-relaxed plan for the initial state in case that one exists.

grab(b_1, A), move(A, B), grab(b_2, B), move(A, Gx), drop(b_1, G), drop(b_2, G)

3.3 What is the value of $h^*(I)$? 8

3.4 What is the value of $h^+(I)$? 6

3.5 List the successor states after expanding the initial state, and for each of them indicate what is the value of h^+.

- $\{at(r, B), at(b_1, A), at(b_2, B)\}$, $h^+ = 7$
- $\{at(r, G), at(b_1, A), at(b_2, B)\}$, $h^+ = \infty$
- $\{at(r, D), at(b_1, A), at(b_2, B)\}$, $h^+ = 8$
- $\{at(r, A), at(b_1, R), at(b_2, B)\}$, $h^+ = 5$

4 (*10 points*) We need to generate a schedule of three courses $C_1, \ldots, C_3$, among three days of the week: $\{M, T, W\}$ subject to the following constraints:

1. Each course has one lecture per week.
2. Courses C_1 and C_2 cannot be on the same day of the week.
3. Course C_2 should be on an earlier day of the week than course C_3. (M comes before T, which comes before W).
4. Courses C_1 and C_3 should be on the same day.

Encode this problem as a SAT formula, and answer the following questions. Note: you may use predicate logic notation to compactly represent sets of propositions and/or conjunctions.

4.1 What propositions are you using?

$p_{i,d}$ for $i \in \{1, 2, 3\}$, and $d \in \{M, T, W\}$ representing whether course C_i is scheduled on day d or not.

4.2 What is the formula? Indicate which parts correspond to each of the constraints above.

$$\bigwedge_{i \in \{1,2,3\}} (p_{i,M} \vee p_{i,T} \vee p_{i,W}) \wedge \neg(p_{i,M} \wedge p_{i,T}) \wedge \neg(p_{i,M} \wedge p_{i,W}) \wedge \neg(p_{i,T} \wedge p_{i,W})$$

$$\wedge$$

$$\bigwedge_{d \in \{M,T,W\}} \neg(p_{1,d} \wedge p_{2,d})$$

$$\wedge$$

$$\neg p_{3,M} \wedge \neg p_{2,W} \wedge p_{3,T} \implies p_{2,M}$$

$$\wedge$$

$$\bigvee_{d \in \{M,T,W\}} (p_{1,d} \wedge p_{3,d})$$

4.3 Explain how testing satisfiability of the formula can provide the solution.

By the first part of the formula, in each satisfying assignment exactly one $p_{c,d}$ proposition has value true for each course c. We should assign each course c to the day d where $p_{c,d}$ has value true.