Mock Exam "Algorithms and Computability" F24

Question 1 10 Points

Construct a 2-tape DTM accepting the language $\{a^n b^n a^{2n} \mid n \in \mathbb{N} \setminus \{0\}\}$ over the alphabet $\{a, b\}$. To this end, first explain informally how the machine processes an input and then define all components of the machine formally.

Hint: You can use the "stay"-direction.

Solution

Intuitively, the machine scans the first tape (containing the input) from left-to-right in three parts:

- It copies the initial prefix of a's (say there are n many) to the second tape (marking the first one to be able to find it again later). If the input does not start with a, the machine rejects. This is implemented using states s and q_1.
- Then it compares the number of b's following the a's by moving the head on the second tape left. If there are not n many b's, then the machine rejects. This is implemented using state q_2.
- Now, the number of a's at the end of the input is compared by similarly moving the head right and left on the second tape. Again, if it is not equal to $2n$ the machine rejects. This is implemented using states q_3 (going right) and q_4 (going left).

Finally, state q_5 is used to check that afterwards, no more letters are on the first tape.

Formally, we define $M = (Q, \Sigma, \Gamma, s, t, r, \delta)$ with $Q = \{s, t, r, q_1, q_2, q_3, q_4, q_5\}$, $\Sigma = \{a, b\}$, $\Gamma =$

$\{a, b, \overline{a}, \overline{b}, \llcorner\!\lrcorner\}$, and

$$
\begin{aligned}
&\delta(s, a, \llcorner\!\lrcorner) = (q_1, a, \overline{a}, +1, +1)\\
&\delta(s, x, y) = (r, 0, 0) \text{ for all other } (x, y) \in \Gamma \times \Gamma\\[1em]
&\delta(q_1, a, \llcorner\!\lrcorner) = (q_1, a, a, +1, +1)\\
&\delta(q_1, b, \llcorner\!\lrcorner) = (q_2, b, \llcorner\!\lrcorner, 0, -1)\\
&\delta(q_1, x, y) = (r, x, y, 0, 0) \text{ for all other } (x, y) \in \Gamma \times \Gamma\\[1em]
&\delta(q_2, b, a) = (q_2, b, a, +1, -1)\\
&\delta(q_2, b, \overline{a}) = (q_3, b, \overline{a}, +1, 0)\\
&\delta(q_2, x, y) = (r, x, y, 0, 0) \text{ for all other } (x, y) \in \Gamma \times \Gamma\\[1em]
&\delta(q_3, a, \overline{a}) = (q_3, a, \overline{a}, +1, +1)\\
&\delta(q_3, a, a) = (q_3, a, a, +1, +1)\\
&\delta(q_3, a, \llcorner\!\lrcorner) = (q_4, a, \llcorner\!\lrcorner, 0, -1)\\
&\delta(q_3, x, y) = (r, x, y, 0, 0) \text{ for all other } (x, y) \in \Gamma \times \Gamma\\[1em]
&\delta(q_4, a, a) = (q_4, a, a, +1, -1)\\
&\delta(q_4, a, \overline{a}) = (q_5, a, \overline{a}, +1, 0)\\
&\delta(q_4, x, y) = (r, x, y, 0, 0) \text{ for all other } (x, y) \in \Gamma \times \Gamma\\[1em]
&\delta(q_5, \llcorner\!\lrcorner, \overline{a}) = (t, \llcorner\!\lrcorner, \overline{a}, 0, 0)\\
&\delta(q_5, x, y) = (r, x, y, 0, 0) \text{ for all other } (x, y) \in \Gamma \times \Gamma
\end{aligned}
$$

Question 2 — 15 Points

Is $L = \{\ulcorner M \urcorner \mid 1101 \in L(M)\}$ computable? Give a formal argument.

Solution

Given a DTM M and an input w for M, let $f(M, w)$ be a DTM with the following behaviour:

1. Delete the input from the tape and write w on the tape.
2. Simulate M.

3. If the simulation halts, accept.

Thus, $1101 \in L(f(M,w))$ if and only if M halts on w.

The function mapping M and w to $f(M,w)$ is computable. Hence, the function

$$g(x) = \begin{cases} \ulcorner f(M,w) \urcorner & \text{if } x = \ulcorner \langle M, w \rangle \urcorner \text{ for some DTM } M \text{ and some input } w \\ \varepsilon & \text{otherwise} \end{cases}$$

witnesses HP $\leq_m L$. Thus, L is not computable.

Question 3 — 5 + 15 + 1 Points

Recall that

$$\mathrm{VC} = \{(G,k) \mid G \text{ is a graph that has a } k\text{-vertex cover}\}.$$

An independent set of a graph $G = (V,E)$ is a set $I \subseteq V$ of vertices such that no two vertices in I are connected by an edge in G. We define

$$\textsc{IndSet} = \{(G,k) \mid G \text{ is a graph that has an independent set with } k \text{ vertices}\}.$$

1. Show $\textsc{IndSet} \in \mathrm{NP}$.
2. Show $\mathrm{VC} \leq_p \textsc{IndSet}$.

   Hint: Draw some small graphs, some vertex cover in it, and find an independent set in the graph. Then, draw some small graphs, some independent set in it, and find a vertex cover in the graph.
3. Is $\textsc{IndSet}$ NP-complete? Explain your answer.

Solution

1. The following algorithm can be implemented on an NTM that runs in polynomial time.

   (a) If the input does not encode a graph $G = (V,E)$ and a natural number k reject.

   (b) If $k > |V|$ reject.

   (c) Guess a subset $I \subseteq V$ of size k.

   (d) For all $v, v' \in I$: If $(v, v') \in E$ reject.

   (e) Accept.
2. Let C be a vertex cover in a graph (V,E). For any two vertices (v, v') not in C, there cannot be an edge between v and v', as it would not be covered by C. Hence, $V \setminus C$ is an independent set.

Similarly, let I be an independent set. Then $V \setminus I$ is a vertex cover: let $(v, v') \in E$ be an edge. Then, we have $v \notin I$ or $v' \notin I$, as I is an independent set. Hence, at least one of the vertices v and v' is in $V \setminus I$ and thus covered by $V \setminus I$.

Hence, there is a tight relation between vertex covers and independent sets in a graph. Formally, we have that (V, E) has a vertex cover of size k if and only if it has an independent set of size $|V| - k$. Hence, the (polynomial-time computable) function

$$f(x) = \begin{cases} (G, |V| - k) & \text{if } x \text{ encodes a graph } G = (V, E) \text{ and a } k \leq |V| \\ \varepsilon & \text{otherwise} \end{cases}$$

witnesses VC $\leq_p$ IndSet.

3. In Item 1, we have shown that IndSet is in NP, in Item 2 we have shown that IndSet is NP-hard. Altogether it is therefore NP-complete.

Question 4 — 10 Points

The partition problem

$$\text{PARTITION} = \{(n_1, n_2, \ldots, n_k) \in \mathbb{N}^* \mid \text{ there exists } S \subseteq \{1, 2, \ldots, k\} \text{ such that } \sum_{j \in S} n_j = \sum_{j' \notin S} n_{j'}\}$$

asks whether a sequence $(n_1, n_2, \ldots, n_k)$ of natural numbers can be split into two parts that sum up to the same value.

For example $(3, 1, 1, 2, 2, 1) \in$ PARTITION, as $1 + 1 + 1 + 2 = 3 + 2$, but $(2, 3, 4) \notin$ PARTITION.

Show that PARTITION is expressible as a 0-1 Integer Linear Program.

Solution

Given an instance $(n_1, n_2, \ldots, n_k)$ we construct a 0/1 integer linear program (with a trivial objective function, say $\max 1$) over the decision variables $x_1, \ldots x_k$ ranging over $\{0, 1\}$. Intuitively, x_j is 1 for those n_j in S and x_j is 0 for those n_j not in S.

Note that $(1 - x_j)$ is equal to 0 if x_j is 1, and is equal to 1 if x_j is 0. Thus, the constraint

$$\sum_{j=1}^{k} x_j \cdot n_j = \sum_{j'=1}^{k} (1 - x_{j'}) \cdot n_{j'}$$

expresses that the numbers in S (left side of the equality) add up to the same value as the numbers not in S (right side of the equality).

However, the constraint is not in standard form, but can easily be rewritten into

$$\sum_{j=1}^{k} 2n_j \cdot x_j \leq \sum_{j=1}^{k} n_j$$
$$\sum_{j=1}^{k} -2n_j \cdot x_j \leq -\sum_{j=1}^{k} n_j$$

which is in standard form.

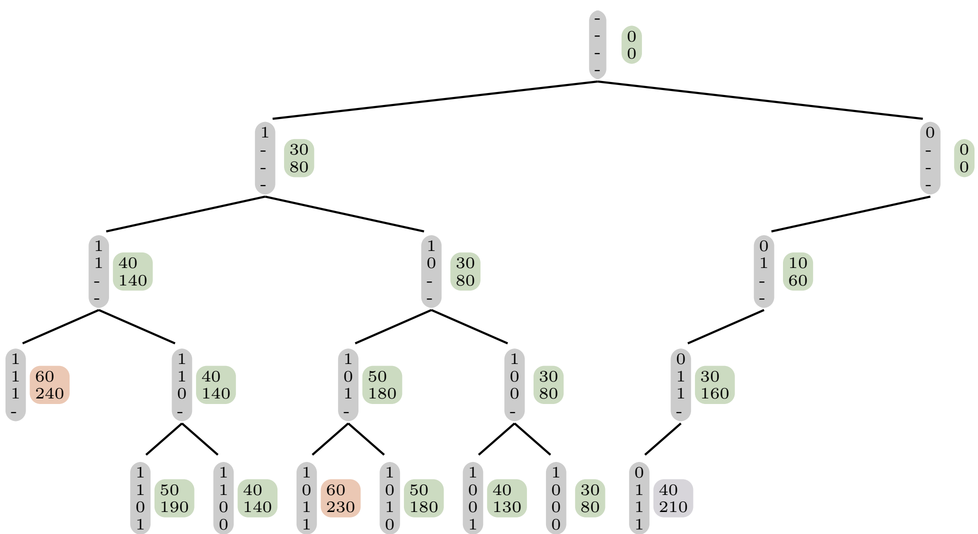
Question 5 — 10 Points

Apply the backtracking algorithm for the knapsack problem to the following instance: Size $W = 50$, threshold $T = 200$, and the following items:

item	1	2	3	4
weight	30	10	20	10
value	80	60	100	50

Show the tree of partial solutions until a valid solution is found or the algorithm terminates.

Solution



Question 6 — 7 + 7 Points

Apply the DPLL algorithm to the following formulas and determine whether they are satisfiable or unsatisfiable.

Illustrate the execution of the algorithm using a tree structure, i.e., showing the simplified formula and the rule applied.

1. $\varphi_1 = (\neg A \vee C \vee \neg D) \wedge (A \vee B \vee C \vee \neg D) \wedge (\neg A \vee \neg E) \wedge \neg C \wedge (A \vee D) \wedge (A \vee C \vee E) \wedge (D \vee E)$
2. $\varphi_2 = (A \vee B \vee \neg D) \wedge (\neg B \vee A) \wedge (\neg A \vee B \vee C) \wedge (\neg B \vee \neg A \vee C) \wedge \neg C \wedge (C \vee D)$

Hint: You can use & for $\wedge$, | for $\vee$, and ! for $\neg$ if it is more convenient.

Solution

1.

$$(\neg A \vee C \vee \neg D) \wedge (A \vee B \vee C \vee \neg D) \wedge (\neg A \vee \neg E) \wedge \neg C \wedge (A \vee D) \wedge (A \vee C \vee E) \wedge (D \vee E)$$

Unit prop. with $\neg C$

$$(\neg A \vee \neg D) \wedge (A \vee B \vee \neg D) \wedge (\neg A \vee \neg E) \wedge (A \vee D) \wedge (A \vee E) \wedge (D \vee E)$$

Pure literal with B

$$(\neg A \vee \neg D) \wedge (\neg A \vee \neg E) \wedge (A \vee D) \wedge (A \vee E) \wedge (D \vee E)$$

Splitting
($A = 0$ branch)

$$(D) \wedge (E) \wedge (D \vee E)$$

Unit prop. with D

$$(E)$$

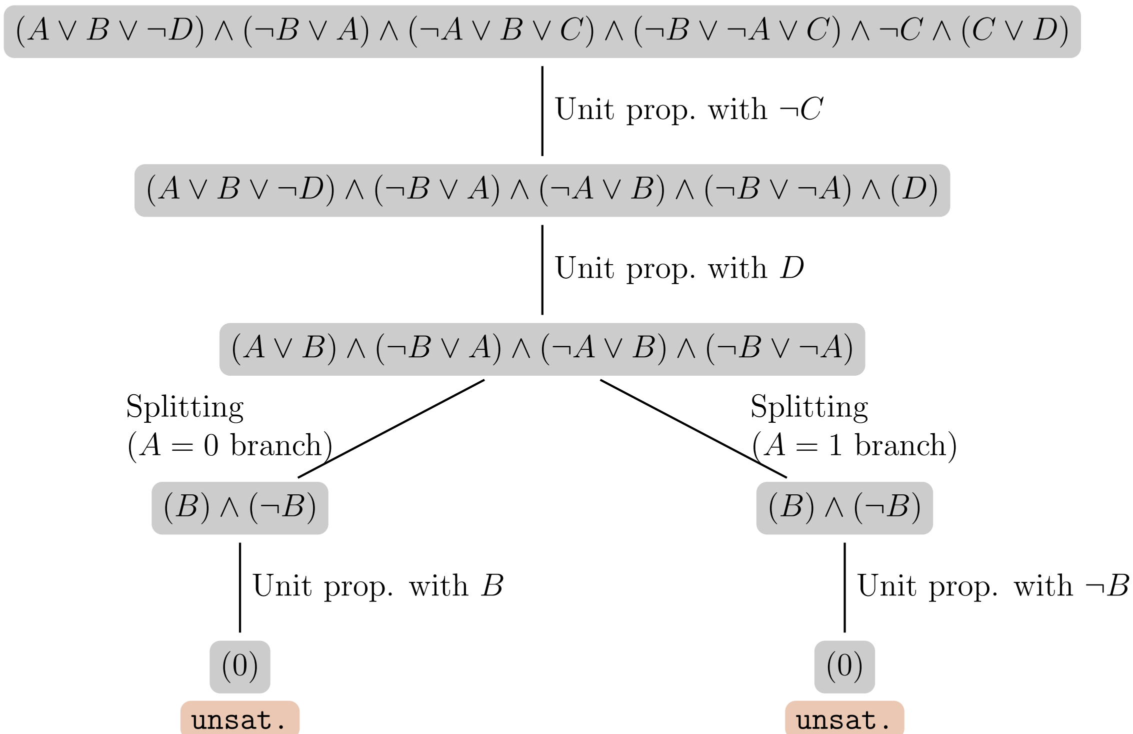
Unit prop. with E

$$(1)$$

`sat.`

The algorithm returns `"satisfiable"`.

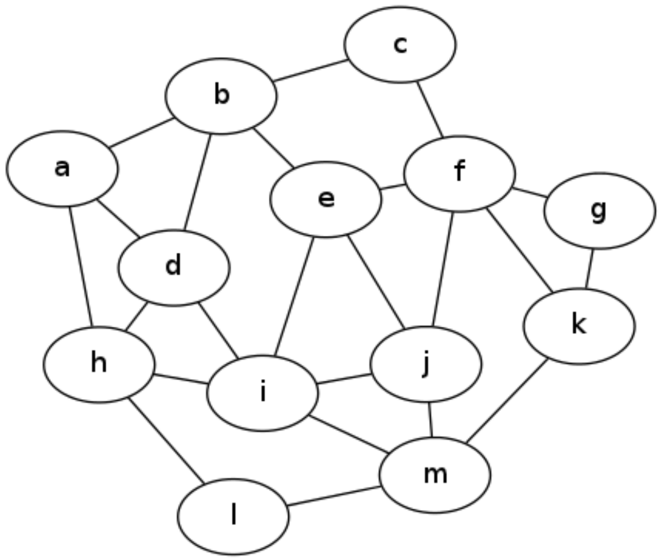
2.



The algorithm returns `"unsatisfiable"`.

Question 7 10 Points

Apply the approximation algorithm for vertex cover to the following graph.



Show the intermediate value of the tentative cover for each iteration.

Solution

- In the first iteration, we cover the edge (a, b) by adding a, b to C covering the edges (a, b), (a, d), (a, h), (b, c), (b, d), and (b, e).

  Thus, after the first iteration, we have $C = \{a, b\}$.

- In the second iteration, we cover the edge (i, j) by adding i, j to C covering the edges (i, d), (i, e), (i, h), (i, j), (i, m), (j, e), (j, f), and (j, m).

  Thus, after the second iteration, we have $C = \{a, b, i, j\}$.

- In the third iteration, we cover the edge (f, k) by adding f, k to C covering the edges (c, f), (e, f), (f, g), (f, k), (g, k), and (k, m).

  Thus, after the third iteration, we have $C = \{a, b, f, i, j, k\}$.

- In the fourth iteration, we cover the edge (h, l) by adding h, l to C covering the edges (d, h), (h, l), and (l, m).

  Thus, after the fourth iteration, we have $C = \{a, b, f, h, i, j, k, l\}$.

- Now, all edges are covered and the algorithm terminates with the vertex cover $C = \{a, b, f, h, i, j, k, l\}$