1 Exercise 1

1. a. A is possible to be the adjacency matrix of the graph. $D^{(2)}$ and $P^{(2)}$ is shown as follows.

$$D^{(2)} = \begin{pmatrix} 0 & \infty & \infty & \infty \\ 1 & 0 & \infty & 2 \\ \mathbf{3} & 2 & 0 & \mathbf{4} \\ -4 & \infty & \infty & 0 \end{pmatrix} P^{(2)} = \begin{pmatrix} Nil & Nil & Nil & Nil \\ 2 & Nil & Nil & 2 \\ \mathbf{2} & 3 & Nil & \mathbf{2} \\ 4 & Nil & Nil & Nil \end{pmatrix}$$

2

2.1 b. 20000/5=4000, 20000 disk pages and 5 main memory pages. this gives 20000/5=4000 runs.

2.2 b. We can afford up to 4-way as we have 5 main memory pages.

2.3 6. ceil($\log_{5-1}(4000)$) = 6 passes

2.4 28×10^4. 20000 disk pages and 5 main memory pages. this gives 20000/5=4000 runs. Second phase: $\lceil \log_4 4000 \rceil$=6 iterations. In total, (6+1)=7 times of read and write of the file. So $7 \times 2 \times 20000 = 28 \times 10^4$.

3

3.1 c. Maximization problem $\implies \dfrac{C^*}{C} \leq \rho \implies \dfrac{120}{C} \leq 1.5 \implies C \geq 80.$

3.2 d. Minimization problem $\implies \dfrac{C}{C^*} \leq \rho \implies \dfrac{C}{60} \leq 2.5 \implies C \leq 150.$

4

4.1 *PUSH*: 1, *POP*: 1, *COPY*: k.

4.2 Charge 2 for each *PUSH* and *POP* operation and 0 for each *COPY*. When we call *PUSH*, we use 1 to pay for the operation, and we store the other 1 on the item pushed. When we call *POP*, we again use 1 to pay for the operation, and we store the other 1 in the stack itself. Because the stack size never exceeds k, the actual cost of a *COPY* operation is at most k, which is paid by the k found in the items in the stack and the stack itself. Since there are k *PUSH* and *POP* operations between two consecutive *COPY* operations, there are k of credit stored, either on individual items (from *PUSH* operations) or in the stack itself (from *POP* operations) by the time a *COPY* occurs. Since the amortized cost of each operation is $\Theta(1)$ and the amount of credit never goes negative, the total cost of n operations is $\Theta(n)$.

Exercise 2, The S-Part, 50 points in total

1. Consider the set of clauses, $\Delta = \{\{S, \neg R\}, \{\neg S, Q, \neg P\}, \{\neg Q\}, \{S, Q, \neg P\}$.

1.1 (6 points) Run DPLL and give the trace until you get to the first conflict. In the splitting rule, always select the unassigned proposition that comes first in the alphabet, and try first truth value T then truth value F.

Rule applied UP Rule ($Q \mapsto F$)

Result $\{\{S, \neg R\}, \{\neg S, \neg P\}, \{S, \neg P\}$

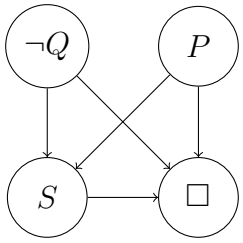
Rule applied Splitting Rule ($P \mapsto T$)

Result $\{\{S, \neg R\}, \{\neg S\}, \{S\}$

Rule applied UP Rule ($S \mapsto T$) ($S \mapsto F$ is also correct)

Result $\{\{\neg R\}, \{\square\}\}$

1.2 (6 points) Draw the implication graph for the conflict and all conflict graphs induced by the implication graph.



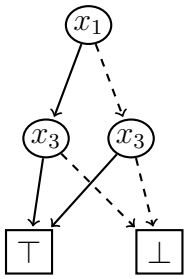
The conflict graph is equal to the implication graph.

Note: some students may also perform an extra UP step setting R to false. In that case, the implication graph has one more node $\neg R$, which would not be present in the conflict graph.

1.3 (2 points) Which is the clause that we can learn with the clause learning method presented in the lecture?

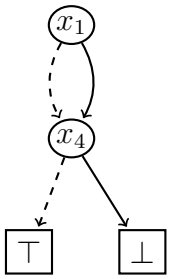
Learned Clause $\{\neg P\}$, as P is the only choice node in the conflict graph.

2.1 (12 points) Some of the following ROBDDs are not correct. For each of them, say if it is correct or not and justify your answer:



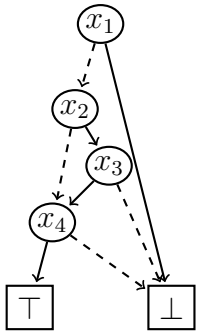
Answer

It is not correct because there are two equivalent nodes that have not been reduced (both nodes labelled with x_3).



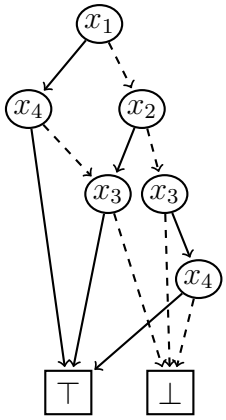
Answer

It is not correct because the node labelled with x_1 is redundant and should be ommited.



Answer

It is correct.



Answer

It is not correct because it does not respect the variable ordering, x_4 is before x_3 in one path and afterwards in another.

2.2 (3 points) Supose that we have three BDDs A, B, and C having 11 nodes each. If we use the apply operation twice in order to compute the BDD $(A \wedge B) \wedge C$, the resulting BDD **could** have (mark all correct options):

☒ 1 node ☒ 10 nodes ☒ 100 nodes ☒ 1000 nodes ☐ 10000 nodes

3 Planning

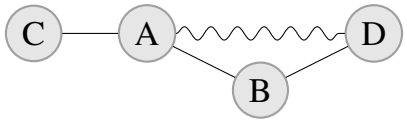
3.1 (6 points) Consider the following pairs of heuristics and tick the boxes where the dominance relation holds; where a heuristic h dominates h' if $h(s) \geq h'(s)$ for all states s in all planning tasks. Heuristics are incomparable if there exist states s and s' so that $h(s) > h'(s)$ and $h'(s') > h(s')$. Mark the correct option.

A)	☐ h^+ dominates h^*	☒ h^* dominates h^+	☐ h^* and h^+ are incomparable
B)	☐ h^{FF} dominates h^*	☐ h^* dominates h^{FF}	☒ h^* and h^{FF} are incomparable
C)	☒ h^{FF} dominates h^+	☐ h^+ dominates h^{FF}	☐ h^{FF} and h^+ are incomparable

3.2 (5 points) Suppose that you want to prove that the optimal solution for a planning task Π has 10 actions using Planning as SAT. Which formulas would you construct? For each formula specify (a) the number of steps, (b) whether sequential or $\forall$-step semantics are used, and (c) what is the expected result for the SAT solver (satisfiable or unsatisfiable), provided that the optimal solution has indeed 10 actions.
We need to construct two formulas, ϕ_9, and ϕ_{10}:

- ϕ_9 has 9 steps and sequential semantics. Therefore, it is satisfiable if and only if the planning task has a solution of cost 9 or less. We expect it to be unsatisfiable (i.e. showing that $h^*(I) \geq 10$).
- ϕ_{10} has 10 steps and sequential semantics. Therefore, it is satisfiable if and only if the planning task has a solution of cost 10 or less. We expect it to be satisfiable (i.e. showing that $h^*(I) \leq 10$).

4 (10 points) Consider the following map with four different locations: A, B, C, D. There are two different types of connections: footways (denoted by straight lines), and cycleways (denoted by curved lines). It is not allowed to use the cycleways without wearing a bicycle helmet. Alice wants to get from location A to location D, but she is not wearing the helmet.



Using the following predicates, fill the PDDL snippets that specify the ride-bike action (there are more actions, but we ignore them here), the objects, initial state and goal:

```
  (:predicates
       (at ?l) % Describes the position of Alice
       (wearing-helmet)
       (footway ?l1 ?l2)
       (cycleway ?l1 ?l2)
  )

(:action ride-bike
  :parameters (?l1 l2)
  :precondition (and (wearing-helmet)
                     (at ?l1)
                     (cycleway (?l1 ?l2)))
  :effect (and (at ?l2) (not (at ?l1)))
)

(:objects
 A B C D
)

(:init
  (at A)
  (footway A C) (footway C A)
  (footway A B) (footway B A)
  (footway D B) (footway B D)
  (cycleway D A) (cycleway A D)

)

(:goal
 (at D)
)
```