Preparing for the Final
Dec 16, 10:30-12:30, HG 1800

Kalev Kask

ICS 271

Fall 2014

Basics

- 2 hours
- closed-book
- 1 (one) sheet of A4 size paper of notes

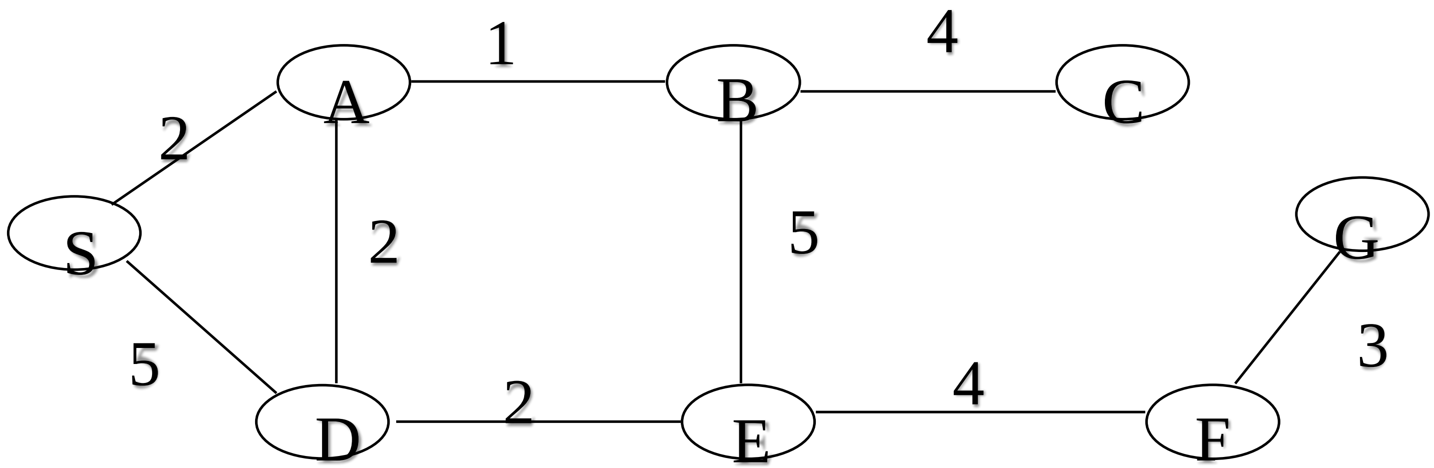
Material Covered

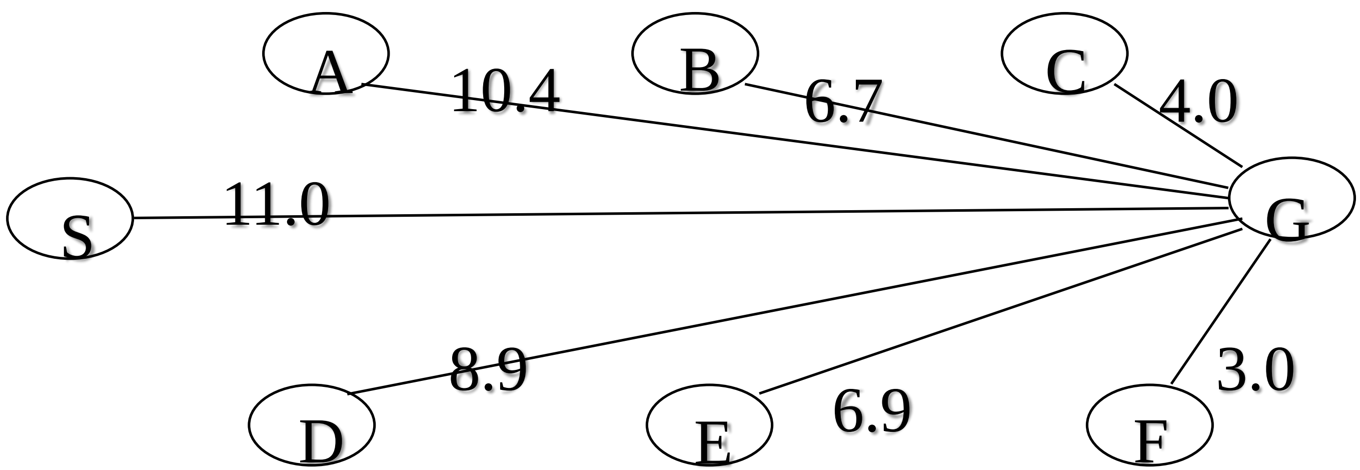
- Chapters 3-10
  - Search
  - Games
  - Constraint Satisfaction
  - Propositional Logic
  - First Order Logic
  - Classical Planning

Chapters 3,4 (Search) Concepts

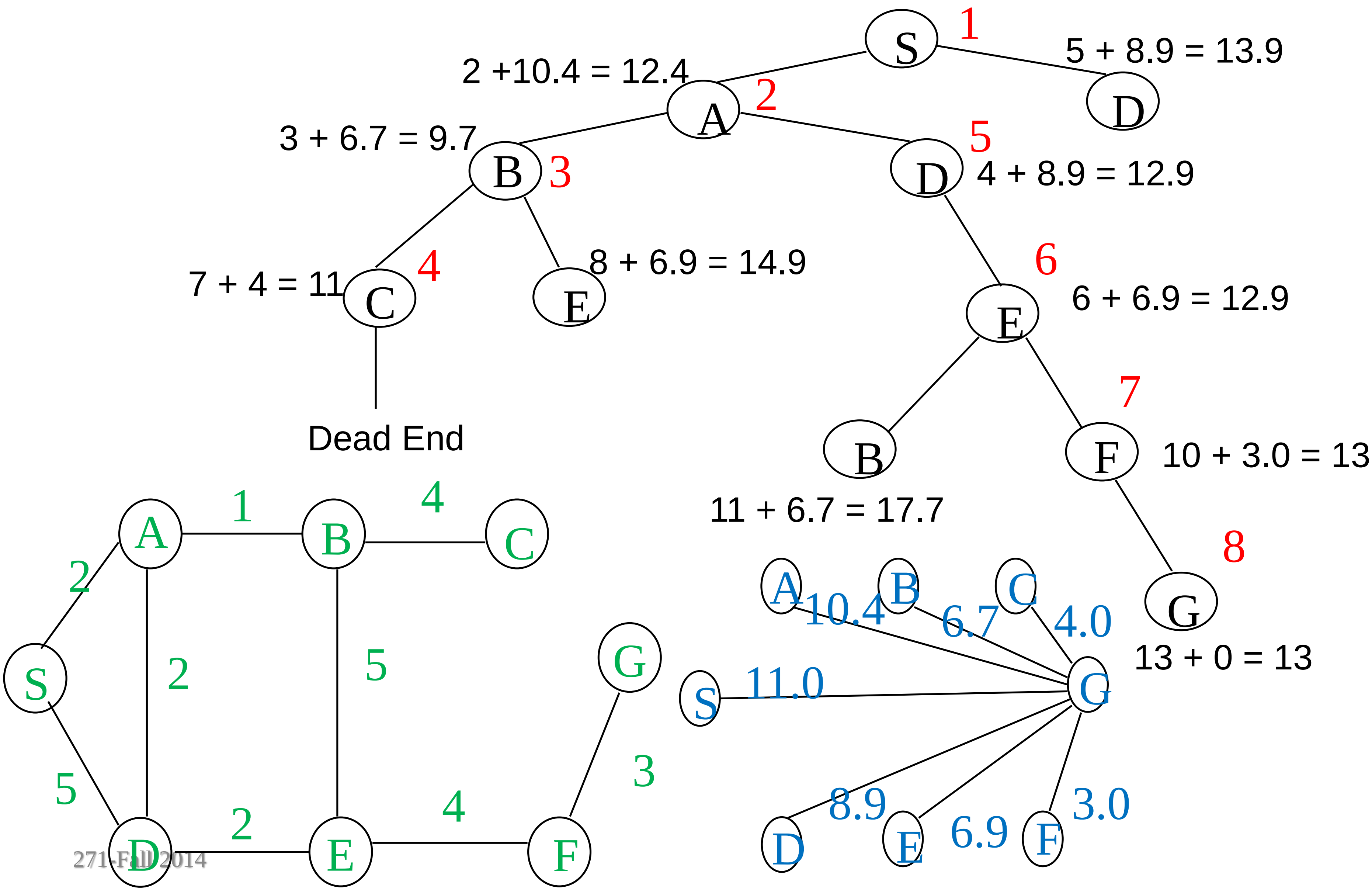
- Search space : states (initial, goal), actions
- Search tree/graph
- Breadth-first, depth-first, uniform-cost search
  - Expanding a node, open (frontier), closed (explored) lists
  - Optimality, complexity
  - Depth limited search, iterative deepening search
- Heuristic search
  - Heuristic fn, admissibility, consistency
  - f, h, g, h*, g*
  - Heuristic dominance
- Greedy search
- A*, IDA*
- Branch-and-Bound DFS
- Generating heuristics from relaxed problems, pattern databases
- Hill-climbing search, SLS, local vs. global maxima

Search Problem

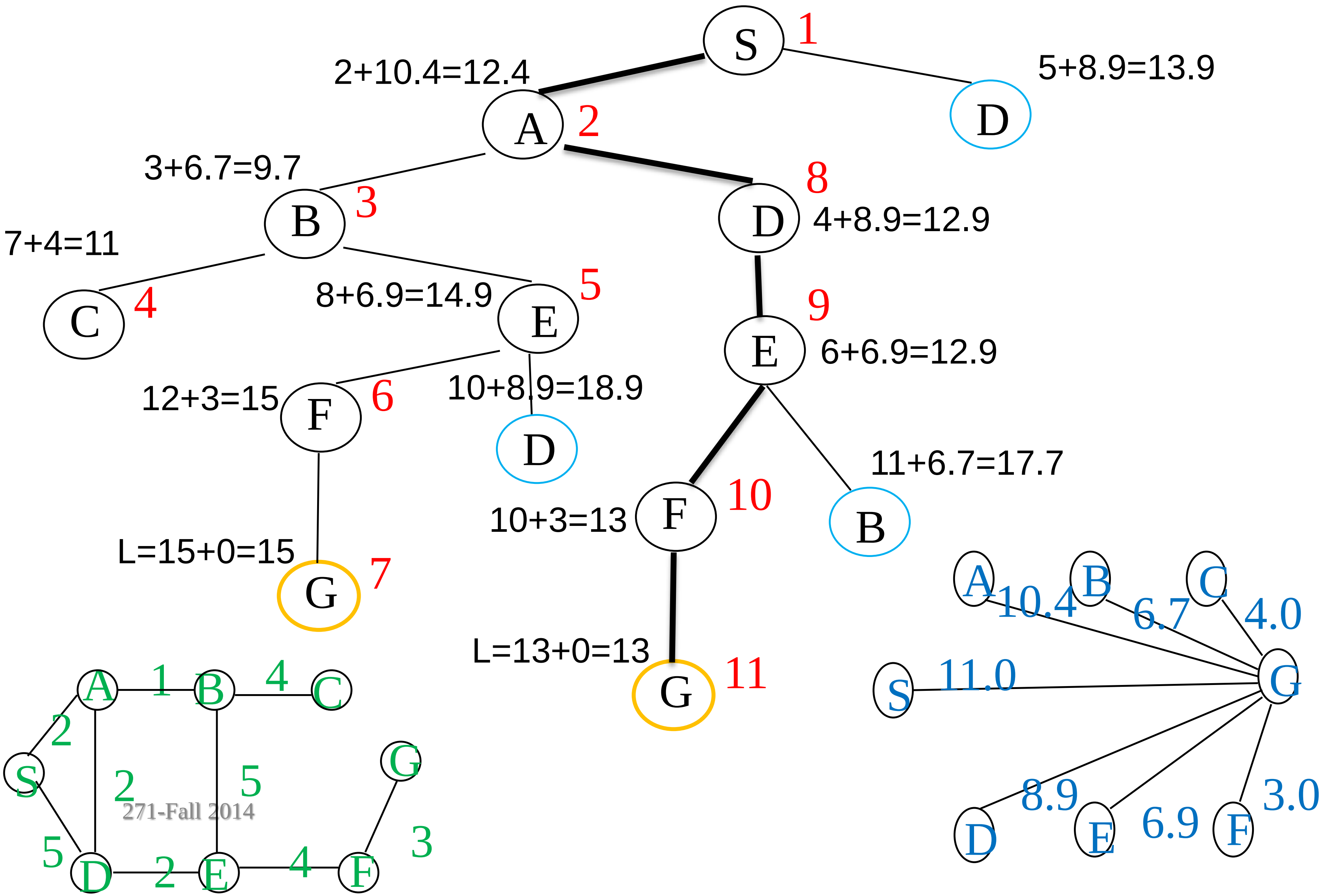




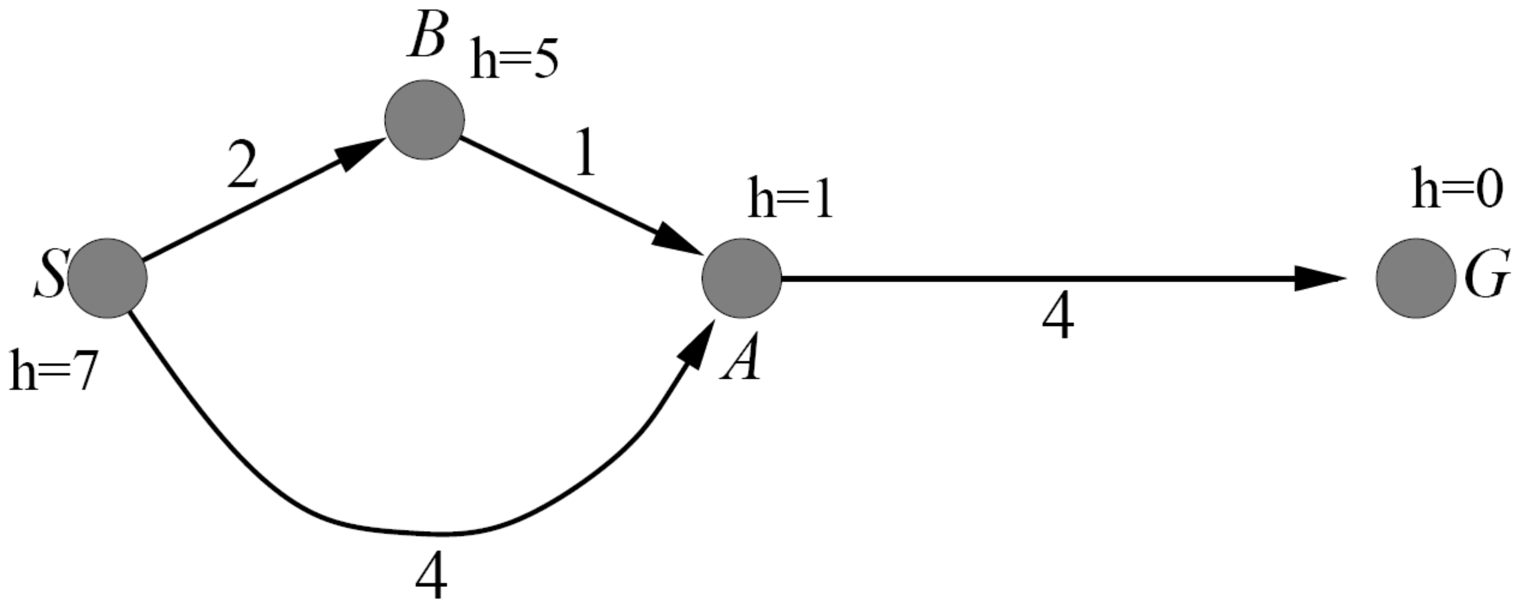
Example of A* Algorithm in Action



Example of Branch and Bound in action



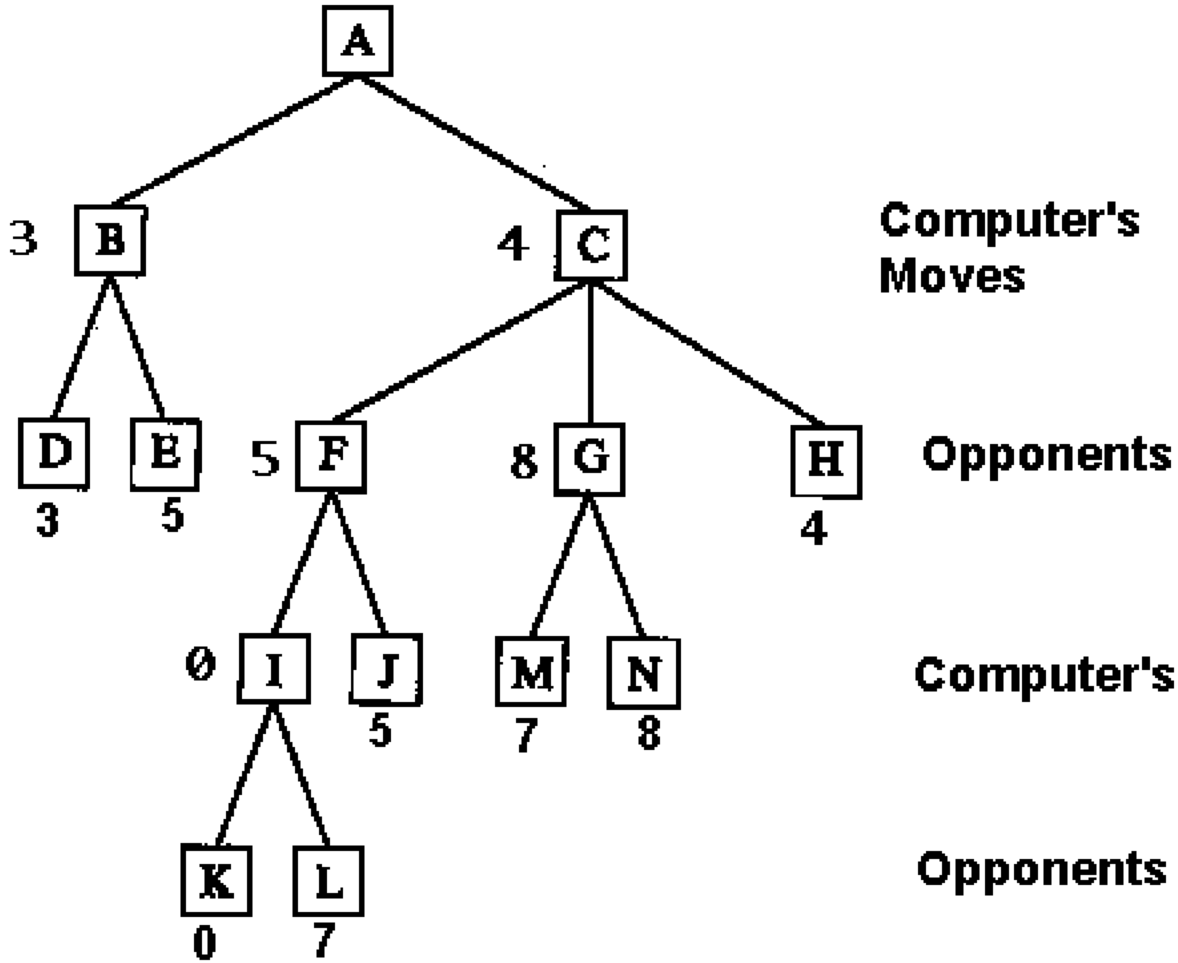
Admissible but not consistent



Chapter 5 (Games) Concepts

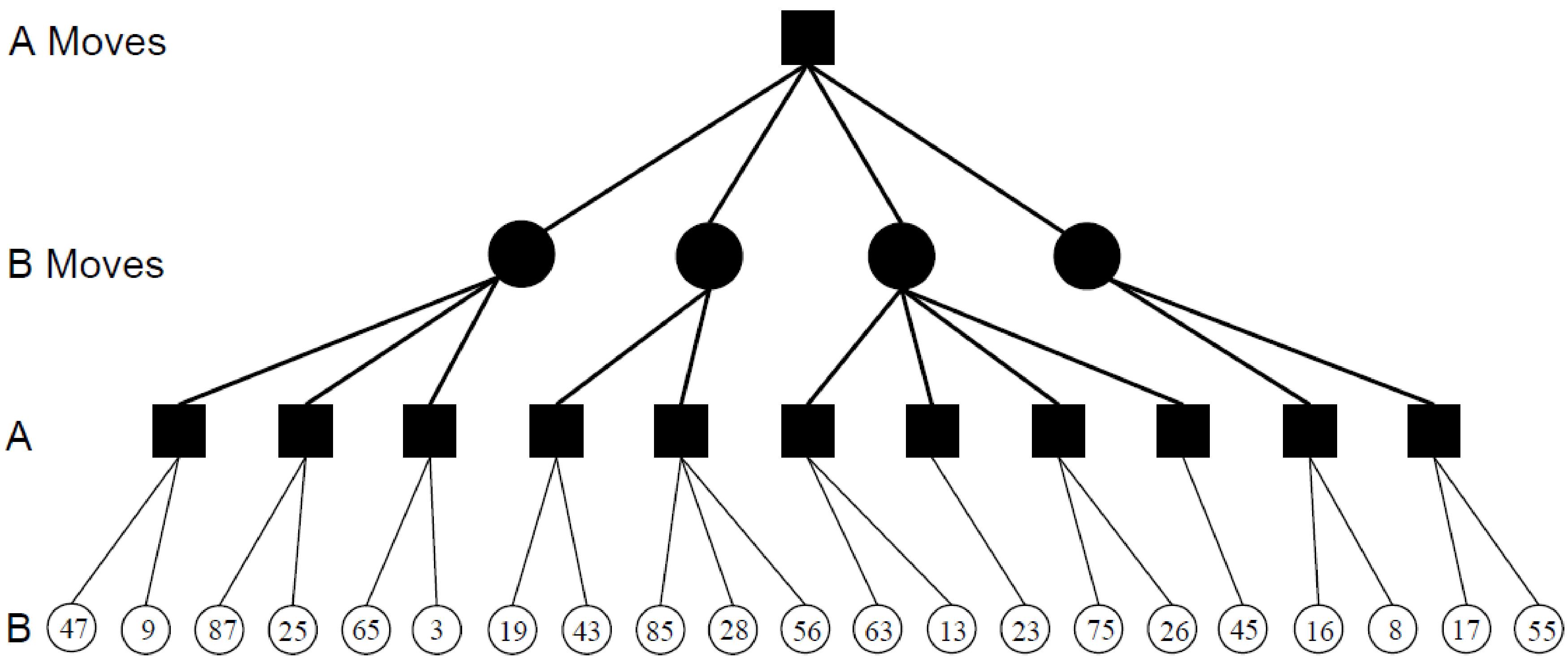
- Game tree
  - Players
  - Actions/moves
  - Terminal utility
  - MIN/MAX nodes
- MINIMAX algorithm
- Alpha/Beta pruning
  - Effect of node/move ordering on pruning
- Evaluation functions
  - Why do we need them?
- Stochastic games

A Game tree

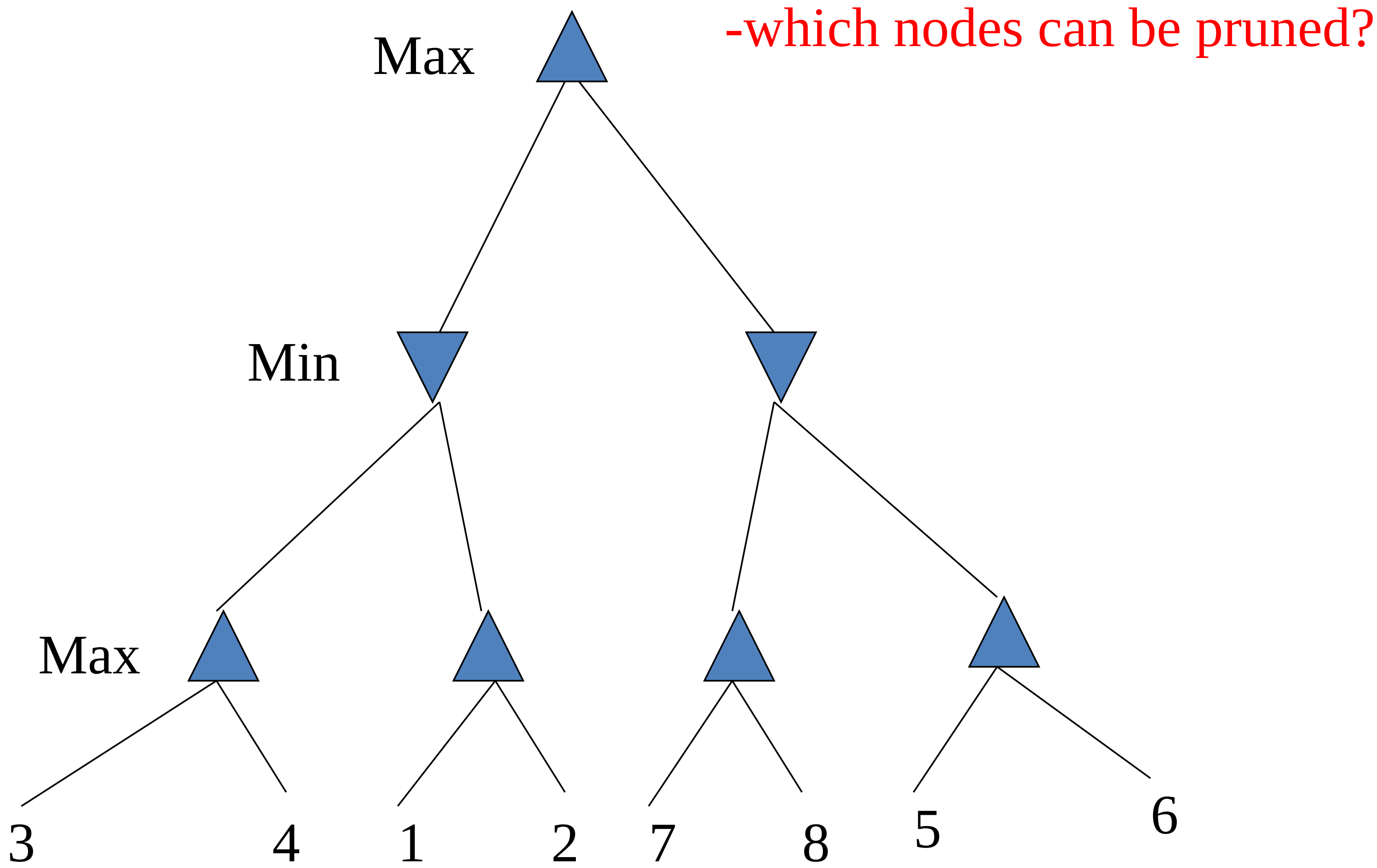


A
3
B
4
C
Computer's
Moves
D
E
5
F
8
G
H
Opponents
3
5
4
0
I
J
M
N
Computer's
5
7
8
K
L
Opponents
0
7

Another game tree

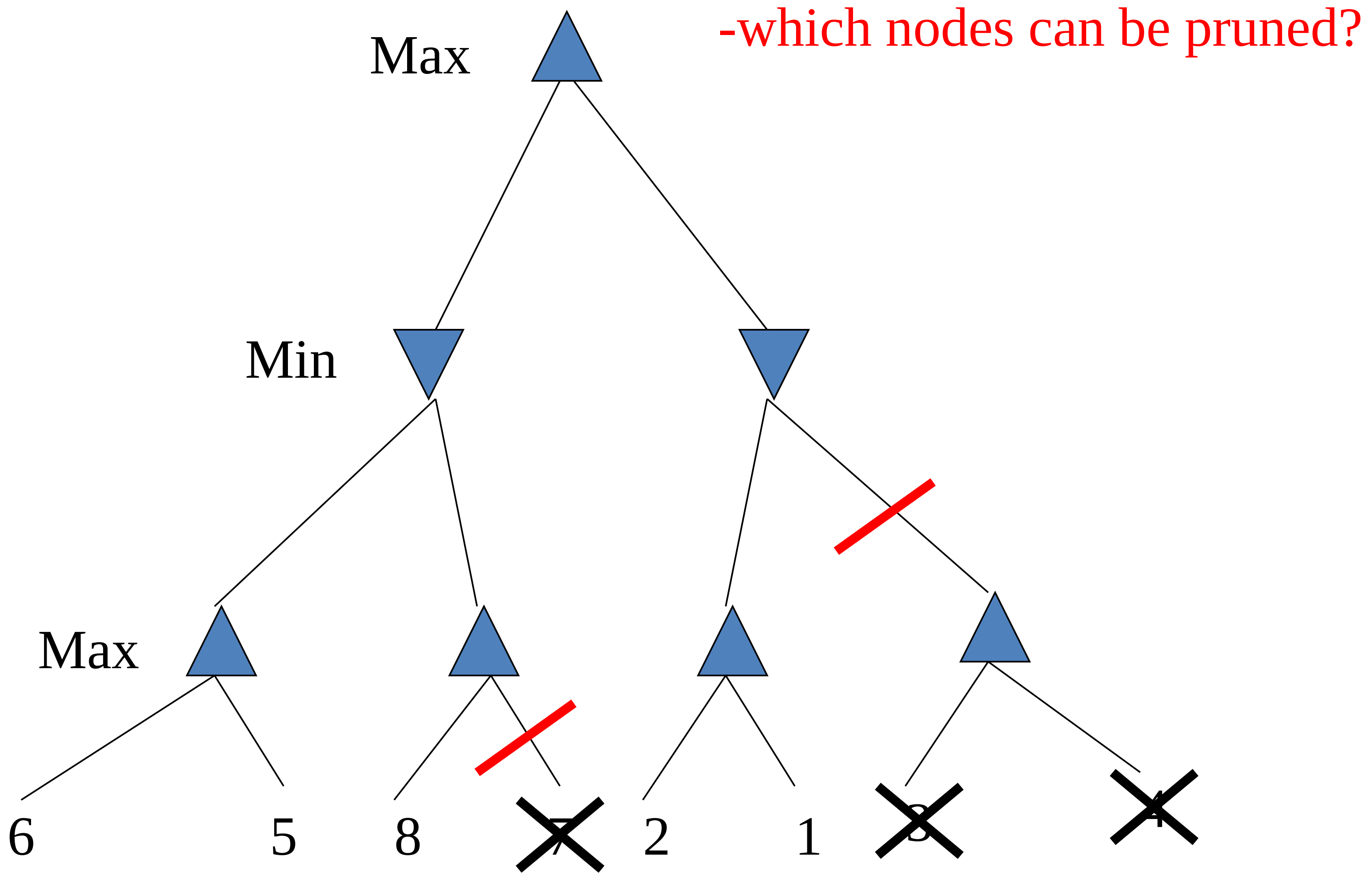


Answer to Example



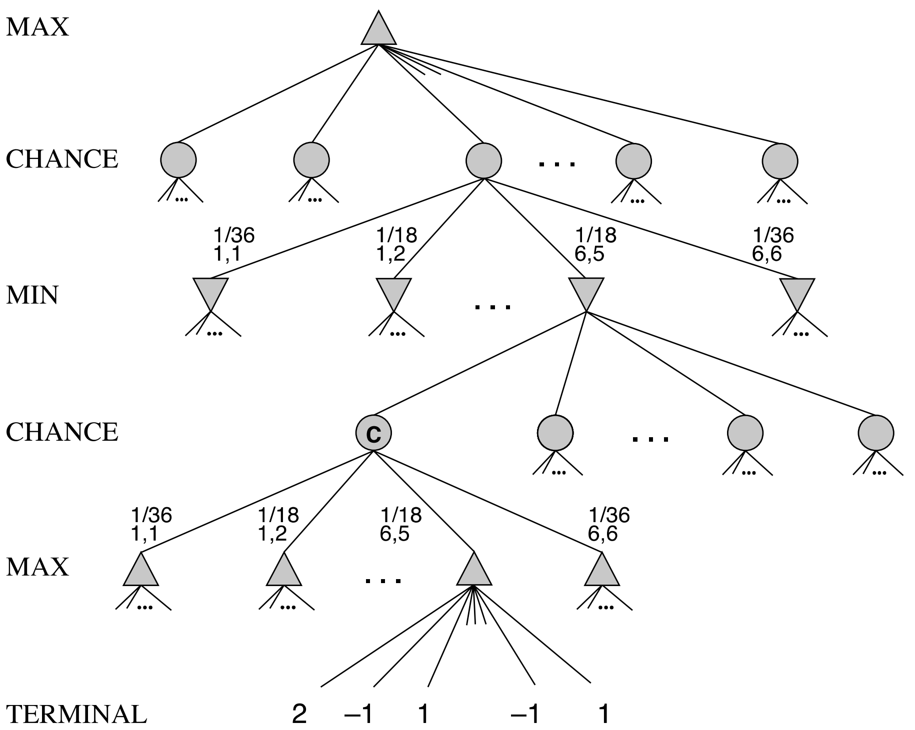
Answer: NONE! Because the most favorable nodes for both are explored last (i.e., in the diagram, are on the right-hand side).

Answer to Second Example
(the exact mirror image of the first example)



Answer: LOTS! Because the most favorable nodes for both are explored first (i.e., in the diagram, are on the left-hand side).

Schematic Game Tree for Backgammon Position



- How do we evaluate good move?
- By expected utility leading to expected minimax
- Utility for max is highest expected value of child nodes
- Utility of min-nodes is the lowest expected value of child nodes
- Chance node take the expected value of their child nodes.

- Try Monte-Carlo here!!!

Chapter 6 (CSP) Concepts

- Variables, domains, constraints
- A solution : assignment of values to variables so that all constraints are satisfied
- Constraint graph
- Local consistency
  - Arc-consistency, path-consistency, k-consistency
- Backtracking search (Q : how is BT search different from DFS?)
  - Variable, value ordering heuristics
- Interleaving search and inference
  - E.g. BT with arc-consistency
- Back-jumping, no-good learning
- Greedy local search
  - Min-conflicts
- Tree-structured CSPs
- Cut-set conditioning, tree-decomposition

Arc-consistency

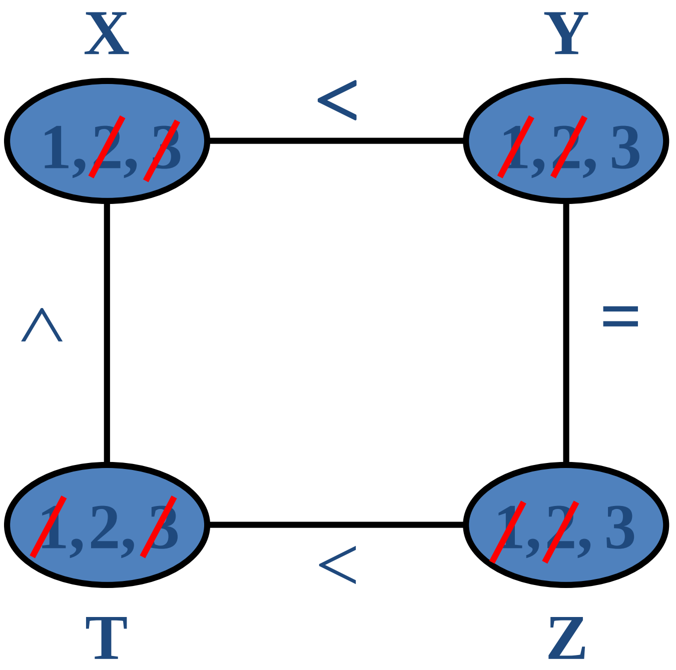
$1 \leq X, Y, Z, T \leq 3$

$X < Y$

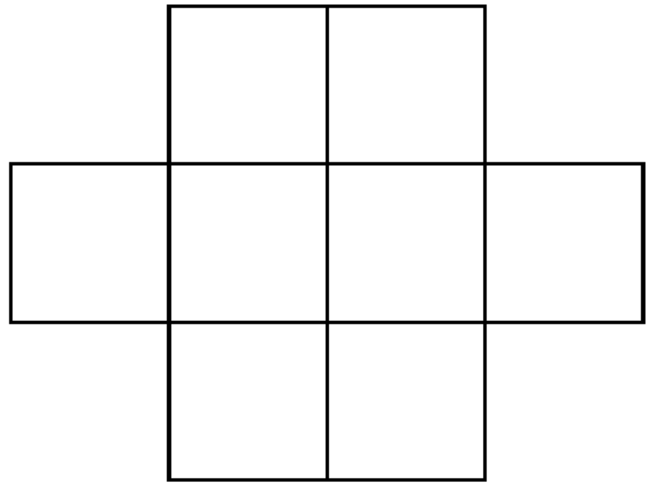
$Y = Z$

$T < Z$

$X < T$



A Constraint problem



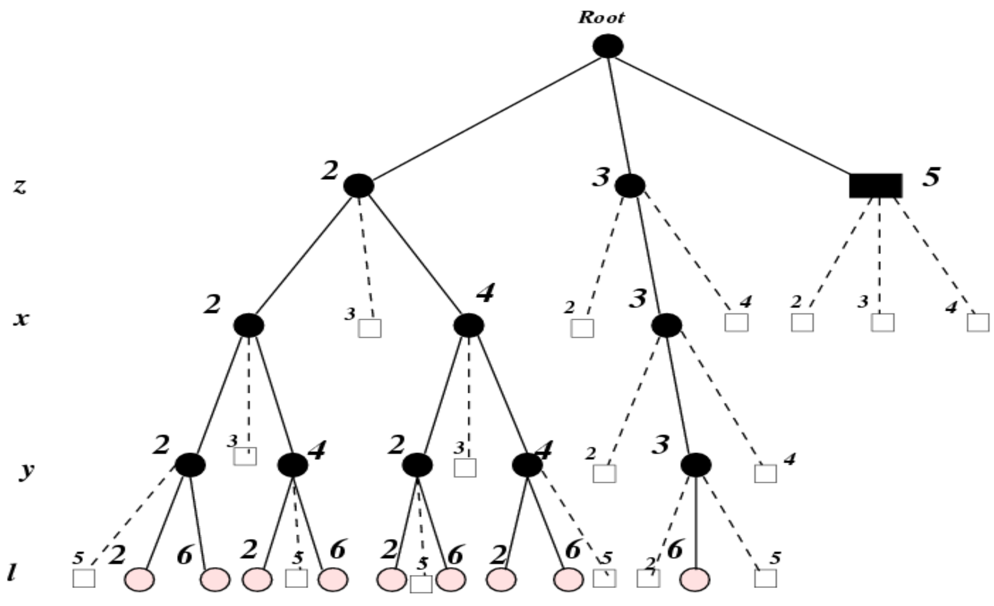
The task is to label the boxes above with the numbers 1-8 such that the labels of any pair of adjacent squares (i.e. horizontal vertical or diagonal) differ by at least 2 (i.e. 2 or more).

(a) Write the constraints in a relational form and draw the constraint graph.

(b) Is the network arc-consistent ? if not, compute the arc-consistent network.

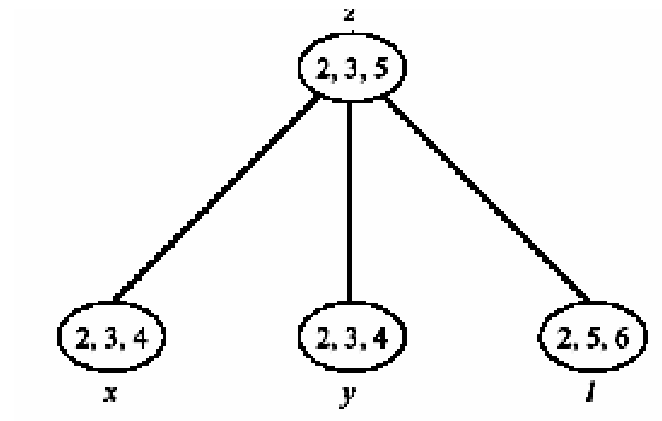
(c) Is the network consistent ? If yes, give a solution.

The effect of variable ordering

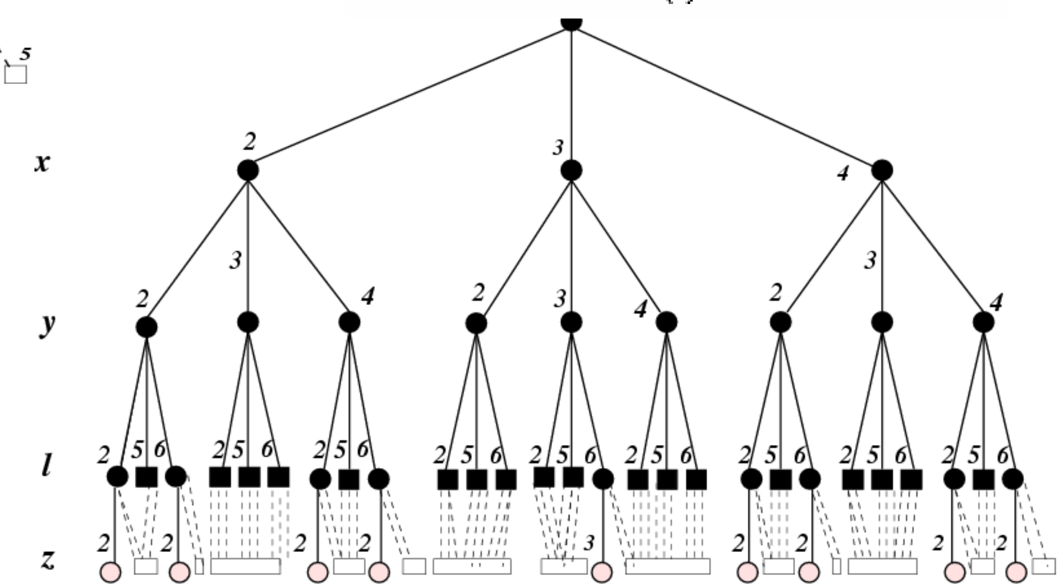


(b)

z divides x, y and t

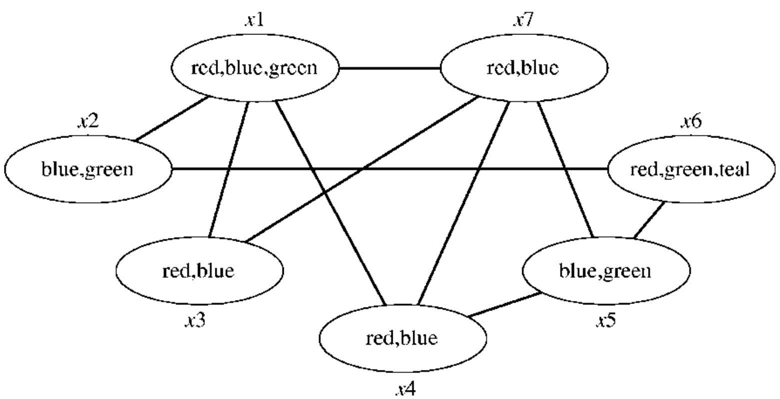


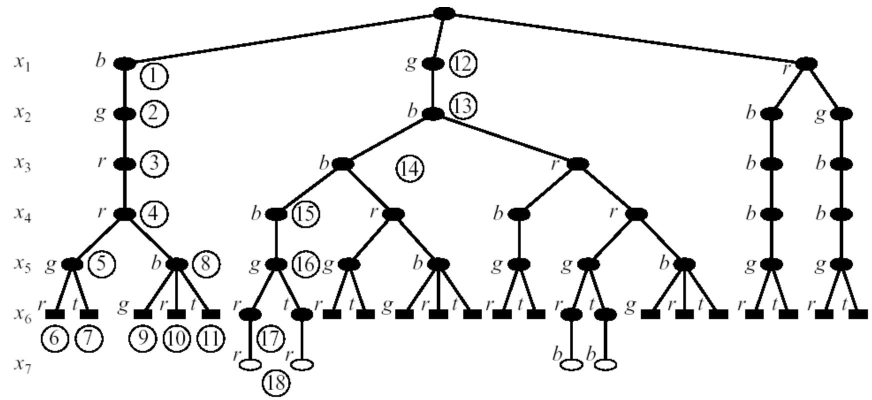
(a)



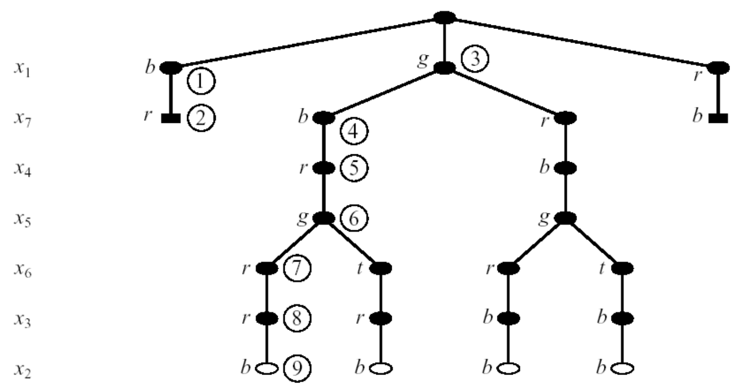
(c)

Backtracking Search for a Solution



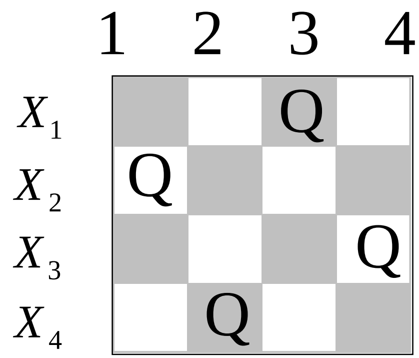


(a)



(b)

Min-Conflicts



At each step, find globally minimizing move!

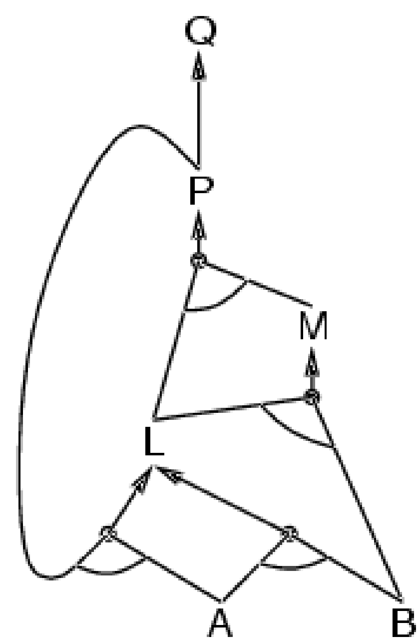
Chapter 7 (Prop Logic) Concepts

- Syntax
  - Propositional symbols
  - Logical connectives
- Semantics
  - Worlds, models
  - Entailment
  - Inference
- Model checking
- Modus Ponens
- CNF
- Horn clauses, Forward/Backward chaining
- Resolution
- DPLL backtracking search

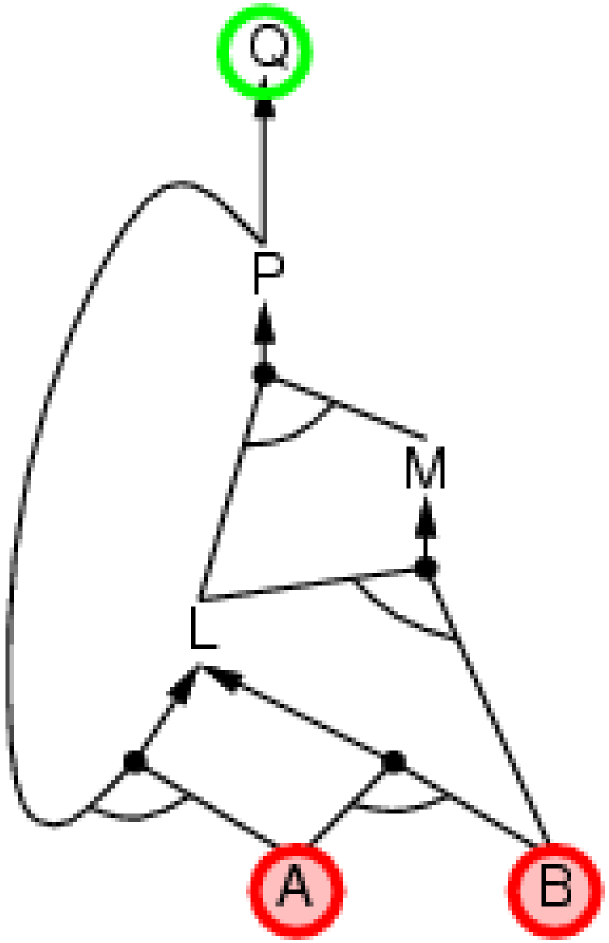
Forward chaining

- Idea: fire any rule whose premises are satisfied in the *KB*,
  - add its conclusion to the *KB*, until query is found

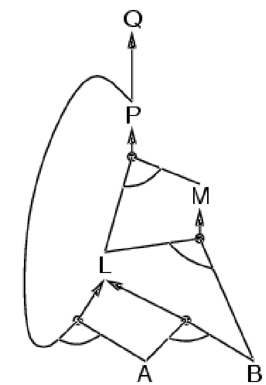
$$P \Rightarrow Q$$
$$L \wedge M \Rightarrow P$$
$$B \wedge L \Rightarrow M$$
$$A \wedge P \Rightarrow L$$
$$A \wedge B \Rightarrow L$$
$$A$$
$$B$$



Backward chaining example



$P \Rightarrow Q$
$L \wedge M \Rightarrow P$
$B \wedge L \Rightarrow M$
$A \wedge P \Rightarrow L$
$A \wedge B \Rightarrow L$
A
B



Chapters 8,9 (FOL) Concepts

- Syntax
  - Variables, const symbols, fn symbols, predicate symbols
  - Terms, atomic sentences
  - Quantifiers
- Semantics
  - Model, interpretation
  - Entailment
  - Inference

Chapters 8,9 (FOL) Concepts cont.

- Universal, existential instantiation
- Unification
- Generalized Modus Ponens
- Definite clauses, Forward/Backward chaining
- Converting a FOL sentence to CNF
- Resolution
  - Answer extraction

FOL Resolution Problem

(Problem 16.10 from Nillson) Use resolution refutation on a set of clauses to prove that there is a green object if we are given:

- If pushable objects are blue, then nonpushable ones are green.
- All objects are either blue or green but not both.
- If there is a nonpushable object, then all pushable ones are blue.
- Object 01 is pushable.
- Object 02 is not pushable.

(a) Convert these statements to expressions in first-order predicate calculus.

(b) Convert the preceding predicate-calculus expressions to clause form.

(c) Combine the preceding clause form expressions with the clause form of the negation of the statement to be proved, and then show the steps used in obtaining a resolution refutation

(d) Use resolution-answer-extraction to find a particular object that is green

Chapter 10 (Planning) Concepts

- Planning as inference, situation calculus
  - States, actions, frame axioms
- STRIPS (PDDL) language
  - Factored representation of states
  - Actions (schema) : PC, AL/DL (EL)
- Planning as search
  - Recursive STRIPS
  - Forward/Backward
- Heuristics for planning, relaxed problem idea
  - Ignore PC, DL
  - Abstraction
- Planning graphs : construction, properties, GraphPlan
- Planning as satisfiability

STRIPS/PDDL

$Init(On(A, Table) \wedge On(B, Table) \wedge On(C, Table)$
$\wedge\ Block(A) \wedge Block(B) \wedge Block(C)$
$\wedge\ Clear(A) \wedge Clear(B) \wedge Clear(C))$
$Goal(On(A, B) \wedge On(B, C))$
$Action(Move(b, x, y),$
PRECOND: $On(b, x) \wedge Clear(b) \wedge Clear(y) \wedge Block(b) \wedge$
$(b \neq x) \wedge (b \neq y) \wedge (x \neq y),$
EFFECT: $On(b, y) \wedge Clear(x) \wedge \neg On(b, x) \wedge \neg Clear(y))$
$Action(MoveToTable(b, x),$
PRECOND: $On(b, x) \wedge Clear(b) \wedge Block(b) \wedge (b \neq x),$
EFFECT: $On(b, Table) \wedge Clear(x) \wedge \neg On(b, x))$

Figure 11.4 A planning problem in the blocks world: building a three-block tower. One solution is the sequence [$Move(B, Table, C)$, $Move(A, Table, B)$].