Name ______________________

Student Number ______________________

HAND IN
answers recorded
on question paper

BISHOP'S UNIVERSITY

Department of Computer Science

CS 467/567

FIRST EXAMINATION

26 February 2025

Instructor: Stefan D. Bruda

Instructions

- This examination is *80 minutes in length* and is *open book*. You are allowed to use any documentation you like. Electronic devices are permitted only if they demonstrably have no communication capabilities. You are *not* allowed to share material with your colleagues. *Any violation of these rules will result in the complete forfeiture of the examination.*
- There is no accident that the total number of marks add up to the length of the test in minutes. The number of marks awarded for each question should give you an estimate on how much time you are supposed to spend answering that question.
- *To obtain full marks provide all the pertinent details.* This being said, do not give unnecessarily long answers. In principle, all your answers should fit in the space provided for this purpose. If you need more space, use the back of the pages or attach extra sheets of paper. However, if your answer is not (completely) contained in the respective space, clearly mention within this space where I can find it.
- Make sure that your name and student number appear on top of each sheet which is not securely stapled to the booklet (just in case). This also applies to any sheet which you detach from the booklet.
- The number of marks for each question appears in square brackets right after the question number. If a question has sub-questions, then the number of marks for each sub-question is also provided.

When you are instructed to do so, turn the page to begin the test.

1		15	/	15			
2	a,b	15	/	15			
3	a,b,c	20	/	20			
4	a,b	15	/	15			
5		5	/	5			
6		10	/	10			
	Total:	80	/	80	= 20	/	20

1. [15] The HAMILTONIAN PATH problem is stated as follows: *Given a graph G, determine whether there exists a path that includes all the vertices of G exactly once.* Recall that the HAMILTONIAN CYCLE problem requires the existence of a cycle (that includes all the vertices of G exactly once) rather than a path. We note that HAMILTONIAN PATH is a generalization of HAMILTONIAN CYCLE given that the Hamiltonian cycle $(v_1, \ldots, v_n)$ is also a Hamiltonian path: sure there is an edge between v_n and v_1 (which completes the cycle), but nobody requires that we consider it so we can ignore that vertex and so $(v_1, \ldots, v_n)$ becomes a path (which is clearly Hamiltonian if the cycle was Hamiltonian to begin with since it contains the same sequence of vertices).

   Given the above observation it is tempting (but incorrect) to state that the identity function is a polynomial reduction from HAMILTONIAN CYCLE to HAMILTONIAN PATH. Explain why this is *not* the case.

ANSWER:

The identity function is clearly computable in polynomial time (there is nothing to compute so the "transformation" takes no time at all). In order for identity to be a reduction it must hold that G has a Hamiltonian cycle iff G has a Hamiltonian path. The "if" part clearly holds as explained above (we remove an edge from a cycle and it becomes a path containing the same vertices). The "only if" part however does not hold, since it is possible for a graph to have a Hamiltonian path without having a Hamiltonian cycle. Indeed, a path must be closed in order to become a cycle by one extra edge, yet the appropriate edge may not exist. Consider for example the graph $(\{v_1, v_2, v_3\}, \{(v_1, v_2), (v_2, v_3)\})$: it clearly has a Hamiltonian path (namely, $\langle v_1, v_2, v_3 \rangle$), yet this path cannot be converted into a Hamiltonian cycle since the edge (v_3, v_1) does not exist (and it is easy to see that there is no Hamiltonian cycle at all in this graph).

2. [15] The QUANTIFIED BOOLEAN FORMULA problem (QBF for short) is a generalization of SAT and is formulated as follows: *Given a set $X = \{x_1, x_2, \dots, x_n\}$ of Boolean variables and a quantified Boolean formula over X of form $\mathbb{Q}_1 x_1 \mathbb{Q}_2 x_2 \dots \mathbb{Q}_n x_n \phi$ with $\mathbb{Q}_i \in \{\forall, \exists\}$ and ϕ a formula in conjunctive normal form (as for SAT), find whether the given formula is satisfiable.*

   QBF instances include universal quantifiers whereas in SAT all the variables are existentially quantified. For example both formulae below are instances of QBF but only the first is an instance of SAT:

$$\exists x_1 \, \exists x_2 \, \exists x_3 \, \exists x_4 : (x_1 \vee \neg x_3 \vee x_4) \wedge (\neg x_2 \vee x_3 \vee \neg x_4)$$
$$\exists x_1 \, \forall x_2 \, \exists x_3 \, \forall x_4 : (x_1 \vee \neg x_3 \vee x_4) \wedge (\neg x_2 \vee x_3 \vee \neg x_4)$$

   In a SAT instance (as seen in class) we further remove the quantifiers ($\forall$ and $\exists$) since there is only one of them.

   (a) [10] Find a polynomial reduction from SAT to QBF. Prove that your function is a reduction and can be computer in polynomial time.

ANSWER:

SAT being an particular case of QBF, a reduction is just the identity function, with the addition that we need to add the quantifiers (that are implicit in SAT). We end up with $\tau(\psi) = \exists x_1 \exists x_2 \dots \exists x_n \psi$ for every SAT instance ψ over $\{x_1, x_2, \dots, x_n\}$. That this is indeed a reduction follows immediately from the fact that QBF is a generalization of SAT.

The reduction is clearly polynomial: we need to copy the original formula ψ (which takes linear time) and then add the n groups $\exists x_i$ in front (which also takes linear time since the number of clauses is no less than twice the number of variables; indeed, each clause contains at least one literal which is either a variable or its negation).

(b) [5] What is the consequence of having a polynomial reduction from SAT to QBF as in Question 2a? Explain.

Answer:

Since we have a reduction from SAT (which is NP-complete) to QBF we conclude that QBF is NP-hard. However tempting it may be to conclude that, it is not necessary that QBF is NP-complete, since QBF is not necessarily in NP (actually QBF is PSPACE-complete and so unlikely to be in NP).

---

3. [20] The MAX-SAT problem is formulated as follows: Giver a not necessarily satisfiable SAT formula (conjunction of disjunctive clauses), find an interpretation that maximizes the number of true clauses in the formula.

   Let $n(l, \phi)$ be the number of clauses from the formula ϕ containing the literal l. The following greedy strategy offers an approximation algorithms for MAX-SAT: For an input formula ϕ, pick a literal l with maximum $n(l, \phi)$ and set the variable in l such that l is true (and so all the clauses that contain l are satisfied). Delete from ϕ all the clauses containing l and $\overline{l}$. Repeat until ϕ contains no literals.

   (a) [5] Show that the above strategy can be implemented in polynomial time.

   Answer:

   Let n be size of the input (i.e., the number of literals in ϕ). Note first that the number of clauses in the formula is also $O(n)$ (since each clause contains at least one literal). Finding the literal with maximum $n(l, \phi)$ can be done through a scan of the formula that is, in $O(n)$ time (keeping one counter for each variable). Deleting clauses is also done through a scan and so in $O(n)$ time. One iteration thus takes $O(n) + O(n) = O(n)$ time. In the worst case each iteration will only delete one clause and so $O(n)$ iterations are necessary. It follows that the processing is done in $O(n) \times O(n) = O(n^2)$ time.

   (b) [5] What is the upper bound for the number of true clauses in an exact (optimal) solution for MAX-SAT?
   Hint. What happens if the formula is satisfiable?

   Answer:

   If the formula is satisfiable then there exists an interpretation that makes all the clauses in the formula true and so the number of true clauses in the optimal solution to MAX-SAT equals the number of clauses in the formula. Since there can be no more true clauses than there are clauses in the formula this is a (tight) upper bound.

(c) [10] Prove that that the above strategy is a 2-approximation algorithm for MAX-SAT. *Hint.* Let the formula ϕ contain k variables and m clauses. Show that the above strategy makes at least $m/2$ clauses true by induction over k. Use the fact that that when we pick the literal l if m^+ clauses contain l and m^- clauses contain $\overline{l}$ then $m^+ \geq m^-$ (for otherwise we would have picked $\overline{l}$).

ANSWER:

The induction mentioned in the hint goes like this:

Basis: For $k = 1$ the only literals are x_1 and $\overline{x_1}$. Picking the literal with the maximum number of occurrences will certainly make more than half the clauses true (else it will not be the literal with the maximum number of occurrences).

Inductive hypothesis: The algorithm satisfies at least $m'/2$ of the m' clauses from a formula over $k-1$ variables.

Inductive step: We pick l with maximum $n(l, \phi)$. We have m^+ clauses that are true and after deleting clauses we are left with $m - m^+ - m^-$ clauses over $k-1$ variables. By induction hypothesis at least $(m - m^+ - m^-)/2$ of these clauses are true, so overall at least $(m - m^+ - m^-)/2 + m^+$ clauses are true. We have $(m - m^+ - m^-)/2 + m^+ = (m - m^+ - m^- + 2m^+)/2 = (m + m^+ - m^-)/2 \geq m/2$ (since $m^+ \geq m^-$).

In all m is an upper bound for the cost of the exact solution (see previous question) and $m/2$ is a lower bound for the cost of the approximate solution, so the approximation ratio of the algorithm is no worse than $m/(m/2) = 2$.

---

4. [15] Recall that the MINIMUM VERTEX COVER problem is the problem of finding the minimum vertex cover of a given graph $G = (V, E)$.

   Let $V = \{v_1, v_2, \ldots, v_n\}$. We define a set $C \subseteq V$ using the variables x_i as follows: $x_i = 1$ whenever $v_i \in C$ and $x_i = 0$ otherwise, $1 \leq i \leq n$. Consider then the following optimization problem over the variables x_i, $1 \leq i \leq n$:

Minimize:	$x_1 + x_2 + \cdots + x_n$	
Subject to:	$x_i + x_j \geq 1$	for each $(i, j) \in E$
	$x_i \in \{0, 1\}$	for each $v_i \in V$

   (a) [10] Is the above optimization problem (over the variables x_i) equivalent to MINIMUM VERTEX COVER? Justify your answer.

   ANSWER:

   The two problems are equivalent.

   The second constraint specifies a subset C of V. Indeed, the only possible values for x_i are 0 and 1, which means that the vertex v_i can either be in C or not (with no other alternative).

   The first constraint specifies that C is a vertex cover. Indeed, $x_i + x_j \geq 1$ means that at least one of the two variables x_i and x_j are assigned 1 and so either v_i or v_j (or both) are in C. This in turn means that C covers the edge (v_i, v_j). Since we have such a constraint for every edge in the graph, C covers all the edges and so is a vertex cover.

   Note now that $x_1 + x_2 + \cdots + x_n = |C|$ and so minimizing $x_1 + x_2 + \cdots + x_n$ minimizes the size of C.

   Putting everything together we have that C is a vertex cover (because of the constraints) and $|C|$ is minimized (because of the objective function), and so C is a minimum vertex cover.

(b) [5] Is the above optimization problem (over the variables x_i) a linear programming problem? If not, can the constraints be reformulated so that it becomes a linear programming problem? Justify your answer (one way or another).

ANSWER:

The problem is not a linear programming problem because the constraint $x_i \in \{0, 1\}$ is not a linear constraint. Furthermore, it is not possible to construct linear constraints equivalent to this constraint. Indeed, a linear constraint can only define a hyper line or a hyper half plane, while $x_i \in \{0, 1\}$ defines a couple of points (which cannot constitute a hyper line, let alone a hyper half plane).

---

5. [5] Given an example (graphical or otherwise) or a linear program for which the simplex region is not bounded but the optimal objective value is finite. Explain why is this the case.

ANSWER:

We can make the simplex unbounded in one direction and the objective function maximized in the other direction. For example we make the simplex unbounded to the right and the objective function maximized to the left, as follows:

Maximize $-x_1$

subject to the constraints: $x_1 \geq 3$, $x_2 \leq 5$, and $x_2 \geq 0$

Alternatively, we can have an objective function that only depends on some variables and so the other variables can be unbounded and we still have a finite optimal objective value.

---

6. [10] Let $P = \langle p_0, p_1, \ldots, p_{n-1} \rangle$ be a sequence of n points. Someone claims that the following algorithms checks whether P describes a convex polygon in counterclockwise order:

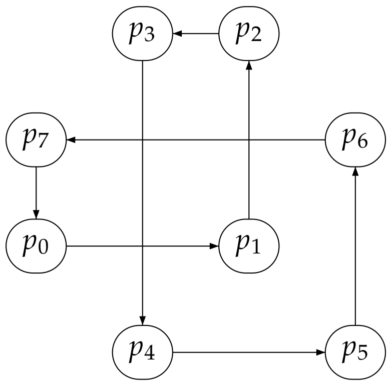
```
algorithm Is-Convex($\langle p_0, p_1, \ldots, p_{n-1} \rangle$):
    for $i = 0$ to $n - 1$ do
        if $(p_i, p_{(i+1) \bmod n}, p_{(i+2) \bmod n})$ form a right turn then
            return False
    return True
```

Disprove this claim.

Hint. A False return definitely means that the polygon is not convex. Try to construct a sequence of points that does not even form a polygon yet only makes left turns.

Answer:

We can keep making right turns until an edge of the polygon intersects a previous edge. The moment this happens the polygon is not convex anymore. Here is an example:


p_3
p_2
p_7
p_6
p_0
p_1
p_4
p_5