Algorithms and Satisfiability

(DAT6)

Exam Assignments

10.00 - 13.00, 3 June 2022

Full name:	
Student number:	
E-mail at student.aau.dk:	

This exam consists of two parts. Exercise 1 is for Algorithms (the A-Part). Exercise 2 is for Satisfiability (the S-Part).

Put your name on this sheet, and on the header of Exercise 2. Most exercises can be answered directly on the exam sheet. If you need any additional sheets of paper, remember to put your name and your student number on all of them.

During the exam you are allowed to consult books, notes, and other written materials. However, the use of any kind of electronic devices, e.g., laptops, tablets, and mobile phones, is ***NOT*** permitted.

Exercise 1, The A-Part, 50 points in total

1. We would like to use Huffman coding to encode the following sequence of text.

$$AABBBBAAAACCCCCCCCCEEEEEEDDDDEEEEEEEEEEEE$$

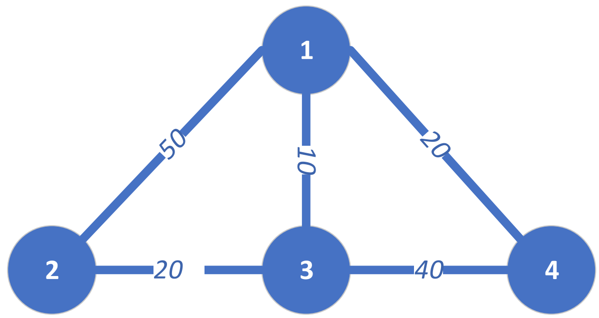
1.1 (*7 points*) Please draw the binary Huffman coding tree based on the above text.

1.2 (*6 points*) Please decode the following Huffman code into text

$$011010101100$$

Please write down your answer here ___________________________.

2. Given a weighted graph G with 4 nodes as shown in the following figure, we run Floyd-Warshall algorithm on G.



After the first iteration ($k = 1$) of the Floyd-Warshall algorithm:

2.1 (*4 points*) In the **distance matrix**, what does the 4-th row (representing node 4) look like? Please write down your answer here ____________________________.

2.2 (*4 points*) In the **predecessor matrix**, what does the 4-th row (representing node 4) look like? Please write down your answer here ____________________________.

After the second iteration ($k = 2$) of the Floyd-Warshall algorithm:

2.3 (*3 points*) In the **distance matrix**, what does the 4-th row (representing node 4) look like? Please write down your answer here ____________________________.

2.4 (*3 points*) In the **predecessor matrix**, what does the 4-th row (representing node 4) look like? Please write down your answer here ____________________________.

3. We want to compare the time complexity between a trivial Fibonacci algorithm (FIB) and a parallel Fibonacci algorithm (P-FIB) as follows.

```
procedure FIB(n)
    if n ≤ 1 then
        return n
    else
        x = FIB(n − 1)
        y = FIB(n − 2)
        return x + y
```

```
procedure P-FIB(n)
    if n ≤ 1 then
        return n
    else
        spawn x = P-FIB(n − 1)
        y = P-FIB(n − 2)
        sync
        return x + y
```

3.1 (*2 points*) What is the type of time complexity of the FIB algorithm? (Please only choose the lowest correct complexity).

☐ **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 (*3 points*) What is the type of time complexity of the **work** (T_1) of P-FIB algorithm? (Please only choose the lowest correct complexity).

☐ **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 (*3 points*) What is the type of time complexity of the **span** (T_∞) of P-FIB algorithm? (Please only choose the lowest correct complexity).

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

4. Consider a stack S that is implemented by an array with three operations:

- *PUSH*(S, x), pushes object x onto stack S.
- *POP*(S), pops the top of stack S and returns the popped object.
- *EXPAND*(S), if the stack is full, the *EXPAND* operation copies n elements from the current stack to a new stack of double size; if the stack is not full, the *EXPAND* operation does nothing.

4.1 (*3 points*) What is the worst case asymptotic cost of a *PUSH*, a *POP*, and an *EXPAND* operation, respectively?
PUSH: ____________________________(*1 point*)
POP: ____________________________(*1 point*)
EXPAND: ____________________________(*1 point*)

4.2 (*9 points*) Please use the **accounting method** to conduct amortized analysis. Specifically, please fill in the following table.

Operation	Actual cost	Amortized cost
PUSH(S, x)	_______(*1 point*)	_______(*2 points*)
POP(S)	_______(*1 point*)	_______(*2 points*)
EXPAND(S)	_______(*1 point*)	_______(*2 points*)

4.3 (*3 points*) From the **amortized costs** you obtained in the above table, please fill in the following statements.

- Each stack operation takes at most cost ________(*1 point*).
- A sequence of n stack operations is at most ________(*1 point*), thus the time complexity of a sequence of n stack operations is ________(*1 point*).

Name: **Student number:**

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)
$\{A, \neg B\}$	$\{A, \neg B, C\}$	☐	☐	
$\{A, \neg B\}$	$\{A, B, C\}$	☐	☐	
$\{\neg A, \neg B\}$	$\{A, B, C\}$	☐	☐	
$\{A, C\}$	$\{A, \neg C\}$	☐	☐	
$\{A, B, C\}$	$\{A, B, \neg D\}$	☐	☐	

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:

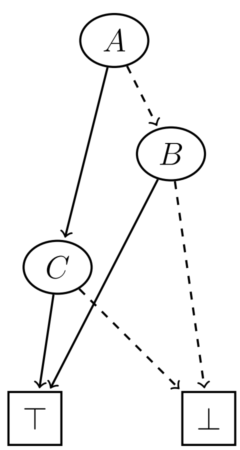
____

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).

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:



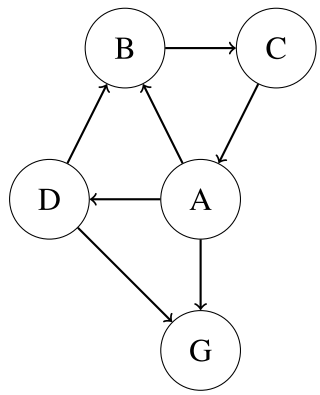
How many satisfying assignments does the BDD represent?

____

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

____

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.

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

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

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

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

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?

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

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