Foundations of Computing: Discrete Mathematics

Exam
January 5, 2015

Instructions (Read Carefully)

What to check. In the multiple-choice questions, there is one and only one correct answer. However, to demonstrate partial knowledge, you are allowed to check 2 or more boxes, but this earns you less than full points for that question. If the correct answer is not among your checked boxes, the score will even be negative. If you don't check anything (or check *all* boxes) your score is 0. Also, if you check boxes at random, your expected score is 0. For more details, read:

[G. S. Frandsen, M. I. Schwartzbach: A singular choice for multiple choice. SIGCSE Bulletin 38(4): 34–38 (2006)].

Useful Definitions. At the end of this document, you can find some definitions that could be useful for answering some questions.

Info about you. Write your full name and CPR no. on every page (top-right).

- IMPORTANT -
Only information written on the first 8 pages will be evaluated.
Anything else that you hand-in will NOT be considered for the final evaluation!

1. Answer the following multiple choice questions:

(a) (*1 pt.*) Consider the following argument:

1	$p \wedge q$	(premise)
2	p	(from 1 using Simplification)
3	$r \to t$	(premise)
4	$p \wedge (r \to t)$	(from 2 and 3 using Conjunction)
5	$\overline{p} \vee (p \wedge (r \to t))$	(from 4 using Addition)

Which of the following tautologies does it prove?

$\boxed{A}$ $((p \wedge q) \wedge (r \to t)) \to (\overline{p} \vee (p \wedge (r \to t)))$

$\boxed{B}$ $\overline{p} \vee (p \wedge (r \to t))$

$\boxed{C}$ $p \wedge q \wedge (r \to t)$

$\boxed{D}$ $p \vee \overline{p}$

(b) (*1 pt.*) If $a = 1800$ and $b = 420$, then

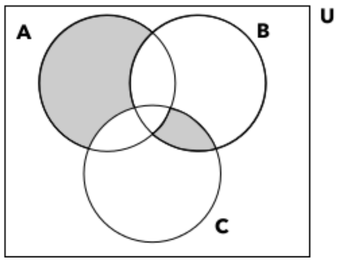
$\boxed{A}$ $\gcd(a, b) = 2^2 \cdot 3 \cdot 5 \cdot 7$

$\boxed{B}$ $lcm(a, b) = 2^3 \cdot 3^2 \cdot 5^2$

$\boxed{C}$ $lcm(a, b) = 2^3 \cdot 3^2 \cdot 5^2 \cdot 7$

$\boxed{D}$ $gcd(a, b) = 2^3 \cdot 3 \cdot 5^2 \cdot 7$

(c) (*1 pt.*) Let A and B and C be subsets of a universal set U. Which of the following sets is equal to the shaded region in the following Venn diagram?



A $A \cap (B \cup C)$

B $(A \cup (B \cap C)) \setminus (A \cap (B \cup C))$

C $(A \setminus B) \cap (A \setminus C)$

D $(A \setminus (B \cup C)) \cup (B \cap C)$

(d) (*1 pt.*) Let

$$f(n) = \frac{n}{2} + \frac{1-(-1)^n}{4}$$

be a function $f : \mathbb{Z} \to \mathbb{Z}$ for all $n \in \mathbb{Z}$, where $\mathbb{Z}$ is the set of all integers. Which of the following statements is correct?

A f is not a function from $\mathbb{Z}$ to $\mathbb{Z}$ because $\frac{n}{2} \notin \mathbb{Z}$.

B f is a function and is onto but not one-to-one.

C f is a function and is onto and one-to-one.

D f is a function and is not onto nor one-to-one.

(e) (*1 pt.*) Let $R = \{(0,1),(1,2),(2,3),(3,2),(2,0)\}$ be a relation on the set $\{0,1,2,3\}$. What is the transitive closure of R?

A $\{(0,1),(1,2),(2,3),(3,2),(2,0),(0,0),(1,1),(2,2),(3,3)\}$

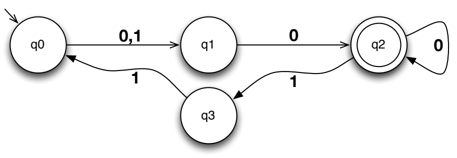
B $\{(0,1),(1,2),(2,3),(3,2),(2,0),(0,2),(1,3),(1,0),(2,2),(3,3),(3,0),(2,1)\}$

C $\{(0,1),(1,2),(2,3),(3,2),(2,0),(0,2),(1,3),(1,0),(2,2),(3,3),(3,0),(2,1),(0,0),(0,3),(1,1),(3,1)\}$

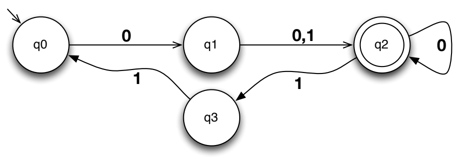
D $\{(0,2),(1,3),(1,0),(2,2),(3,3),(3,0),(2,1),(0,0),(0,3)\}$

(f) (*1 pt.*) Which of the following automata recognises the language defined by the regular expression $(00+10)0^*((1100+1110)0^*)^*$

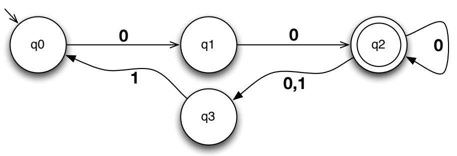
A



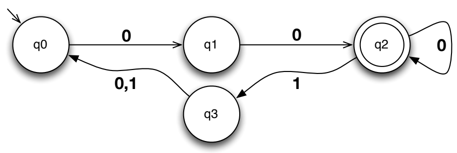
B



C



D



(g) (*1 pt.*) Say which of the following grammars generates the language:

$$\{w\, 0^{n+1}\, 1^n \quad | \quad w \in \{0,1\}^* \text{ and } n \geq 0\}$$

$\boxed{\text{A}}$
$$\begin{array}{lcllll} S & \rightarrow & AB & | & \epsilon & \\ A & \rightarrow & 0A & | & 1A & | \ \epsilon \\ B & \rightarrow & 0B1 & | & \epsilon & \end{array}$$

$\boxed{\text{B}}$
$$\begin{array}{lclll} S & \rightarrow & AB & | & \epsilon \\ A & \rightarrow & 0A & | & 1A \\ B & \rightarrow & 0B1 & & \end{array}$$

$\boxed{\text{C}}$ Impossible, no grammar can generate such language.

$\boxed{\text{D}}$
$$\begin{array}{lcllll} S & \rightarrow & A0B & | & \epsilon & \\ A & \rightarrow & 0A & | & 1A & | \ \epsilon \\ B & \rightarrow & 0B1 & | & \epsilon & \end{array}$$

(h) (*1 pt.*) Which of the following languages is *not* decidable?

$\boxed{\text{A}}$ $\{a^n b^n c^n \mid n \geq 0\}$

$\boxed{\text{B}}$ $\{w |$ w is a Java program with no syntax error$\}$

$\boxed{\text{C}}$ $\{w |$ w is a Java program that never crashes$\}$

$\boxed{\text{D}}$ $\{w |$ w is a Java program that simulates an automaton for recognising $10^*1\}$

(i) (*1 pt.*) Professor White teaches the same course in five different classes, each class belonging to a different university. For passing the exam, each student has to complete one of three possible projects. The only constraint that Professor White enforces is that students from the same class must choose the same project. On the other hand, the same project can be chosen by more than one class. In how many possible ways can the projects be assigned?

A 243

B 81

C 10

D 60

(j) (*1 pt.*) In year 2185, planet Earth is politically divided into two countries: Country A and Country B. In both countries, citizens who are sick can get a free appointment at their local GP (general practitioner). After a short visit, the GP can decide to send patients back home and wait for three days or proceed with further expensive tests. It is known that among those patients who are sent back home, 1% of them die, possibly because of something that needed immediate treatment. On the other hand, 0.9% of the patients who receive further tests immediately after visiting the GP also die.

In Country A, GPs send patients back home in 90% of the cases. On the other hand, GPs from Country B send patients back home without further tests only in 10% of the cases.

What is the probability that a patient going to a GP in Country A dies? How about Country B?

A 0.99% in Country A and 0.91% in Country B

B 0.99% in Country A and 0.99% in Country B

C 0.91% in Country A and 0.91% in Country B

D 0.91% in Country A and 0.99% in Country B

2. Answer the following questions. Be brief but precise, your correct use of mathematical notation is an important aspect.

(a) (*2 pt.*) Give an example of a simple and undirected graph for each one of the specified conditions, or reason why no such graph exists. In order to give an example, either draw the corresponding graph or give the set of vertices and edges.

i) A tree with six vertices and six edges.

ii) A disconnected graph with 10 vertices and 8 edges.

iii) A graph with seven vertices: one vertex of degree 1, one of degree 2, one of degree 3, two vertices of degree 4, one of degree 5, and one vertex of degree 6.

(b) (*2 pt.*) Use the induction principle to show that for any positive integer n, the number $n + n^2$ is even.

Some useful information for the exam

Logics. Here are some of the rules for arguments in propositional logic.

$$\text{(Modus Ponens)}\ \begin{array}{c} p \\ p \rightarrow q \\ \hline \therefore q \end{array} \qquad \text{(Modus Tollens)}\ \begin{array}{c} \neg q \\ p \rightarrow q \\ \hline \therefore \neg p \end{array}$$

$$\text{(Addition)}\ \begin{array}{c} p \\ \hline \therefore p \vee q \end{array} \qquad \text{(Simplification)}\ \begin{array}{c} p \wedge q \\ \hline \therefore p \end{array} \qquad \text{(Conjuntion)}\ \begin{array}{c} p \\ q \\ \hline \therefore p \wedge q \end{array}$$

$$\text{(Or Elimination)}\ \begin{array}{c} p \vee q \\ \neg q \\ \hline \therefore p \end{array} \qquad \begin{array}{c} p \vee q \\ \neg p \\ \hline \therefore q \end{array}$$

Sets. A set is an (unordered) collection of objects, called *elements* or *members*.
The *union* of two sets A and B is the set

$$A \cup B = \{x \ : \ x \in A \vee x \in B\}.$$

The *intersection* of A and B is the set

$$A \cap B = \{x \ : \ x \in A \wedge x \in B\}.$$

Given n sets $A_1, A_2, \ldots, A_n$,

$$\bigcup_{i=1}^{n} A_i = A_1 \cup \ldots \cup A_n \qquad \bigcap_{i=1}^{n} A_i = A_1 \cap \ldots \cap A_n.$$

The *difference* of two sets A and B, denoted by $A - B$ (or by $A \setminus B$), is the set containing those elements in A but not in B.

The *Cartesian product* of two or more sets $A_1, A_2, \ldots, A_n$, denoted by $A_1 \times A_2 \times \ldots \times A_n$, is the set of all ordered n-tuples $(a_1, a_2, \ldots, a_n)$, where $a_i \in A_i$ for $1 \leq i \leq n$.

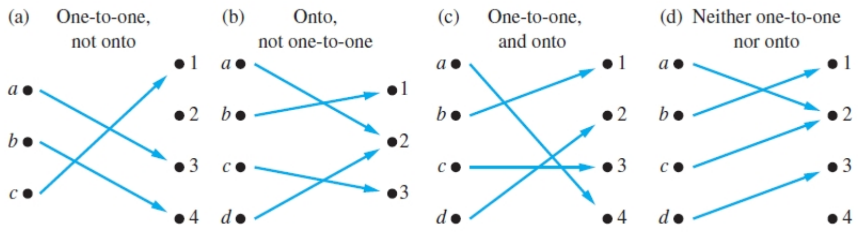
Functions. Given two non-empty sets A and B, a *function* f from A to B is an assignment of exactly one element of B to each element of A.

A function $f : A \to B$ is *onto* (or a surjection) if and only if for every element $b \in B$ there is an element $a \in A$ such that $f(a) = b$.

A function $f : A \to B$ is *one-to-one* (or an injunction) if $f(a) = f(b)$ implies $a = b$ for all a and b in the domain of f.

A function f is a *bijection* if it is both one-to-one and onto.

Example:



Relations. A relation $\mathcal{R}$ on a set A is a subset of the cartesian product $A \times A$.

A relation $\mathcal{R}$ on A is *reflexive* whenever

$$\forall a \in A.\ (a,a) \in \mathcal{R}$$

A relation $\mathcal{R}$ on A is called *symmetric* if

$$\forall a, b \in A, (a,b) \in \mathcal{R} \Rightarrow (b,a) \in \mathcal{R}.$$

A relation $\mathcal{R}$ on A is *antisymmetric* if

$$\forall a, b \in A,\ ((a,b) \in \mathcal{R} \wedge (b,a) \in \mathcal{R}) \Rightarrow a = b.$$

A relation $\mathcal{R}$ on A is *transitive* if

$$\forall a, b, c \in A,\ ((a,b) \in \mathcal{R} \wedge (b,c) \in \mathcal{R}) \Rightarrow (a,c) \in \mathcal{R}.$$

The *reflexive closure* of a binary relation $\mathcal{R}$ on A is the smallest reflexive relation on A that contains $\mathcal{R}$.

The *symmetric closure* of a binary relation $\mathcal{R}$ on A is the smallest symmetric relation on A that contains $\mathcal{R}$.

The *transitive closure* of a binary relation $\mathcal{R}$ on A is the smallest transitive relation on A that contains $\mathcal{R}$.

Probability Theory *Bayes' Theorem* allows to manipulate conditional probabilities:

$$p(A_i|B) = \frac{p(B|A_i)p(A_i)}{p(B)}$$

such that $p(B) = p(B|A_1)p(A_1) + p(B|A_2)p(A_2)$.

Choose r objects from n	Order matters, not all elements (r-permutations)	Order matters, all elements (permutations)	Order does not matter, not all elements (combinations)
Without repetitions	$P(n,r) = \frac{n!}{(n-r)!}$	$P(n,n) = n!$	$C(n,r) = \binom{n}{r} = \frac{n!}{r!\,(n-r)!}$
With repetitions	n^r	$\frac{n!}{n_1!n_2!\cdots n_k!}$ where $n = n_1 + n_2 + \ldots + n_k$	$\binom{n+r-1}{r}$

Number Theory. Given two integers a and b, with $a \neq 0$, we say that *a divides b* if there is an integer c such that $b = ac$, or in other words, if $\frac{b}{a}$ is an integer. If a divides b then a is a *factor* (or *divisor*) of b, and b is said to be a *multiple* of a.

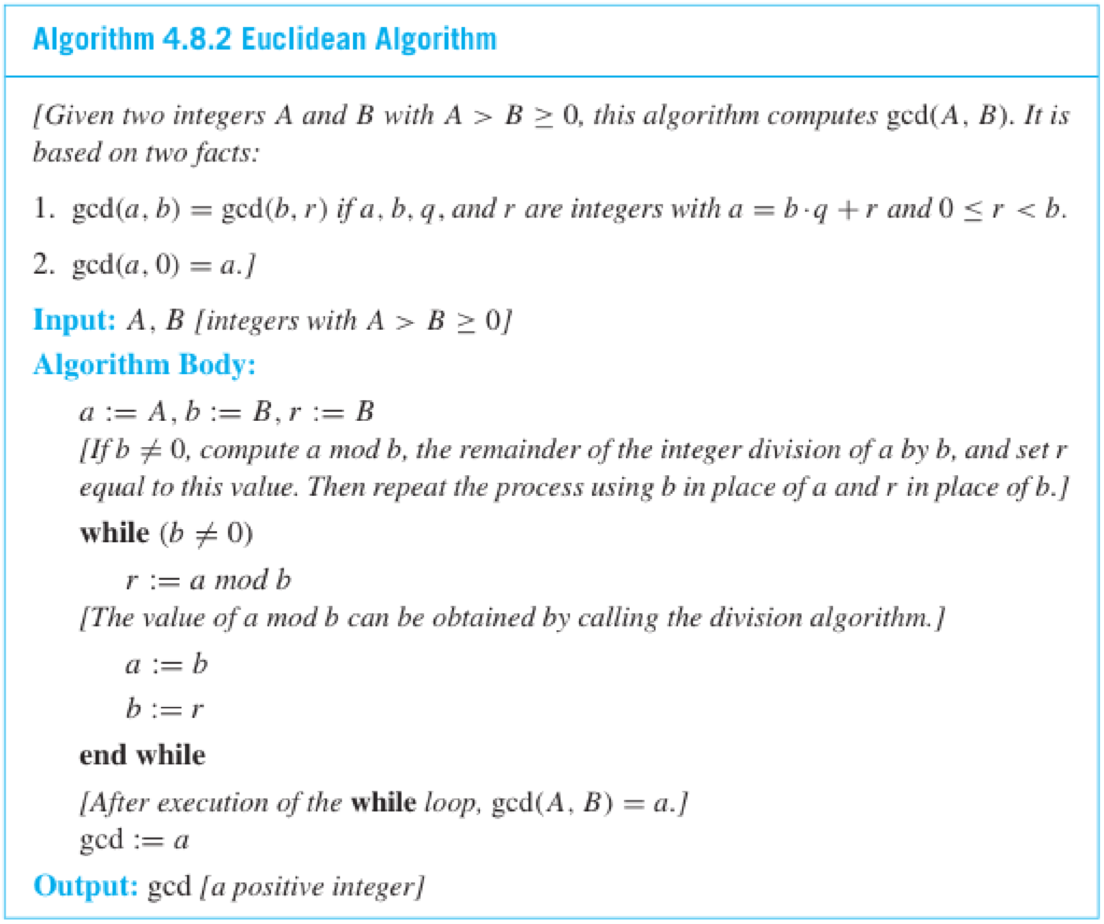
The *greatest common divisor* of two integers a and b, denoted by $\gcd(a,b)$, is the largest integer that divides both a and b.

The *Euclidean algorithm* provides a very efficient way to compute the greatest common divisor of two integers.

Given two positive integers a and b, the smallest positive integer that is a multiple of both a and b is the *least common multiple*, denoted by $\operatorname{lcm}(a,b)$.

The Quotient-Remainder Theorem. Let a be an integer and d a positive integer. Then there exist unique integers q and r, with $0 \leq r < d$, such that $a = dq + r$.

The value d is called the *divisor*, a is the *dividend*, q is the *quotient*, and r is the *remainder*. Then $q = a \text{ div } d$, $r = a \bmod d$. Remember that the remainder cannot be negative.



Algorithm 4.8.2 Euclidean Algorithm

[Given two integers A and B with $A > B \geq 0$, this algorithm computes $\gcd(A, B)$. It is based on two facts:

1. $\gcd(a, b) = \gcd(b, r)$ *if a, b, q, and r are integers with $a = b \cdot q + r$ and $0 \leq r < b$.*
2. $\gcd(a, 0) = a$.*]*

Input: A, B *[integers with $A > B \geq 0$]*

Algorithm Body:

```
$a := A, b := B, r := B$
[If $b \neq 0$, compute a mod b, the remainder of the integer division of a by b, and set r
equal to this value. Then repeat the process using b in place of a and r in place of b.]
while ($b \neq 0$)
    $r := a \bmod b$
[The value of a mod b can be obtained by calling the division algorithm.]
    $a := b$
    $b := r$
end while
[After execution of the while loop, $\gcd(A, B) = a$.]
gcd := $a$
```

Output: gcd *[a positive integer]*

Graph Theory. A graph $G = (V, E)$ is a structure consisting of a set of *vertices* (or nodes) V, and a set of *edges* E connecting some of these vertices.

Handshake Theorem. Let G be an undirected graph. Then,

$$\sum_{v \in V} deg(v) = 2m$$

where m is the number of edges of G and V is the set of vertices.

Let n be a nonnegative integer, and v, w two vertices in an undirected graph G.
A *walk from v to w* is an alternating sequence of vertices and edges

$$v_0 e_1 v_1 e_2 \cdots v_{n-1} e_n v_n$$

going from $v = v_0$ to $w = v_n$. We can repeat edges and vertices.
A *trail from v to w* is a walk from v to w with no repeated edges.

A *path from v to w* is a trail with no repeated vertices. Thus it is a sequence of vertices and edges with no repeated edges nor vertices.
A *circuit* is a trail that starts and ends at the same vertex, and has length greater than zero.

A circuit is *simple* if it does not contain repeat vertices (except the first and last).

An undirected graph is called *connected* if there is a walk between every pair of distinct vertices of a graph. Otherwise, it is called *disconnected*.

A *tree* is an undirected simple graph G that satisfies any of the following equivalent conditions:

a) G is connected and has no cycles.

b) G has no cycles, and a simple cycle is formed if any edge is added to G.

c) G is connected, but is not connected if any single edge is removed from G.

d) Any two vertices in G can be connected by a unique simple path.

A *trivial tree* is a graph that consists of a single vertex. A graph is called a *forest* if, and only if, it does not have any circuit and is not connected.

Decidability.

- A program can either *accept* an input or *reject* it.
- The set of strings accepted by a program P is the language *recognised* by P . A language is *Turing recognisable* if there exists some program recognising it.
- A program may *loop*, because either it terminates (accepting or rejecting), or it doesnt terminate.
- A program may fail to accept an input by either entering a rejecting configuration or by looping.
- A non-looping program is called a *decider*, it always accepts or rejects an input.
- A decider that recognises a language L is said to decide L. A language is *decidable* if some program decides it.