University of Ottawa
CSI 4105 – Final Exam
Instructor: Lucia Moura

December 21, 2001
9:30 am
Duration: 3 hs

Closed book; no calculators.

Last name: ______________________________

First name: ______________________________

Student number: ______________________________

There are 5 questions and 105 marks total.

This exam paper should have 18 pages,
including this cover page.

1 – Short answers	/ 25
2 – NP-completeness Reductions	/ 15
3 – Approximation Algorithms	/ 20
4 – Exact Exponential Algorithms	/ 20
5 – Set covering is NP-complete	/ 25
Total	/ 105

(The exam is out of 100 marks; there are 5 bonus marks.)

1 Short answers — 25 points

Some of the questions below are of the type "true or false"; others require short answers. In all cases, briefly justify your answer.

Note: You may use any result shown in class or in homework without proving it.

- If VERTEX-COVER $\in$ **P** then SAT $\in$ **P**. [TRUE] [FALSE]
  Justify:

- If $\mathbf{P} = \mathbf{NP}$ then **NP=co-NP** . [TRUE] [FALSE]

- It is possible that CLIQUE $\in$ **P** and HAM-CYCLE $\notin$ **P**. [TRUE] [FALSE]
  Justify:

... continued

- Let $\mathcal{C}_1$ be the set of languages that can be decided in polynomial time by a Turing machine. Let $\mathcal{C}_2$ be the set of languages that can be decided in polynomial time by a RAM program.
  Select one: [$\mathcal{C}_1 \subseteq \mathcal{C}_2$] [$\mathcal{C}_1 = \mathcal{C}_2$] [$\mathcal{C}_2 \subseteq \mathcal{C}_1$] [none of the previous]

  Justify:

- Let L_1 and L_2 be languages in **NP**. If $L_1 \leq_P L_2$ and $L_2 \leq_P L_1$, then both L_1 and L_2 are NP-complete. [TRUE] [FALSE]

  Justify:

- If there exists a 10-approximation algorithm for the general traveling salesman problem, then the decision problem corresponding to the integer programming problem can be decided in polynomial time. [TRUE] [FALSE]

  Justify:

- Consider the approximation algorithm for the minimum-weight vertex-cover problem, based on linear programming. Suppose that the optimal solution for the linear programming problem has minimum weight 10.5. Give an upper and a lower bound for the weight of the minimum-weight vertex cover.

  Lower Bound =
  Upper Bound =

  Justify:

- Suppose you face an optimization problem for which its corresponding decision problem is NP-complete. Which are the advantages and disadvantages of using an approximation algorithm over a branch-and-bound algorithm ?

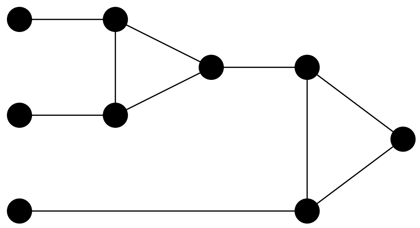
2 NP-completeness Reductions — 15 points

Part A — 8 points 3-CNF-SAT $\leq_P$ 3-COLOUR

Consider the following instance of 3-CNF-SAT:

$$\phi = (x_1 \vee x_2 \vee \neg x_3) \wedge (x_3 \vee x_4 \vee x_5).$$

A. Give the corresponding graph for the 3-COLOUR problem, applying the reduction used in Assignment 2.
Recall that the widget used in the reduction has the following form:



B. Give a satisfying truth assignment for ϕ:

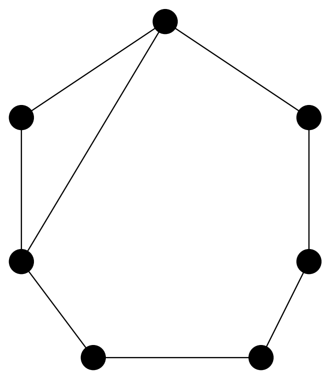
$x_1 =$ $x_2 =$ $x_3 =$ $x_4 =$ $x_5 =$

C. On top of the graph that you gave for part A, show a 3-colouring corresponding to the satisfying truth assignment that you gave in part B.

(this is a blank page)

Part B — 7 points CLIQUE $\leq_P$ VERTEX-COVER

Considering the following instance for the clique problem, give the corresponding instance of the vertex cover problem, according to the reduction studied:

Instance of CLIQUE:	Instance of VERTEX-COVER:
G : 	G' :
$k = 3$	$k' =$

... continued

3 Approximation Algorithms — 20 points

Part A — 7 points Parallel machine scheduling.

Consider the following instance for the parallel machine scheduling problem:

`number of machines:` $m = 2$
`number of jobs:` $n = 4$
`processing time of jobs:` $p_1 = 10,\ \ p_2 = 9,\ \ p_3 = 20,\ \ p_4 = 1$

If you apply the greedy approximation algorithm studied, what is the resulting schedule obtained?

`Machine 1:`

`Machine 2:`

What is the makespan C_{max} for this schedule ?

$C_{max} =$

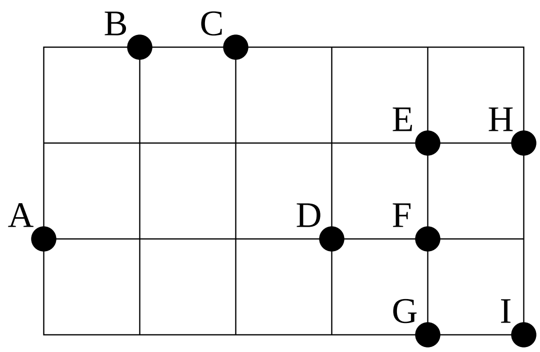
What is the **optimal** makespan C^*_{max} for this example? Justify.

$C^*_{max} =$

... continued

Part B — 13 points Traveling Salesman problem satisfying the triangle inequality

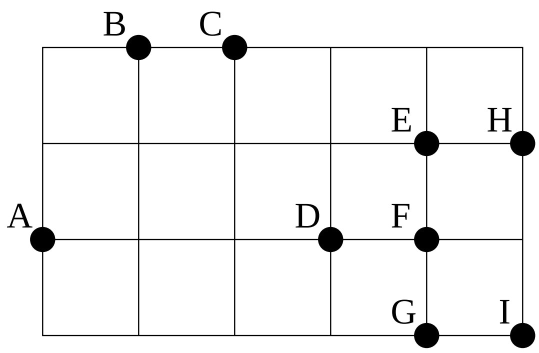
Consider the points in the unit grid below:



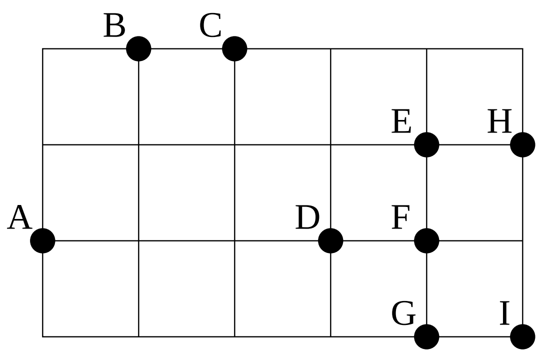
Let G the complete graph with vertices corresponding to points A, B, C, D, E, F, G, H, I. Let the cost on an edge $\{u, v\}$ be given by the distance in the plane between u and v.

Apply the approximation algorithm for the traveling salesman problem with the triangle inequality to the above example, showing the results of the two main steps in the following diagrams:

A. Compute the minimum spaning tree (using Prim's algorithm) from root A. Number the vertices according to a pre-order walk of the tree.



B. Show below the tour produced by the approximation algorithm.



... continued

C. For your convenience, the distances between vertices are calculated in the table below:

	A	B	C	D	E	F	G	H	I
A		$\sqrt{5}$	$\sqrt{8}$	3	$\sqrt{17}$	4	$\sqrt{17}$	$\sqrt{26}$	$\sqrt{26}$
B			1	$\sqrt{8}$	$\sqrt{10}$	$\sqrt{13}$	$\sqrt{18}$	$\sqrt{17}$	5
C				$\sqrt{5}$	$\sqrt{5}$	$\sqrt{8}$	$\sqrt{13}$	$\sqrt{10}$	$\sqrt{18}$
D					$\sqrt{2}$	1	$\sqrt{2}$	$\sqrt{5}$	$\sqrt{5}$
E						1	2	1	$\sqrt{5}$
F							1	$\sqrt{2}$	$\sqrt{2}$
G								$\sqrt{5}$	1
H									2
I									

What is the cost of the tour produced by the algorithm?

D. What is the approximation ratio of this algorithm?
Based on this approximation ratio and your answer for part C only, give a lower bound on the size of the optimal tour.

`approximation ratio:`

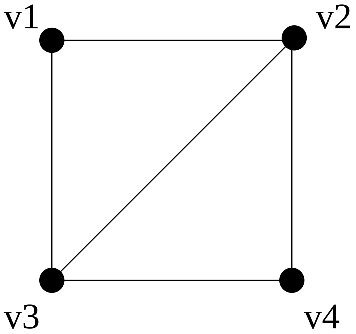
`lower bound on size of optimal tour:`

4 Exact Exponential Algorithms — 20 points

Part A — 8 points Backtracking

Consider the 3-colouring problem: Given a graph G, compute a 3-colouring of G, if one exists; otherwise decide that a 3-colouring does not exist.
Note that you do not have to generate all 3-colourings.

Show the **portion of the state space tree generated by a backtracking** algorithm when applied to the following graph:



Indicate what each level represents, and use any additional labels that help clarifying your tree representation.

Part B — 12 points Branch-and-bound

In the following you will find a state space tree for the branch-and-bound method when run on the following example of the **Knapsack** problem:

```
Knapsack weight capacity:  W=4


Profit and weight for items:
```

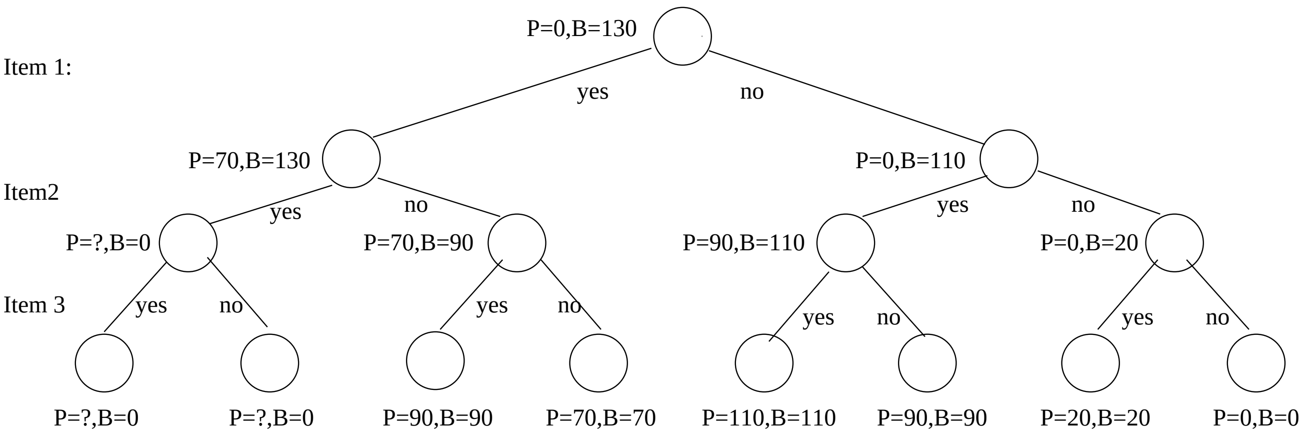
item i	p_i	w_i	p_i/w_i
item 1	70	2	35
item 2	90	3	30
item 3	20	1	20

Recall that in the knapsack problem we are **maximizing** total profit. The state space tree and the upper bounds have already been calculated and are provided below (P represents the total profit of items that have been chosen so far and B represents an upper bound on the profit for the subtree rooted at the node).

A. BREADTH-FIRST branch-and-bound:

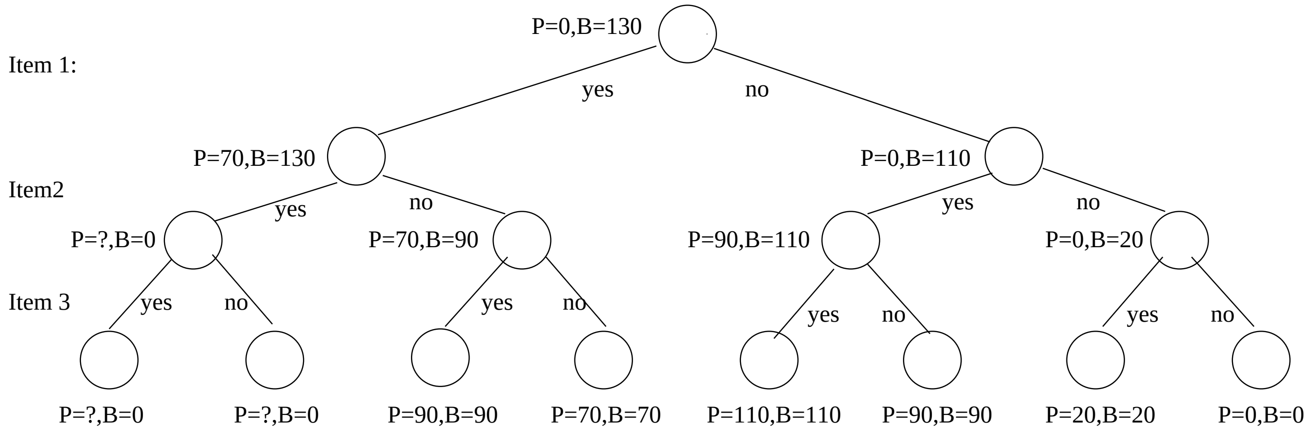
In the space state tree provided below, indicate:

- the order in which nodes are visited in the **breadth-first** branch-and-bound (please, number the nodes in order of visit);
- the nodes that are not visited;
- the nodes that are visited but discarded (cross them out)



B. BEST-FIRST branch-and-bound:

Do exactly the same as in part A, but for a **best-first** branch-and-bound algorithm:



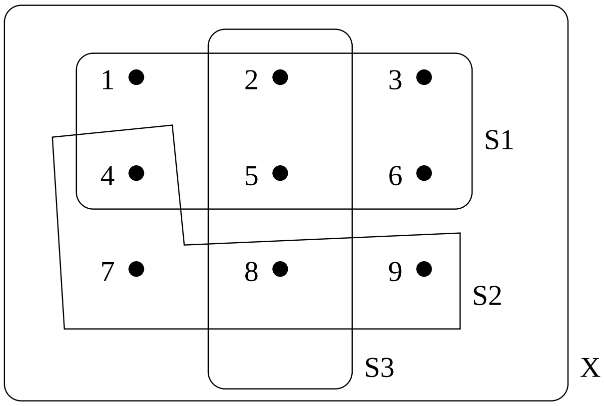
... continued

5 Set covering is NP-complete — 25 points

Recall the **set covering problem**:

We are given a finite set $X = \{x_1, x_2, \ldots, x_n\}$ and a family of subsets $\mathcal{F} = \{S_1, S_2, \ldots, S_m\}$, where $S_i \subseteq X$, and such that every element of X belongs to at least one subset in $\mathcal{F}$. Determine the minimum cardinality subset $\mathcal{C}$ of $\mathcal{F}$ that **covers** X (Note that: we say that $\mathcal{C}$ *covers* X if every element of X belongs to at least one subset of $\mathcal{C}$).

Example:

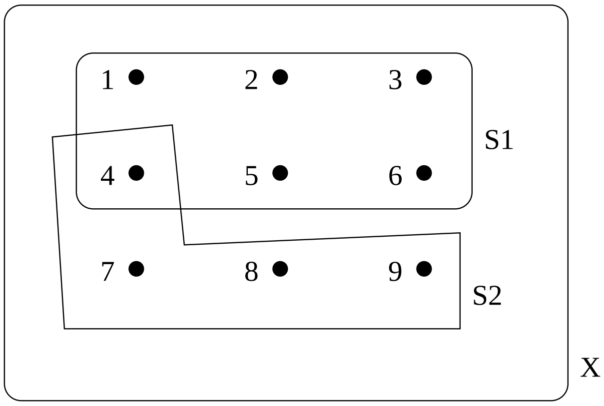


$X = \{1, 2, 3, 4, 5, 6, 7, 8, 9\}$

$\mathcal{F} = \{S_1, S_2, S_3\}$

$S_1 = \{1, 2, 3, 4, 5, 6\}$
$S_2 = \{4, 7, 8, 9\}$
$S_3 = \{2, 5, 8\}$

Solution:
$\mathcal{C} = \{S_1, S_2\}$ covers X and its cardinality equals 2, which is minimum for this example.



The **decision problem** associated to the set covering problem is described below:

Given X, $\mathcal{F}$ and an integer k, is there a subset $\mathcal{C} \subseteq \mathcal{F}$ of cardinality k that covers X ?

The **language** associated to this decision problem is:

$$\text{SET-COVERING} = \{< X, \mathcal{F}, k >: X \text{ is a finite set}, \mathcal{F} \text{ is a family of subsets of } X \text{ and there exists a subset } \mathcal{C} \subseteq \mathcal{F} \text{ of cardinality } k \text{ that covers } X\}$$

... continued

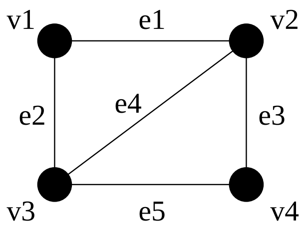
Part A — 8 points Prove that SET-COVERING $\in NP$.

... continued

Part B — 17 points Prove that SET-COVERING is NP-hard, by showing that:

$$\text{VERTEX-COVER} \leq_P \text{SET-COVERING}$$

The idea for the reduction algorithm is illustrated below:

Input (vertex cover instance)	Output (set covering instance)
G :  $k' = 3$	$X = \{e_1, e_2, e_3, e_4, e_5\}$ $\mathcal{F} = \{S_1, S_2, S_3, S_4\}$ $S_1 = \{e_1, e_2\}$, $S_2 = \{e_1, e_3, e_4\}$ $S_3 = \{e_2, e_4, e_5\}$, $S_4 = \{e_3, e_5\}$ $k = 3$

... continued

(this is a blank page)

(this is a blank page)