TDDC17 LE17 HT2023

Course Summary and Exam Questions

Fredrik Heintz

Dept. of Computer Science
Linköping University

fredrik.heintz@liu.se
@FredrikHeintz

Outline:

- **Course summary**
- **Exam questions**

LINKÖPING UNIVERSITY

- Chapter 1, Introduction, pp.1-53.
  - Read, good to understand background and history.
  - Look at slides. Exam questions may be taken explicitly from the slides.
- Chapter 2, Intelligent Agents, pp.54-80.
  - Read, some questions are possible.
  - Draw diagram of different agent types, explain different agent types and pros and cons and application environments. Different task environment types, PEAS.
  - Look at the slides. Exam questions may be taken explicitly from the slides.
- Chapter 3, Solving Problems by Searching, pp.81-100, 102-108, 115-121.
  - Read sec.3.1 - 3.4, 3.4:1-4,6, 3.5:1-3, 3.6:1-4, some questions are possible.
  - Conceptual definitions, Measuring problem solving performance. Pros and cons of different search algorithms.
  - sec. 3.5.2-3: A* search and proof of optimality. Very important.
  - sec. 3.6.1-3: Be aware of heuristic functions, ways to construct them.
  - Problem solving as search, heuristic search.
  - Look at the slides. Exam questions may be taken explicitly from the slides.

- Chapter 4, Search in Complex Environments, pp.128-137.
  - Read sec. 4.1, some questions are possible.
  - Genetic algorithms, simulated annealing hill-climbing, local beam search.
  - Look at the slides. Exam questions may be taken explicitly from the slides.
- Chapter 5, Constraint Satisfaction Problems, pp.164-182.
  - Read sec. 5.1-4, some questions are possible.
  - AC3 algorithm, backtracking search for csps, arc, node and path consistency, forward checking, variable and value ordering, etc.
  - Look at the slides. Exam questions may be taken explicitly from the slides.
- Chapter 6, Adverserial Search and Games, pp. 192-210.
  - Read sec. 6.1-4, some questions are possible.
  - Understand Minimax search, alpha-beta pruning. Heuristic alpha-beta search, Monte Carlo Tree search.
  - Look at the slides. Exam questions may be taken explicitly from the slides.

- Chapter 7, Logical Agents pp.226-260.
  - Read sec 7.1-6, 7.7.1-2, some questions are possible.
  - Assumption: familiarity with propositional logic
  - Basic concepts of logic, simple propositional resolution proof, CNF, SAT-solving,.
  - What are some of the weaknesses with propositional logic when modeling the Wumpus World?
  - Should understand validity, satisfaction, consistency, models, inference, etc.
  - Should understand DPLL, WalkSAT.
  - Should understand Answer Set Programming and be able to compute Answer sets.
  - Look at the slides. Slides are very important for answer sets.
- Chapter 8, First-Order Logic, pp. 261-289
  - Read sec. 8.1-3, some questions are possible.
  - Assumption: familiarity with 1st-order logic. Basics, syntax, semantics.
  - Sec 8.3.4: look at the wumpus world, compare propositional and 1st-order representations.
  - Look at the slides. Exam questions may be taken explicitly from the slides.

- Chapter 9, Inferences in 1st-Order Logic, pp. 298-300.
  - Read sec. 9.1
  - Important to understand propositionalization of 1st-order logic. This is described on the slides and is used for answer set programming.
  - Look at the slides. Exam questions may be taken explicitly from the slides.
- Chapter 11, Automated Planning, pp. 362-398.
  - Read sec. 11.1-11.2.2, 11.3:371-374,11.5-7. Some questions are possible.
  - The basic classical planning problem, the state space and its properties, planning using forward state space search, (relaxation) heuristics, pattern database heuristics, satisficing and optimizing planning, landmark heuristics.
  - Planning under uncertainty. Levels of observability. Probabilistic planning and stochastic systems. Policies and histories. Rewards, discount factors, and (expected) utility. Bellman's principle of optimality. Markov decision processes. Policy iteration.
  - Look at the slides. Slides are very important for automated planning questions. Exam questions may be taken explicitly from the slides.

- Chapter 12, Quantifying Uncertainty, pp. 403-422.
  - Read sec. 12.1-12.6.
  - This chapter provides the basis for understanding what probabilistic reasoning is about. It is a pre-requisite for the next chapter. You should be familiar with the material, in particular Bayes' rule, naive bayes, marginalization, conditionalization, product and chain rule., etc.
  - Look at the slides. Exam questions may be taken explicitly from the slides.
- Chapter 13, Probabilistic Reasoning, pp. 430-438, 445.
  - Read sec 13.1, 13.2:1, 13.3:intro. some questions are possible.
  - Be able to answer questions about and work with Bayesian Networks.
  - Look at the slides. Exam questions may be taken explicitly from the slides.
- Chapter 19, Learning from Example, pp. 669-90, 694-704
  - Read sec. 19.1-4, 19.6
  - Look at the slides! Exam questions may be taken explicitly from the slides.
  - Understand supervised learning, training, classification, regression, decision trees, linear models, neural networks, deep learning, overfitting, curse of dimensionality

- Chapter 22, Deep Learning, pp. 801-819, 823-26, 833-35
  - Read sec. 22.1-4, 22.6, 22.8
  - Look at the slides! Exam questions may be taken explicitly from the slides.
  - Neural Networks, Gradients, loss functions, convolutional NNs, recurrent NNs, unsupervised and transfer learning.
- Chapter 23, Reinforcement Learning, pp. 840-858
  - Read sec. 23.1-3, 23.4.1-3, (also ch 16: 16.1-2 for background)
  - Understand the Q-learning algorithm and concepts learnt from the lab.
  - Review Q-learning example from lecture
  - Look at the slides! Exam questions may be taken explicitly from the slides.
  - Understand reinforcement learning, utility, policy, Q-learning, exploration, curse of dimensionality

Some Exam Questions

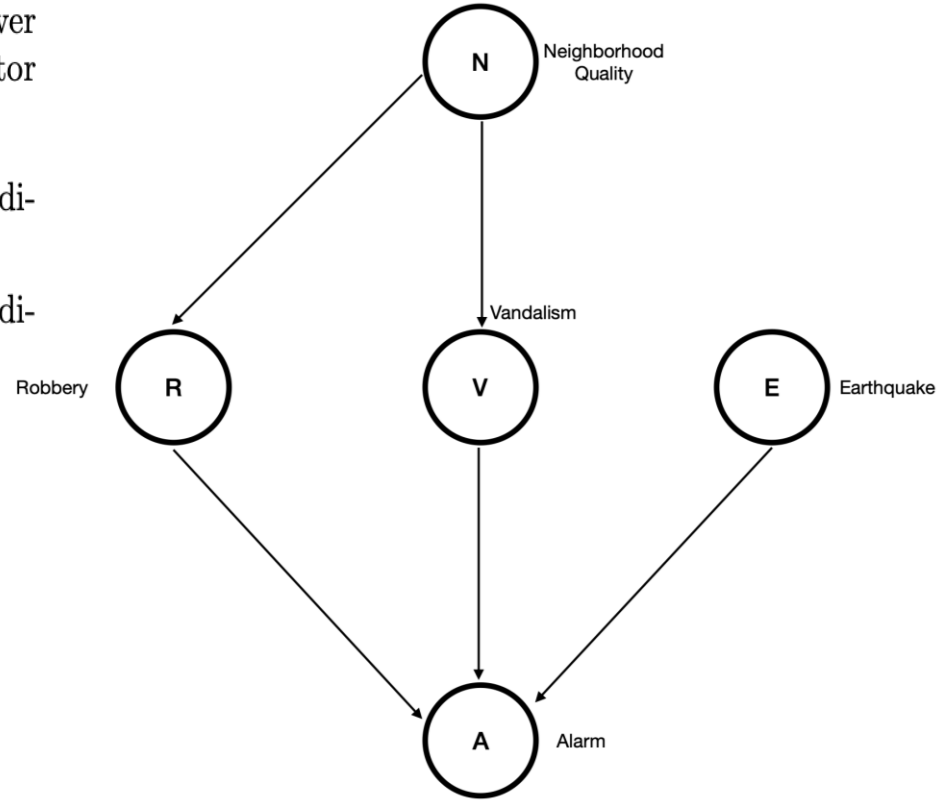
6. Use the Bayesian network in Figure 2 together with the conditional probability tables below to answer the following questions. If you do not have a hand-held calculator with you, make sure you set up the solution to the problems appropriately for partial credit.

   (a) Write the formula for the full joint probability distribution $P(A, R, V, E, N)$ in terms of (conditional) probabilities using the chain rule. **[1p]**

   (b) Write the formula for the full joint probability distribution $P(A, R, V, E, N)$ in terms of (conditional) probabilities derived from the bayesian network below. **[1p]**

   (c) What is the probability, $P(a, r, \neg v, e, N = high)$? **[1p]**

   (d) What is the probability, $P(a \mid \neg e, v)$? **[2p]**



A	R	V	E	$P(A \mid R, V, E)$
T	T	T	T	0.98
F	T	T	T	0.02
T	T	T	F	0.75
F	T	T	F	0.25
T	T	F	T	0.65
F	T	F	T	0.35
T	T	F	F	0.70
F	T	F	F	0.30
T	F	F	F	0.05
F	F	F	F	0.95
T	F	F	T	0.10
F	F	F	T	0.90
T	F	T	F	0.55
F	F	T	F	0.45
T	F	T	T	0.35
F	F	T	T	0.65

R	N	$P(R \mid N)$
T	high	.2
F	high	.8
T	low	.75
F	low	.25

V	N	$P(V \mid N)$
T	high	.2
F	high	.8
T	low	.75
F	low	.25

E	P(E)
T	.05
F	.95

N	P(N)
high	.75
low	.25

(a) Write the formula for the full joint probability distribution $P(A, R, V, E, N)$ in terms of (conditional) probabilities using the chain rule. **[1p]**

a.) $$\mathbf{P}(A, R, V, E, N) = \mathbf{P}(A \mid R, V, E, N)\mathbf{P}(R \mid V, E, N)\mathbf{P}(V \mid E, N), \mathbf{P}(E \mid N)\mathbf{P}(N)$$

(b) Write the formula for the full joint probability distribution $P(A, R, V, E, N)$ in terms of (conditional) probabilities derived from the bayesian network below. **[1p]**

b.) Choose a topological order for nodes:
(N, R, V, E, A)

$$\mathbf{P}(N)\mathbf{P}(R \mid N)\mathbf{P}(V \mid N)\mathbf{P}(E)\mathbf{P}(A \mid R, V, E)$$

(c) What is the probability, $P(a, r, \neg v, e, N = high)$? **[1p]**

c.) $$P(a, r, \neg v, e, N = high) =$$

$$P(N = high)P(r \mid N = high)P(\neg v \mid N = high)P(e)P(a \mid r, \neg v, e)$$

$$= 0.75 * 0.2 * 0.8 * 0.05 * 0.65 = 0.0039$$

(d) What is the probability, $P(a \mid \neg e, v)$? **[2p]**

d). An Inference Example

$$\mathbf{P}(X \mid \mathbf{e}) = \alpha * \mathbf{P}(X, \mathbf{e}) = \alpha * \sum_{\mathbf{y}} \mathbf{P}(X, \mathbf{e}, \mathbf{y})$$

$X = \{Alarm\}$
$\mathbf{E} = \{Vandalism, Earthquake\}$
$\mathbf{e} = \{\neg earthquake, vandalism\}$
$\mathbf{Y} = \{Robbery, Nquality\}$

Query: $P(alarm \mid \neg earthquake, vandalism)$

$$P(alarm \mid \neg earthquake, vandalism) = \frac{P(alarm, \neg earthquake, vandalism)}{P(\neg earthquake, vandalism)}$$

$$= \alpha P(alarm, \neg earthquake, vandalism)$$

$$= \alpha \sum_{\mathbf{r},\mathbf{n}} P(alarm, \neg earthquake, vandalism, \mathbf{r}, \mathbf{n})$$

$$= \alpha[P(a, \neg e, v, r, N = h) + P(a, \neg e, v, r, N = l) + P(a, \neg e, v, \neg r, N = h) + P(a, \neg e, v, \neg r, N = l)]$$

where $\alpha = \frac{1}{P(\neg e, v)} = \frac{1}{\sum_{\mathbf{a},\mathbf{r},\mathbf{n}} P(\neg e, v, \mathbf{a}, \mathbf{r}, \mathbf{n})}$

$$= \alpha[P(a, \neg e, v, r, N = h) + P(a, \neg e, v, r, N = l) + P(a, \neg e, v, \neg r, N = h) + P(a, \neg e, v, \neg r, N = l)]$$

$$\text{where } \alpha = \frac{1}{P(\neg e, v)} = \frac{1}{\sum_{\mathbf{a},\mathbf{r},\mathbf{n}} P(\neg e, v, \mathbf{a}, \mathbf{r}, \mathbf{n})}$$

We know how to compute each probability:

$$P(a, \neg e, v, r, N = h)$$

$$= P(N = h)P(r \mid N = high)P(v \mid N = high)P(\neg e)P(a \mid r, v, \neg e)$$

$$= 0.75 * 0.2 * 0.2 * 0.95 * 0.75 = 0.021375$$

And

$$\sum_{\mathbf{a},\mathbf{r},\mathbf{n}} P(\neg e, v, \mathbf{a}, \mathbf{r}, \mathbf{n}) = P(\neg e, v, a, r, N = h) + \cdots + P(\neg e, v, \neg a, \neg r, N = l)$$

To avoid computing α in this way, compute $\mathbf{P}(A \mid \neg e, v) = \alpha \mathbf{P}(A \mid \neg e, v)$ instead where,

$$\alpha = \frac{1}{P(a, \neg e, v) + P(\neg a, \neg e, v)}$$

Question:DPLL [Total points: **[6p]**]

The following questions pertain to the extended Davis-Putnam algorithm (DPLL) considered in the course book. DPLL takes as input a formula in propositional logic, transforms it to its equivalent conjunctive normal form (CNF) representation and determines whether the formula is satisfiable or not. The questions pertain to the different heuristics used in DPLL.

(When asked to "complete the table ..." in the questions below, write your own complete table in the answer page. Do not fill in these tables on the exam page.)

a. Complete the table below by applying the Splitting rule to variable B and CNF: **[1p]**

$\alpha = (A \vee B \vee C \vee D) \wedge (\neg A \vee B \vee \neg C) \wedge (\neg B \vee \neg C \vee D) \wedge (A \vee \neg B \vee C \vee D)$
$\wedge (B \vee \neg C \vee \neg D) \wedge (\neg A \vee \neg C \vee D) \wedge (\neg A \vee \neg D) \wedge (A \vee \neg B)$

Table 1:

Heuristic	Model_1	Simplified CNF_1	Model_2	Simplified CNF_2
Initial call	{}	α	-	-
$SR[B]$	$\{B : True\}$		$\{B : False\}$	

b. Complete the table below by applying the Unit Clause Heuristic (UCH) as long as possible to the CNF: **[1.5p]**

$\alpha = \neg P \wedge (\neg Q \vee R \vee S) \wedge (\neg P \vee Q) \wedge (P \vee \neg R)$

Use one row for each call and state the variable the heuristic is being applied to in the 1st column, the extension to the model in the 2nd column and the resulting formula in the 3rd column. Add as many additional rows as required.

Table 2:

Heuristic	Model	Simplified CNF
Initial call	{}	α
$UCH[-]$		
$UCH[-]$		
$UCH[-]$		

c. Complete the table below by applying the Pure Symbol Heuristic (PSH) as long as possible to the CNF: **[1.5p]**

$\alpha = (\neg P \vee Q) \wedge (R \vee \neg Q) \wedge (M \vee \neg N) \wedge (N \vee \neg M) \wedge (Q \vee N)$

Use one row for each call and state the variable the heuristic is being applied to in the 1st column, the extension to the model in the 2nd column and the resulting formula in the 3rd column. Add as many additional rows as required.

Table 3:

Heuristic	Model	Simplified CNF
Initial call	{}	α
$PSH[-]$		
$PSH[-]$		
$PSH[-]$		

d. Suppose we had three formulas in propositional logic, $F1, F2$ and $F3$. Explain how one would show whether $F1 \wedge F2 \models F3$ using the DPLL algorithm. In doing this, it is important to show the formal relation between $\models$ and satisfaction. **[2p]**

a. Complete the table below by applying the Splitting rule to variable B and CNF: **[1p]**

$$\alpha = (A \vee B \vee C \vee D) \wedge (\neg A \vee B \vee \neg C) \wedge (\neg B \vee \neg C \vee D) \wedge (A \vee \neg B \vee C \vee D) \wedge (B \vee \neg C \vee \neg D) \wedge (\neg A \vee \neg C \vee D) \wedge (\neg A \vee \neg D) \wedge (A \vee \neg B)$$

Table 1:

Heuristic	Model_1	Simplified CNF_1	Model_2	Simplified CNF_2
Initial call	$\{\}$	α	-	-
$SR[B]$	$\{B : True\}$		$\{B : False\}$	

a. -

$$\alpha = (A \vee B \vee C \vee D) \wedge (\neg A \vee B \vee \neg C) \wedge (\neg B \vee \neg C \vee D) \wedge (A \vee \neg B \vee C \vee D) \wedge (B \vee \neg C \vee \neg D) \wedge (\neg A \vee \neg C \vee D) \wedge (\neg A \vee \neg D) \wedge (A \vee \neg B)$$

Table 4:

Heuristic	Model_1	Simplified CNF_1	Model_2	Simplified CNF_2
Initial call	$\{\}$	α	-	-
$SR[B]$	$\{B : True\}$	β	$\{B : False\}$	γ

$$\beta = (\neg C \vee D) \wedge (A \vee C \vee D) \wedge (\neg A \vee \neg C \vee D) \wedge (\neg A \vee \neg D) \wedge (A)$$

$$\gamma = (A \vee C \vee D) \wedge (\neg A \vee \neg C) \wedge (\neg C \vee \neg D) \wedge (\neg A \vee \neg C \vee D) \wedge (\neg A \vee \neg D)$$

b. Complete the table below by applying the Unit Clause Heuristic (UCH) as long as possible to the CNF: **[1.5p]**

$$\alpha = \neg P \wedge (\neg Q \vee R \vee S) \wedge (\neg P \vee Q) \wedge (P \vee \neg R)$$

Use one row for each call and state the variable the heuristic is being applied to in the 1st column, the extension to the model in the 2nd column and the resulting formula in the 3rd column. Add as many additional rows as required.

Table 2:

Heuristic	Model	Simplified CNF
Initial call	{}	α
$UCH[-]$		
$UCH[-]$		
$UCH[-]$		

b. -

$$\alpha = \neg P \wedge (\neg Q \vee R \vee S) \wedge (\neg P \vee Q) \wedge (P \vee \neg R)$$

Table 5:

Heuristic	Model	Simplified CNF
Initial call	{}	α
$UCH[\neg P]$	$\{P : False\}$	$(\neg Q \vee R \vee S) \wedge (\neg R)$
$UCH[\neg R]$	$\{P : False, R : False\}$	$(\neg Q \vee S)$

c. Complete the table below by applying the Pure Symbol Heuristic (PSH) as long as possible to the CNF: **[1.5p]**

$$\alpha = (\neg P \vee Q) \wedge (R \vee \neg Q) \wedge (M \vee \neg N) \wedge (N \vee \neg M) \wedge (Q \vee N)$$

Use one row for each call and state the variable the heuristic is being applied to in the 1st column, the extension to the model in the 2nd column and the resulting formula in the 3rd column. Add as many additional rows as required.

Table 3:

Heuristic	Model	Simplified CNF
Initial call	{}	α
$PSH[-]$		
$PSH[-]$		
$PSH[-]$		

c. -

$$\alpha = (\neg P \vee Q) \wedge (R \vee \neg Q) \wedge (M \vee \neg N) \wedge (N \vee \neg M) \wedge (Q \vee N)$$

Table 6:

Heuristic	Model	Simplified CNF
Initial call	{}	α
$PSH[P]$	$\{P : False\}$	$(R \vee \neg Q) \wedge (M \vee \neg N) \wedge (N \vee \neg M) \wedge (Q \vee N)$
$PSH[R]$	$\{P : False, R : True\}$	$(M \vee \neg N) \wedge (N \vee \neg M) \wedge (Q \vee N)$
$PSH[Q]$	$\{P : False, R : True, Q : True\}$	$(M \vee \neg N) \wedge (N \vee \neg M)$

Initially, P, R are pure symbols. Choose one: (P)
R is a pure symbol. Choose one: (R)
Q is a pure symbol. Choose one: (Q)

d. Suppose we had three formulas in propositional logic, $F1, F2$ and $F3$. Explain how one would show whether $F1 \wedge F2 \models F3$ using the DPLL algorithm. In doing this, it is important to show the formal relation between $\models$ and satisfaction. **[2p]**

d. The answer uses the Deduction Theorem and the relationship between satisfiability and validity.

1. The deduction theorem states that If $\Gamma \models \alpha$ then $\models \Gamma \rightarrow \alpha$.
2. $\Gamma \rightarrow \alpha$ is valid iff $\neg(\Gamma \rightarrow \alpha)$ which is the same as saying that $\Gamma \wedge \neg\alpha$ is unsatisfiable.
3. Let β be $F_1 \wedge F_2 \wedge \neg F_3$ in CNF.
4. If DPLL-Satisfiable?(β) is true then $F_1 \wedge F_2 \models F_3$ is false
5. If DPLL-Satisfiable?(β) is false then $F_1 \wedge F_2 \models F_3$ is true

Answer set question [propositionalization or grounding]:

Assume the following signature:

$\Sigma = \langle \mathcal{O} = \{\text{jim}, \text{sarah}\}, \mathcal{P} = \{\text{sameNeighborhood}, \text{strangers}, \text{person}, \text{criminal}, \text{goodNeighbor}\}, \mathcal{V} = \{X, Y\}\rangle$

Propositionalize (ground) the following answer set program Π:

r1: $\text{person}(X)$.

r2: $\text{sameNeighborhood}(X, X)$.

r3: $\text{strangers}(X, Y) \leftarrow not\ \text{sameNeighborhood}(X, Y), \text{person}(X), \text{person}(Y)$.

r4: $\text{goodNeighbor}(X) \leftarrow not\ \text{criminal}(X), \text{person}(X)$.

$Prop(\Pi)$:

r1: $\text{person}(\text{jim})$.

r2: $\text{person}(\text{sarah})$.

r3: $\text{sameNeighborhood}(\text{jim}, \text{jim})$.

r4: $\text{sameNeighborhood}(\text{sarah}, \text{sarah})$.

r5: $\text{strangers}(\text{jim}, \text{jim}) \leftarrow not\ \text{sameNeighborhood}(\text{jim}, \text{jim}), \text{person}(\text{jim}), \text{person}(\text{jim})$.

r6 $\text{strangers}(\text{sarah}, \text{sarah}) \leftarrow not\ \text{sameNeighborhood}(\text{sarah}, \text{sarah}), \text{person}(\text{sarah}), \text{person}(\text{sarah})$.

r7: $\text{strangers}(\text{jim}, \text{sarah}) \leftarrow not\ \text{sameNeighborhood}(\text{jim}, \text{sarah}), \text{person}(\text{jim}), \text{person}(\text{sarah})$.

r8: $\text{strangers}(\text{sarah}, \text{jim}) \leftarrow not\ \text{sameNeighborhood}(\text{sarah}, \text{jim}), \text{person}(\text{sarah}), \text{person}(\text{jim})$.

r9: $\text{goodNeighbor}(\text{jim}) \leftarrow not\ \text{criminal}(\text{jim}), \text{person}(\text{jim})$.

r10: $\text{goodNeighbor}(\text{sarah}) \leftarrow not\ \text{criminal}(\text{sarah}), \text{person}(\text{sarah})$.

Question:Answer Sets [Total points: **[8p]**]

The following questions pertain to Answer Set Programming.

Assume the answer set program below Π_1, has the following signature:

$\langle \mathcal{O} = \{a, b\}, \mathcal{P} = \{p, q, r, s\}, \mathcal{V} = \{X\} \rangle$.

Given the positive answer set program Π_1, consisting of the following rules:

r1: $s(a)$.

r2: $p(X) \leftarrow not\ q(X), s(X)$.

r3: $q(X) \leftarrow not\ p(X)$.

r4: $r(X) \leftarrow p(X)$.

r5: $r(X) \leftarrow q(X)$.

a. Propositionalize the answer set program Π_1. Refer to the propositionalized version of Π_1 as Π_p. **[1p]**

b. What is the cardinality of the possible answer sets for Π_p? Explain why. **[1p]**

Given the following *possible* answer sets for Π_p :

$$S_1 = \{s(a), q(b), r(b), q(a), r(a)\}$$
$$S_2 = \{s(a), r(b), q(a), r(a), p(a), p(b)\}$$
$$S_3 = \{s(a), q(b), r(b), p(a), r(a))\}$$

c. For each S_i, provide the reduct, $\Pi_p^{S_i}$ for Π_p. **[1.5p]**

d. Generate $Cn(\Pi_p^{S_i})$ for each reduct $\Pi_1^{S_i}$ of Π_p. **[1.5p]**

e. Which of the three answer sets S_1, S_2, S_3 are *actual* answer sets for Π_p? (Explain why)**[1p]**

f. The consequence relation, $\models$, for classical logic is said to be monotonic. What does this mean? (Be precise and succinct in your explanation) **[1p]**

g. Why is Answer Set Programming considered to be a nonmonotonic reasoning formalism (Be precise and use an example)? **[1p]**

Assume the answer set program below Π_1, has the following signature:
$\langle \mathcal{O} = \{a, b\}, \mathcal{P} = \{p, q, r, s\}, \mathcal{V} = \{X\} \rangle$.
Given the positive answer set program Π_1, consisting of the following rules:

r1: $s(a).$
r2: $p(X) \leftarrow not\ q(X), s(X).$
r3: $q(X) \leftarrow not\ p(X).$
r4: $r(X) \leftarrow p(X).$
r5: $r(X) \leftarrow q(X).$

a. Propositionalize the answer set program Π_1. Refer to the propositionalized version of Π_1 as Π_p. **[1p]**

a. -

r1: $s(a).$
r2: $p(a) \leftarrow not\ q(a), s(a).$
r3: $q(a) \leftarrow not\ p(a).$
r4: $r(a) \leftarrow p(a).$
r5: $r(a) \leftarrow q(a).$
r6: $p(b) \leftarrow not\ q(b), s(b).$
r7: $q(b) \leftarrow not\ p(b).$
r8: $r(b) \leftarrow p(b).$
r9: $r(b) \leftarrow q(b).$

b. What is the cardinality of the possible answer sets for Π_p? Explain why. **[1p]**

b. $2^7 = 128$. Seven unique literals in the heads of the rules.

Given the following *possible* answer sets for Π_p :

$$S_1 = \{s(a), q(b), r(b), q(a), r(a)\}$$
$$S_2 = \{s(a), r(b), q(a), r(a), p(a), p(b)\}$$
$$S_3 = \{s(a), q(b), r(b), p(a), r(a))\}$$

a. -

r1: $s(a).$
r2: $p(a) \leftarrow not\ q(a), s(a).$
r3: $q(a) \leftarrow not\ p(a).$
r4: $r(a) \leftarrow p(a).$
r5: $r(a) \leftarrow q(a).$
r6: $p(b) \leftarrow not\ q(b), s(b).$
r7: $q(b) \leftarrow not\ p(b).$
r8: $r(b) \leftarrow p(b).$
r9: $r(b) \leftarrow q(b).$

c. For each S_i, provide the reduct, $\Pi_p^{S_i}$ for Π_p. **[1.5p]**

c. -

$$S_1 = \{s(a), q(b), r(b), q(a), r(a)\}$$
$$S_2 = \{s(a), r(b), q(a), r(a), p(a), p(b)\}$$
$$S_3 = \{s(a), q(b), r(b), p(a), r(a))\}$$

Rule	$\Pi_1^{S_1}$
r1:	$s(a).$
r2:	*delete*
r3:	$q(a).$
r4:	$r(a) \leftarrow p(a).$
r5:	$r(a) \leftarrow q(a).$
r6:	*delete*
r7:	$q(b).$
r8:	$r(b) \leftarrow p(b).$
r9:	$r(b) \leftarrow q(b).$

Rule	$\Pi_1^{S_2}$
r1:	$s(a).$
r2:	*delete.*
r3:	*delete*
r4:	$r(a) \leftarrow p(a).$
r5:	$r(a) \leftarrow q(a).$
r6:	$p(b) \leftarrow s(b).$
r7:	*delete*
r8:	$r(b) \leftarrow p(b).$
r9:	$r(b) \leftarrow q(b).$

Rule	$\Pi_1^{S_3}$
r1:	$s(a).$
r2:	$p(a) \leftarrow s(a).$
r3:	*delete*
r4:	$r(a) \leftarrow p(a).$
r5:	$r(a) \leftarrow q(a).$
r6:	*delete*
r7:	$q(b).$
r8:	$r(b) \leftarrow p(b).$
r9:	$r(b) \leftarrow q(b).$

c. -

$S_1 = \{s(a), q(b), r(b), q(a), r(a)\}$
$S_2 = \{s(a), r(b), q(a), r(a), p(a), p(b)\}$
$S_3 = \{s(a), q(b), r(b), p(a), r(a))\}$

Rule	$\Pi_1^{S_1}$
r1:	$s(a).$
r2:	*delete*
r3:	$q(a).$
r4:	$r(a) \leftarrow p(a).$
r5:	$r(a) \leftarrow q(a).$
r6:	*delete*
r7:	$q(b).$
r8:	$r(b) \leftarrow p(b).$
r9:	$r(b) \leftarrow q(b).$

Rule	$\Pi_1^{S_2}$
r1:	$s(a).$
r2:	*delete.*
r3:	*delete*
r4:	$r(a) \leftarrow p(a).$
r5:	$r(a) \leftarrow q(a).$
r6:	$p(b) \leftarrow s(b).$
r7:	*delete*
r8:	$r(b) \leftarrow p(b).$
r9:	$r(b) \leftarrow q(b).$

Rule	$\Pi_1^{S_3}$
r1:	$s(a).$
r2:	$p(a) \leftarrow s(a).$
r3:	*delete*
r4:	$r(a) \leftarrow p(a).$
r5:	$r(a) \leftarrow q(a).$
r6:	*delete*
r7:	$q(b).$
r8:	$r(b) \leftarrow p(b).$
r9:	$r(b) \leftarrow q(b).$

d. Generate $Cn(\Pi_p^{S_i})$ for each reduct $\Pi_1^{S_i}$ of Π_p. **[1.5p]**

d. -

- $Cn(\Pi_p^{S_1}) : \{\}, \{s(a), q(a), q(b)\}, \{s(a), q(a), q(b), r(a), r(b)\}$
- $Cn(\Pi_p^{S_2}) : \{\}, \{s(a)\}$
- $Cn(\Pi_p^{S_3}) : \{\}, \{s(a), q(b)\}, \{s(a), q(b), p(a), r(b)\}, \{s(a), q(b), p(a), r(b), r(a)\}$

e. Which of the three answer sets S_1, S_2, S_3 are *actual* answer sets for Π_p? (Explain why)**[1p]**

e. $Cn(\Pi_p^{S_1}) = S_1$, $Cn(\Pi_p^{S_2}) \neq S_2$, $Cn(\Pi_p^{S_3}) = S_3$. S_1 and S_3 are answer sets.

f. The consequence relation, $\models$, for classical logic is said to be monotonic. What does this mean? (Be precise and succinct in your explanation) **[1p]**

f. Intuitively, this means that once something is proven relative to a theory, if one extends the theory, anything previously proven will still be proven in the extended theory. This is a form of locality. $\Gamma \models \alpha$ implies that $\Gamma \cup \Gamma' \models \alpha$.

g. Why is Answer Set Programming considered to be a nonmonotonic reasoning formalism (Be precise and use an example)? **[1p]**

g. ASP is considered to be a nonmonotonic formalism, because something inferable from a program Π, may not be inferable when the program is extended with new rules. By inferable, we mean true in all answer sets for the program Π. As an example $\Pi = \{b \leftarrow not\ a\}$ entails b since it has one answer set $\{b\}$. If Π is extended with the rule $a \leftarrow$, then b is no longer entailed since the answer set for the new program is $\{a\}$.

Question:Adversarial Search [Total points: **[5p]**]

The following question pertains to adversarial search. Consider the game tree in the figure below in which the leaf states show heuristic values and where all heuristic values are from the MAX players point of view. Assume search is in the left to right direction.

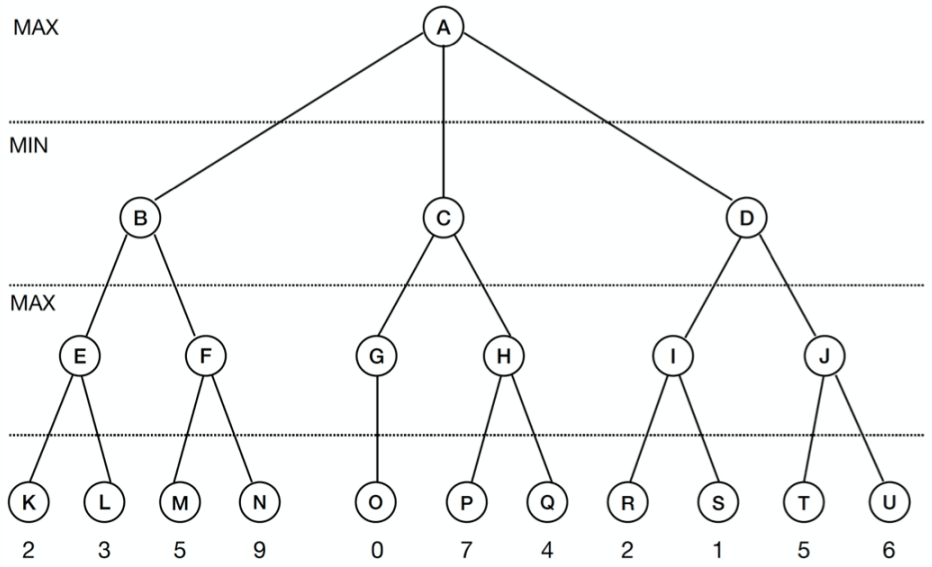


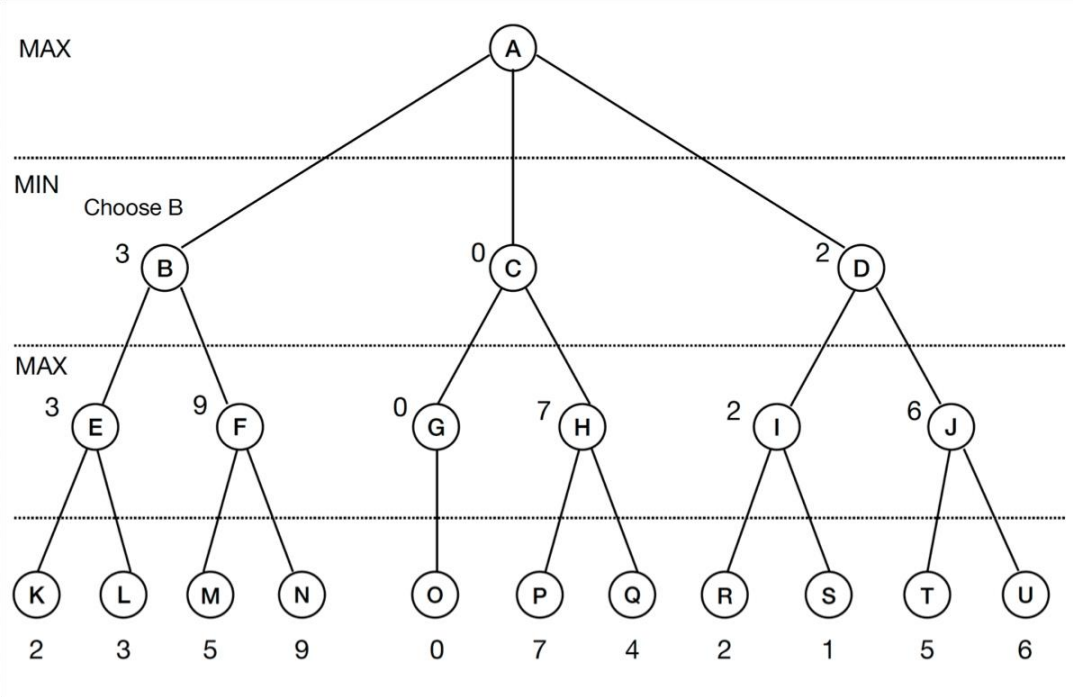
Figure 2: A Minimax Game Tree

a. Apply the MinMax algorithm to the game tree in the figure and state what move the first player (maximizer) would make. Provide a table with heuristic values for each node, or label the tree using your own uploaded image. **[2p]**

b. In the game tree above, what nodes would not need to be examined using the alpha-beta procedure? Justify your answer in terms of the relevant α/β values in the nodes of the tree and why certain branches would be cutoff based on this evaluation. Provide the explanation textually or using a combination of text and an uploaded image. **[3p]**

a. Apply the MinMax algorithm to the game tree in the figure and state what move the first player (maximizer) would make. Provide a table with heuristic values for each node, or label the tree using your own uploaded image. **[2p]**

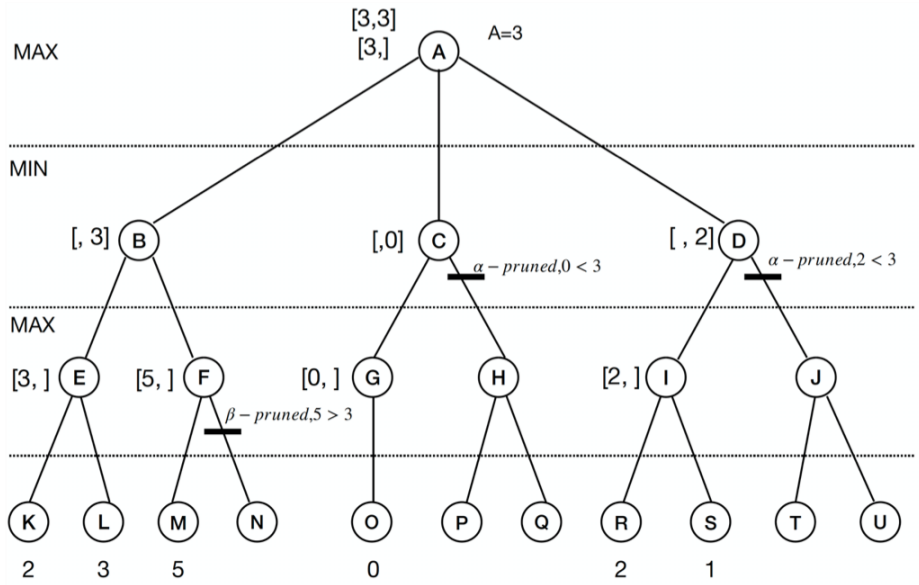
a. MAX chooses B.



b. In the game tree above, what nodes would not need to be examined using the alpha-beta procedure? Justify your answer in terms of the relevant α/β values in the nodes of the tree and why certain branches would be cutoff based on this evaluation. Provide the explanation textually or using a combination of text and an uploaded image. **[3p]**

b.

- α-pruning: Search can be stopped below any MIN node having a beta value less than or equal to the alpha value of any of its MAX node ancestors.
- β-pruning: Search can be stopped below any MAX node having a alpha value greater than or equal to the beta value of any of its MIN node ancestors.



Question:Constraints [Total points: **[4p]**]

The following question pertains to constraint satisfaction problems.

An initial constraint graph is shown in Figure 2 for a graph coloring problem consisting of three variables, V_1, V_2, V_3, with their initial values depicted. Note that each edge represents two directed edges, one in each direction.

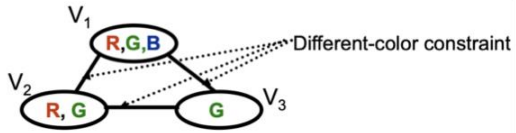


Figure 2: Graph Coloring Constraint Graph

The table below in Figure 3 shows the first 2 steps when applying the AC-3 algorithm to the constraint graph in Figure 2. For edge $V_1 \rightarrow V_2$, no revision of V_1's domain is necessary. For edge $V_1 \rightarrow V_3$, revision of V_1's domain is necessary and the value G is removed from V_1's domain. This is labeled as $V_1(G)$ in the second column in the table below.

Arc examined	Value deleted
$V_1 - V_2$	none
$V_1 - V_3$	V_1(G)

Figure 3: AC-3 Table

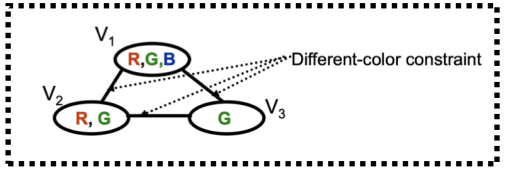
(a) Apply the AC-3 algorithm (4th ed: p.187, 3rd ed: p. 209, note pages may vary somewhat for Int'l editions) to the constraint graph in Figure 2 , by completing the table shown in Figure 3. (Note that there will be more rows in the table than depicted here.) **[2p]**

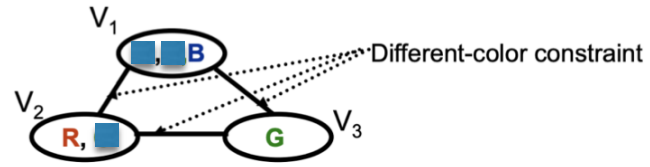
(b) Show the final result of the application of the AC-3 algorithm to the constraint graph in Figure 2, by providing the resulting arc-consistent constraint graph. Using this, provide a solution to the original graph coloring problem.**[1p]**

(c) Provide a new constraint graph for the graph coloring problem in general, that is arc-consistent, but has no solutions. (Note, this has nothing to do with the 1st two questions (a,b). Simply show a constraint graph in this domain with the property of being arc-consistent with no solutions to the original problem.) **[1p]**

(a) Apply the AC-3 algorithm (4th ed: p.187, 3rd ed: p. 209, note pages may vary somewhat for Int'l editions) to the constraint graph in Figure 2 , by completing the table shown in Figure 3. (Note that there will be more rows in the table than depicted here.) **[2p]**

All arcs
$\overrightarrow{V_1 \rightarrow V_2}$
$\overrightarrow{V_1 \rightarrow V_3}$
$\overrightarrow{V_2 \rightarrow V_1}$
$\overrightarrow{V_2 \rightarrow V_3}$
$\overrightarrow{V_3 \rightarrow V_1}$
$\overrightarrow{V_3 \rightarrow V_2}$

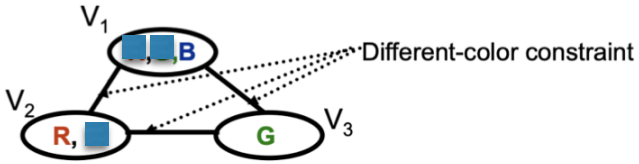




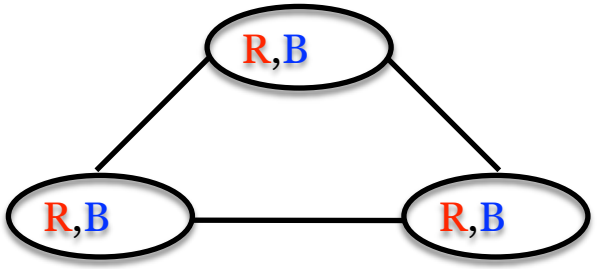
New arcs
$\overrightarrow{V_1 \rightarrow V_2}$
$\overrightarrow{V_2 \rightarrow V_1}$
$\overrightarrow{V_3 \rightarrow V_1}$

Arc Examined	Values Deleted
$V_1 \rightarrow V_2$	None
$V_1 \rightarrow V_3$	$V_1(G)$
$V_2 \rightarrow V_1$	None
$V_2 \rightarrow V_3$	$V_2(G)$
$V_3 \rightarrow V_1$	None
$V_3 \rightarrow V_2$	None
$V_1 \rightarrow V_2$	$V_1(R)$
$V_2 \rightarrow V_1$	None
$V_3 \rightarrow V_1$	None

(b) Show the final result of the application of the AC-3 algorithm to the constraint graph in Figure 2, by providing the resulting arc-consistent constraint graph. Using this, provide a solution to the original graph coloring problem.[**1p**]



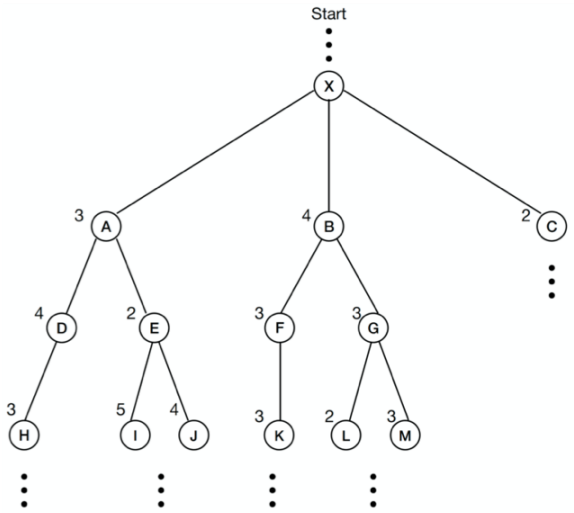
(c) Provide a new constraint graph for the graph coloring problem in general, that is arc-consistent, but has no solutions. (Note, this has nothing to do with the 1st two questions (a,b). Simply show a constraint graph in this domain with the property of being arc-consistent with no solutions to the original problem.) [**1p**]



Question:Search

The following questions pertain to search.

1. Which of the following are true and which are false? Explain your answers.
   - **a.** Depth-first search always expands at least as many nodes as A^* search with an admissible heuristic. **[1p]**
   - **b.** $h(n) = 0$ is an admissible heuristic for the 8-puzzle. **[1p]**
   - **c.** Breadth-first search is complete even if zero step costs are allowed. **[1p]**
2. Consider a state space where the start state is k=1 and each state has two successors: numbers $3k$ and $3k + 1$.
   - **a.** Draw the portion of the state space for states beginning with 1 and ending with 40. **[1p]**
   - **b.** Suppose the goal state is 36. List the order in the portion of the state space specified in (a) in which the nodes will be visited for breadth-first search, depth-first search, depth-limited search with limit 2, and iterative deepening search. **[2p]**
3. Hill Climbing is a local search algorithm that has the advantage of using little memory and can sometimes find solutions in large or infinite state spaces when systematic search algorithms can not be used. But hill-climbing has its problems too, such as getting stuck at local maximums and local flat maximums. Using the diagram below, which depicts a search space where the algorithm has arrived at X, explain what this problem is. Describe an extension to Hill-Climbing that might improve such situations. **[2p]**

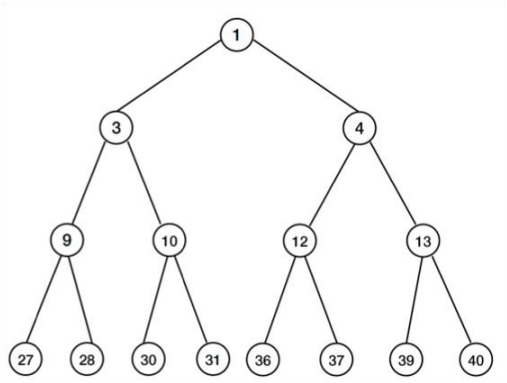


1. Which of the following are true and which are false? Explain your answers.
   a. Depth-first search always expands at least as many nodes as A^* search with an admissible heuristic. **[1p]**
   b. $h(n) = 0$ is an admissible heuristic for the 8-puzzle. **[1p]**
   c. Breadth-first search is complete even if zero step costs are allowed. **[1p]**

1. -
   a. Depth-first search always expands at least as many nodes as A^* search with an admissible heuristic.
   False: a lucky DFS might expand exactly d nodes to reach the goal. A^* largely dominates any graph-search algorithm that is guaranteed to find optimal solutions.
   b. $h(n] = 0$ is an admissible heuristic for the 8-puzzle.
   True: h(n) = 0 is always an admissible heuristic, since costs are nonnegative.
   c. Breadth-first search is complete even if zero step costs are allowed.
   True: depth of the solution matters for breadth-first search, not cost.

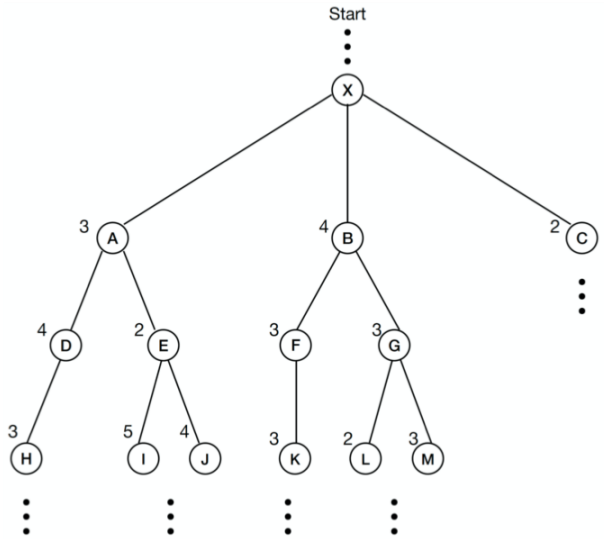
2. Consider a state space where the start state is k=1 and each state has two successors: numbers $3k$ and $3k + 1$.
   a. Draw the portion of the state space for states beginning with 1 and ending with 40. **[1p]**
   b. Suppose the goal state is 36. List the order in the portion of the state space specified in (a) in which the nodes will be visited for breadth-first search, depth-first search, depth-limited search with limit 2, and iterative deepening search. **[2p]**

a. -



b.
- Breadth-First: 1,3,4,9,10,12,13,27,28,30,31,36
- Depth-First: 1,3,9,27,28,10,30,31,4,12,36
- Depth-Limited Search[2]: 1,3,9,10,4,12,13. (No solution)
- Iterative-Deepening Search D0: 1., D1: 1,3,4. D2: 1,3,9,10, 4,12,13. D3: 1,3,9, 27, 28, 10, 30, 31, 4, 12, 36.

3. Hill Climbing is a local search algorithm that has the advantage of using little memory and can sometimes find solutions in large or infinite state spaces when systematic search algorithms can not be used. But hill-climbing has its problems too, such as getting stuck at local maximums and local flat maximums. Using the diagram below, which depicts a search space where the algorithm has arrived at X, explain what this problem is. Describe an extension to Hill-Climbing that might improve such situations. **[2p]**



3. At X, B would be chosen with heuristic value 4. This already rules out reaching node I with a higher heuristic value. At B, a local minimum is reached where F and G have heuristic values less than B. So B= 4 is the final answer which is non-optimal. Any of the stochastic alternatives could help in this situation since, rather than always choosing B, other children, less optimal, could be chosen with a certain probability. A "bad" move could lead to that part of the search space with a more optimal choice or solution.

4. Consider the following sliding puzzle problem: | B | B | B | | W | W | W | The goal is to have all the white (W) tiles to the left of all the black (B) tiles. The puzzle has two legal moves with associated costs:
   (a) A tile can move into an adjacent empty location. This has cost 1.
   (b) A tile can hop over one or two other tiles into the empty position. This has a cost equal to the number of tiles jumped over.

   a. Provide an admissible heuristic for this problem and explain why it is an admissible heuristic. **[2p]**

One alternative:

$$H(n) = \frac{\text{\# of B's to the left of a W + \# of W's to the right of a B}}{2}$$

Why?

	Just numerator	*H(n)*
WW_BBWB	2 + 1	3/2
↓ Cost of move: 2		
WWWBB_B	0 + 0	
	Not admissible!	Admissible!

Additional detail:

.._BBW.. [2+1+z] —> ..WBB_..[0+0+z]
..BWW_..[1+2+z] —> .._WWB..[0+0+z]
.._WBW..[1+1+z] —> ..WWB..[0+0+z]
..BWB_..[1+1+z] —> .._WBB..[0+0+z]
etc.

Applying rule 2 with cost 2,
always decreases h(n) by max 3/2 or 2/2,
both values which are less than 2

.._BW.. [1+1+z] —> ..WB_..[0+0+z]
..BW_..[1+1+z] —> .._WB..[0+0+z]
etc.

Applying rule 1 with cost 1,
always decreases h(n) by max 2/2,
which is less than or equal to 1

Another alternative: $h(n) = 1/2 \sum_i w_i$

Take half the sum of the number of white tiles to the right of each black tile

It will take at least one move for a white tile to move past each black tile. Dividing by half ensures that we never overestimate since it's possible for a white tile to hop two black tiles in a single move.

TDDC17 Exam

- When: 23-10-27 14:00 -18:00
  - Using WISEflow, which means you do the exam on your own laptop.
  - You need to install the FLOWlock Browser
  - See link on course page for more information
- 2nd occasion: 23-01-04 14:00 -18:00

TDDC17 AI LE17 HT2023:
Course summary
Exam questions

www.ida.liu.se/~TDDC17

LINKÖPING UNIVERSITY