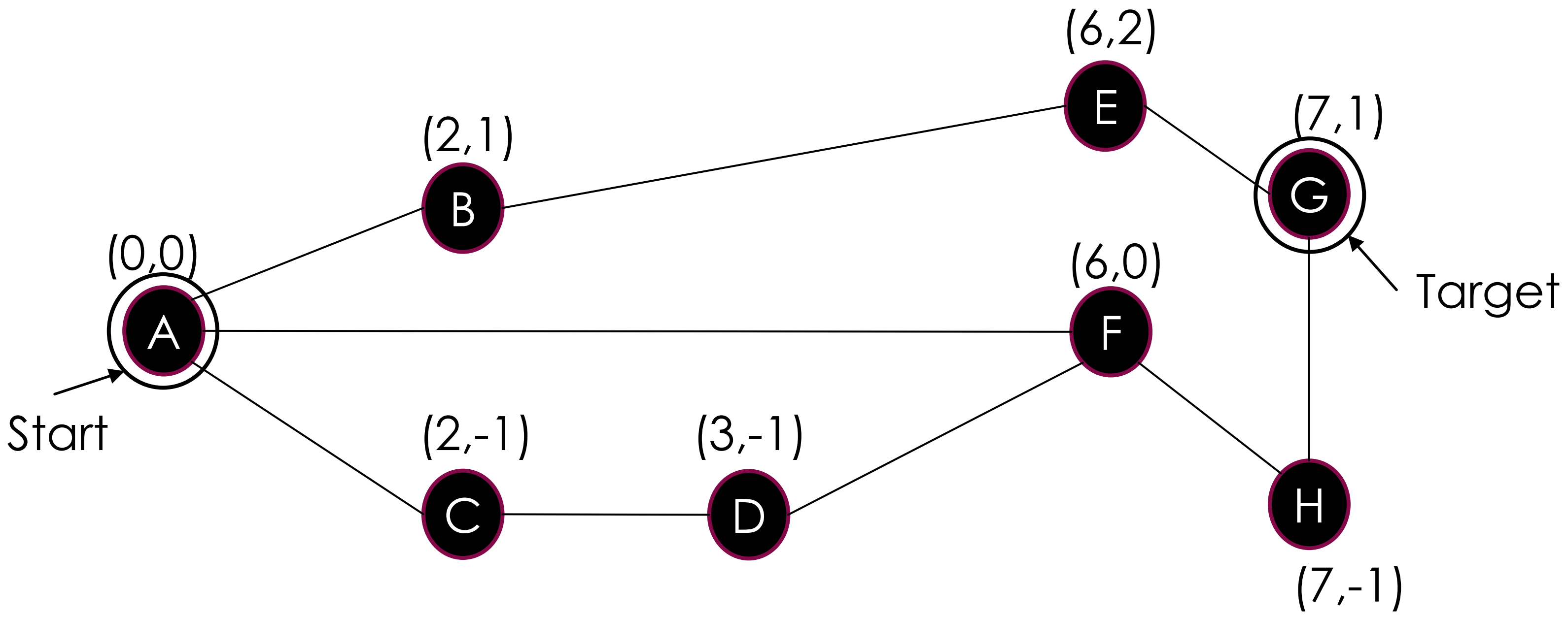
2020 AI Review

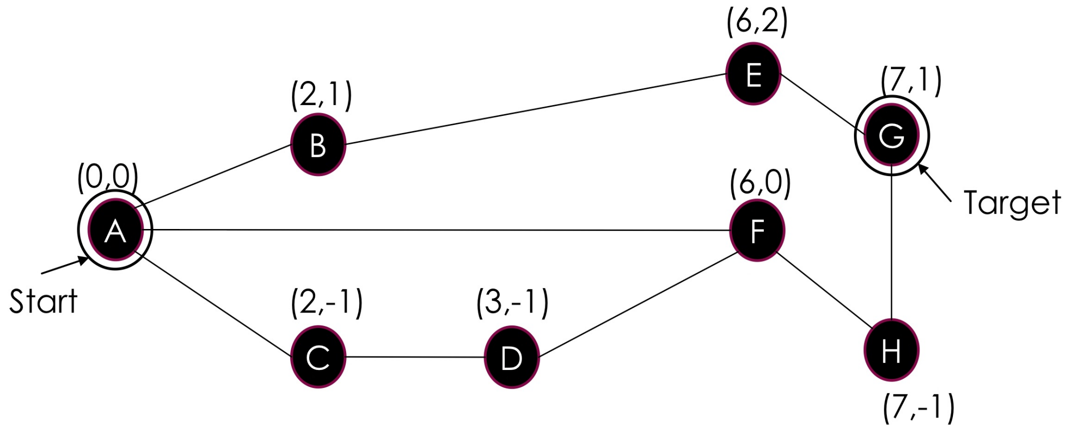
Question 1

Use A* algorithm and design a reasonable heuristic function ***h(x)*** to find a path from the Start to the Target in the following graph.

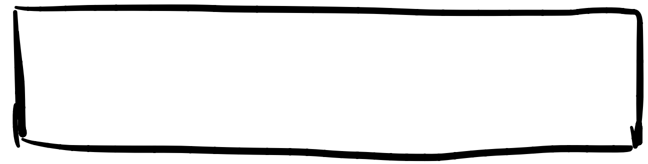
1) Assume your A* cost function is ***f(x) = g(x) + h(x)***. Draw the search tree and give the value of ***f(x)*** for each node.

2) List the Priority Queue and Explored Set for each step.





quene

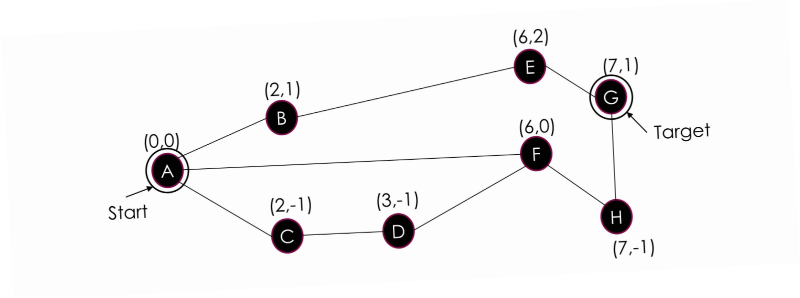


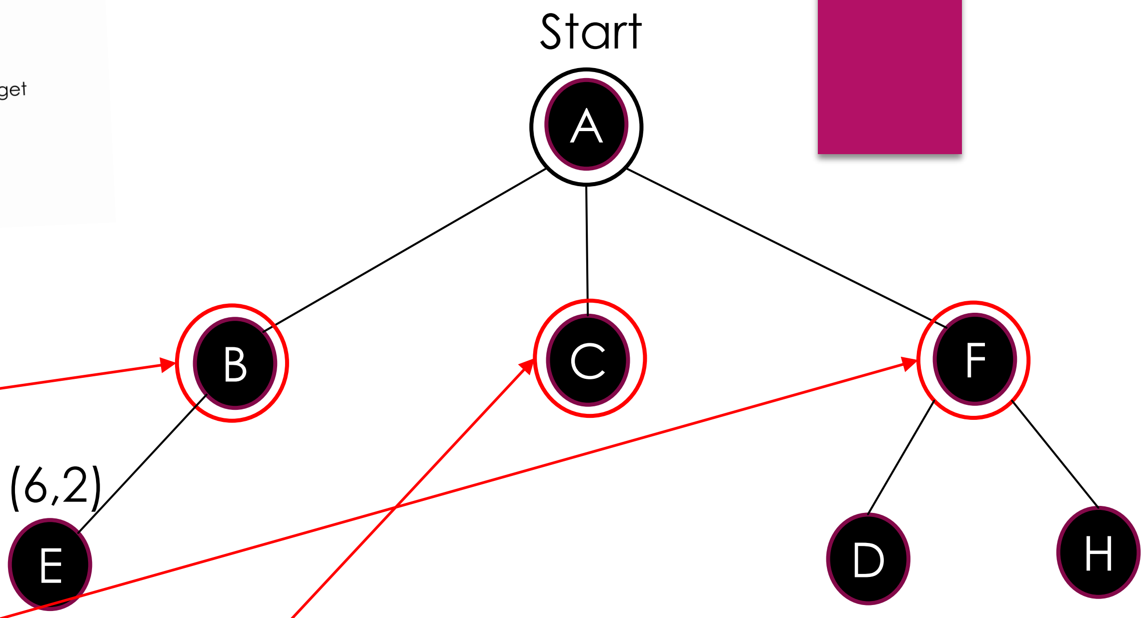
(0, 0)

A* Algorithm

```
function A-STAR-SEARCH(initialState, goalTest)
    returns SUCCESS or FAILURE :  /* Cost f(n) = g(n

    frontier = Heap.new(initialState)
    explored = Set.new()

    while not frontier.isEmpty():
        state = frontier.deleteMin()
        explored.add(state)

        if goalTest(state):
            return SUCCESS(state)

        for neighbor in state.neighbors():
            if neighbor not in frontier ∪ explored:
                frontier.insert(neighbor)
            else if neighbor in frontier:
                frontier.decreaseKey(neighbor)

    return FAILURE
```

Question 1 Analysis





Initial
Priority Queue: A(7.071)
Explored Set: {}

Step1
Priority Queue: B(2.236+5),C(2.236+5.383),F(6+1.414)
Explored Set: {A(7.071)}

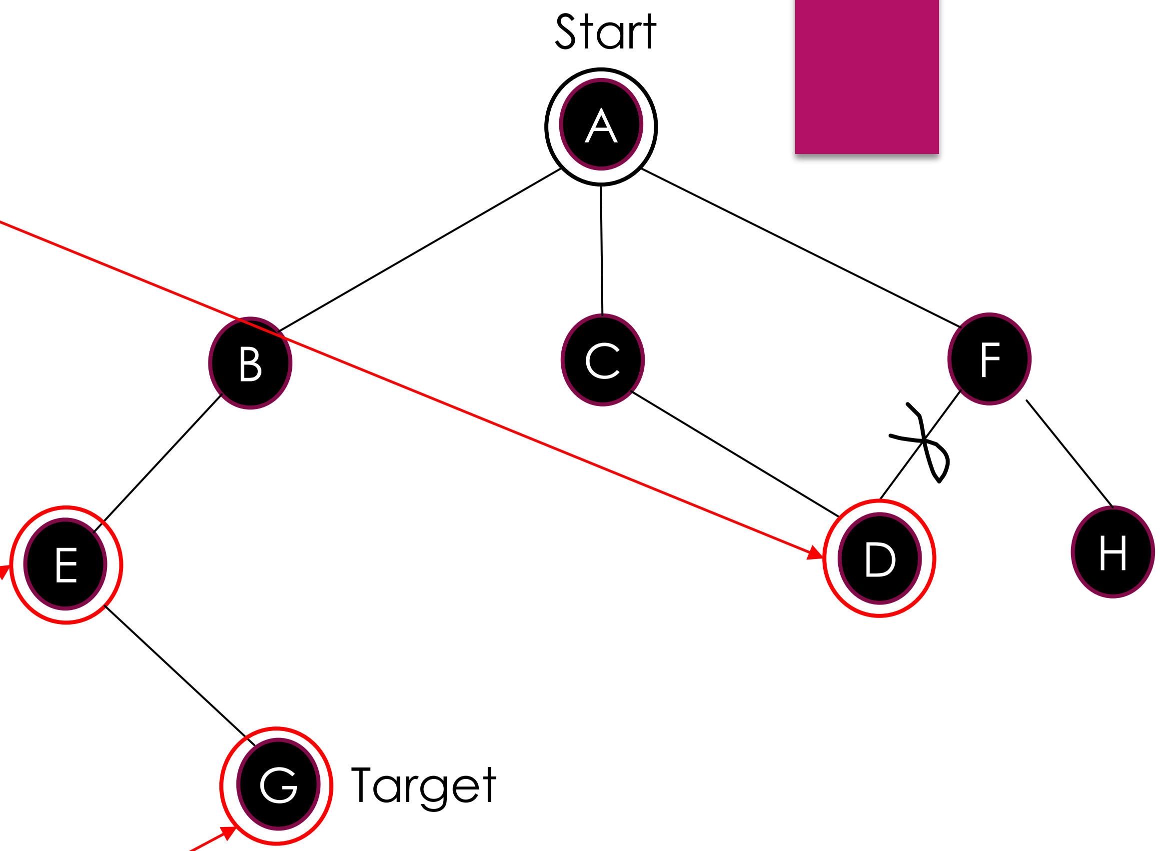
Step2
Priority Queue: F(6+1.414), C(2.236+5.383), E(6.359+1.414)
Explored Set: {A(7.071), B(2.236+5)}

Step3
Priority Queue: C(2.236+5.383),
E(6.359+1.414),D(9.162+4.472),H(7.414+2)
Explored Set: {A(7.071), B(2.236+5), F(6+1.414)}

Step4
Priority Queue:
D(3.236+4.472),E(6.359+1.414),~~D(9.162+4.472)~~,H(7.414+2)
Explored Set: {A(7.071), B(2.236+5), F(6+1.414)
C(2.236+5.383)}

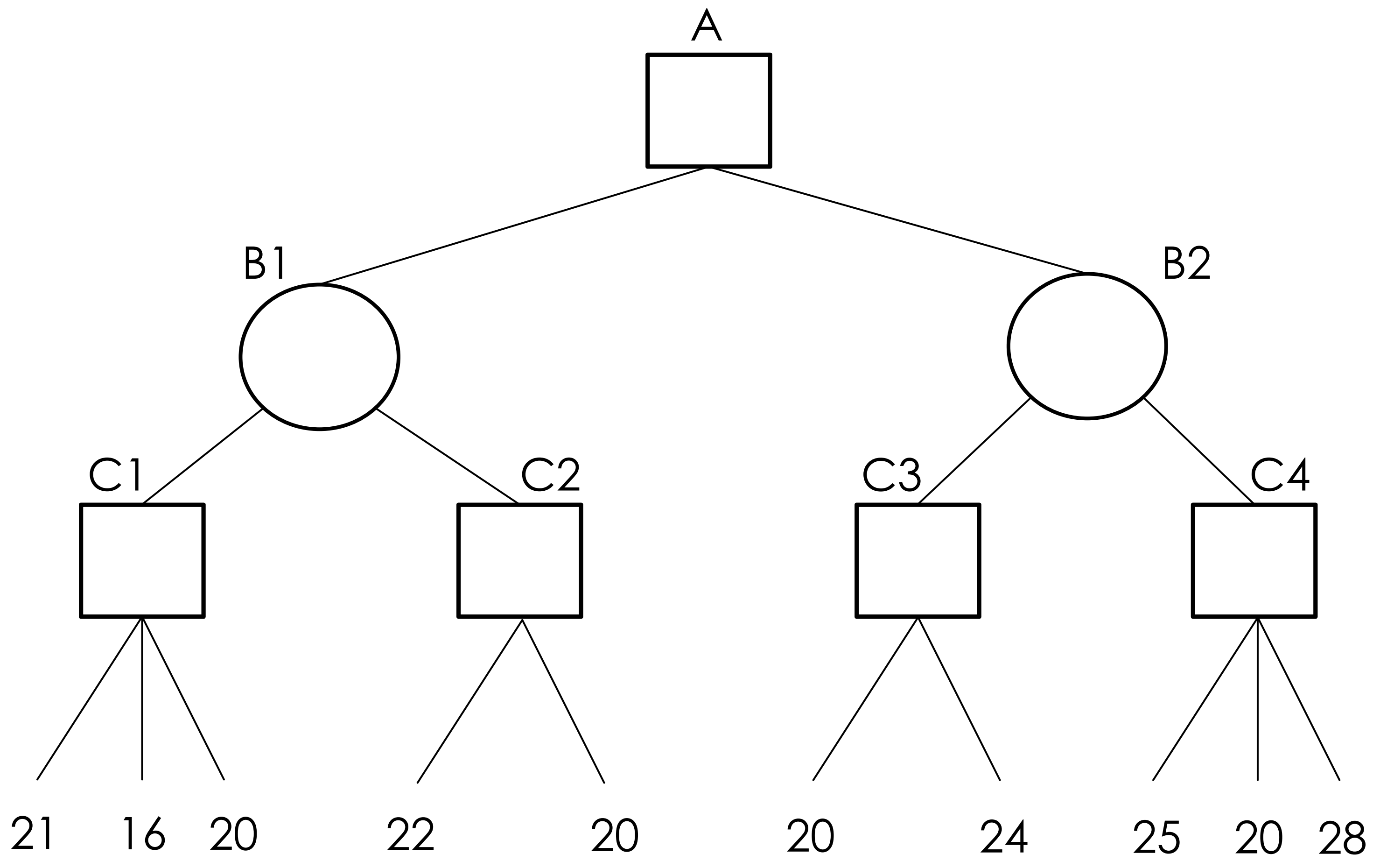
Step5
Priority Queue: E(6.359+1.414),H(7.414+2)
Explored Set: {A(7.071), B(2.236+5), F(6+1.414)
C(2.236+5.383), D(3.236+4.472)}

Step6
Priority Queue: G(7.773),H(7.414+2)
Explored Set: {A(7.071), B(2.236+5), F(6+1.414)
C(2.236+5.383), D(3.236+4.472), E(6.359+1.414)}



Question 2

For the following game tree, in which the numbers at the leaf nodes indicate their utility values, apply Alpha-Beta pruning to prune unnecessary branches. Please directly label the nodes with (Alpha, Beta) values, and put a "X" on branches that should be pruned.



Alpha-Beta Pruning

function ALPHA-BETA-SEARCH(*state*) **returns** an action
  $v \leftarrow$ MAX-VALUE(*state*, $-\infty, +\infty$)
  return the *action* in ACTIONS(*state*) with value v

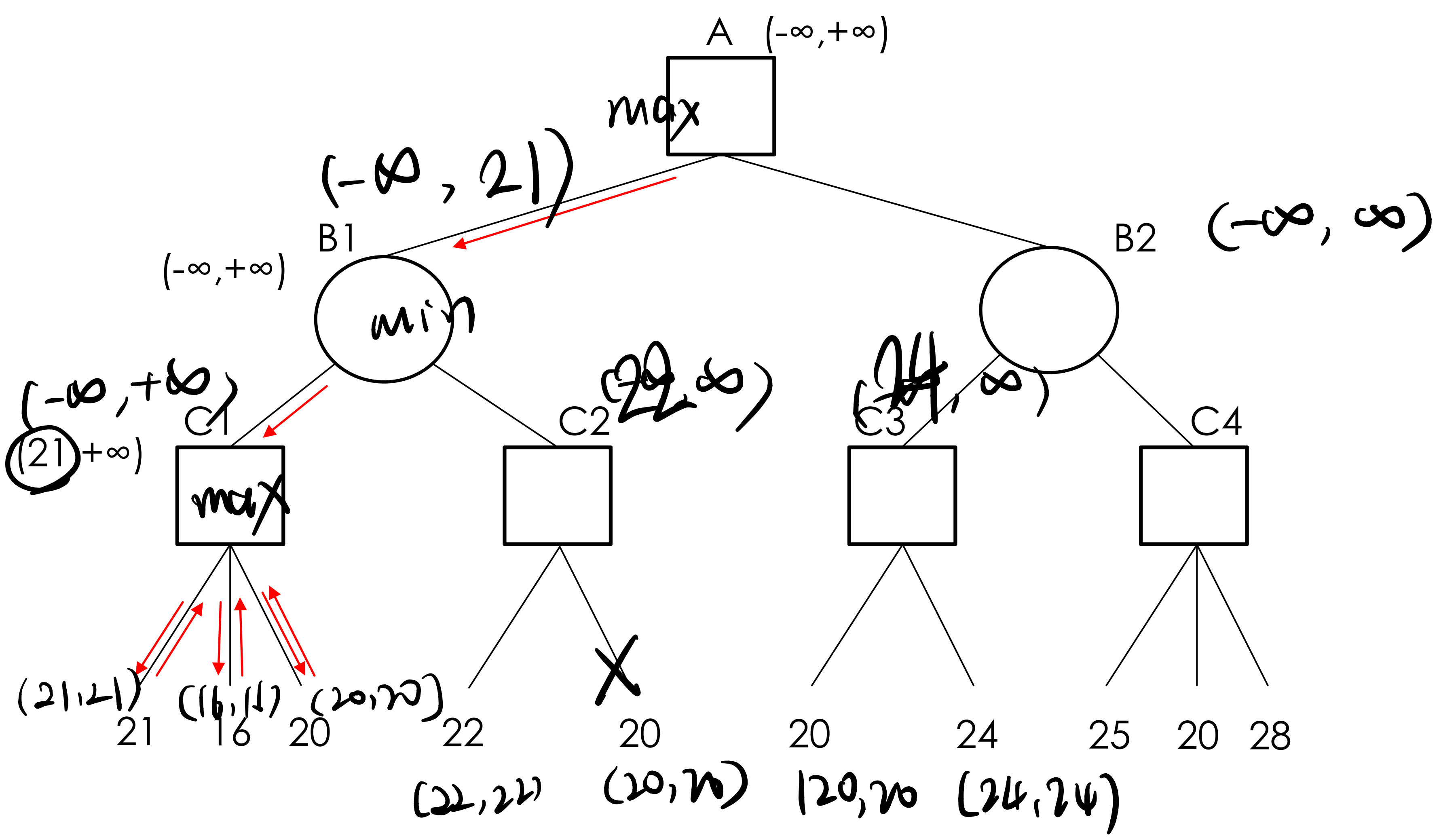
---

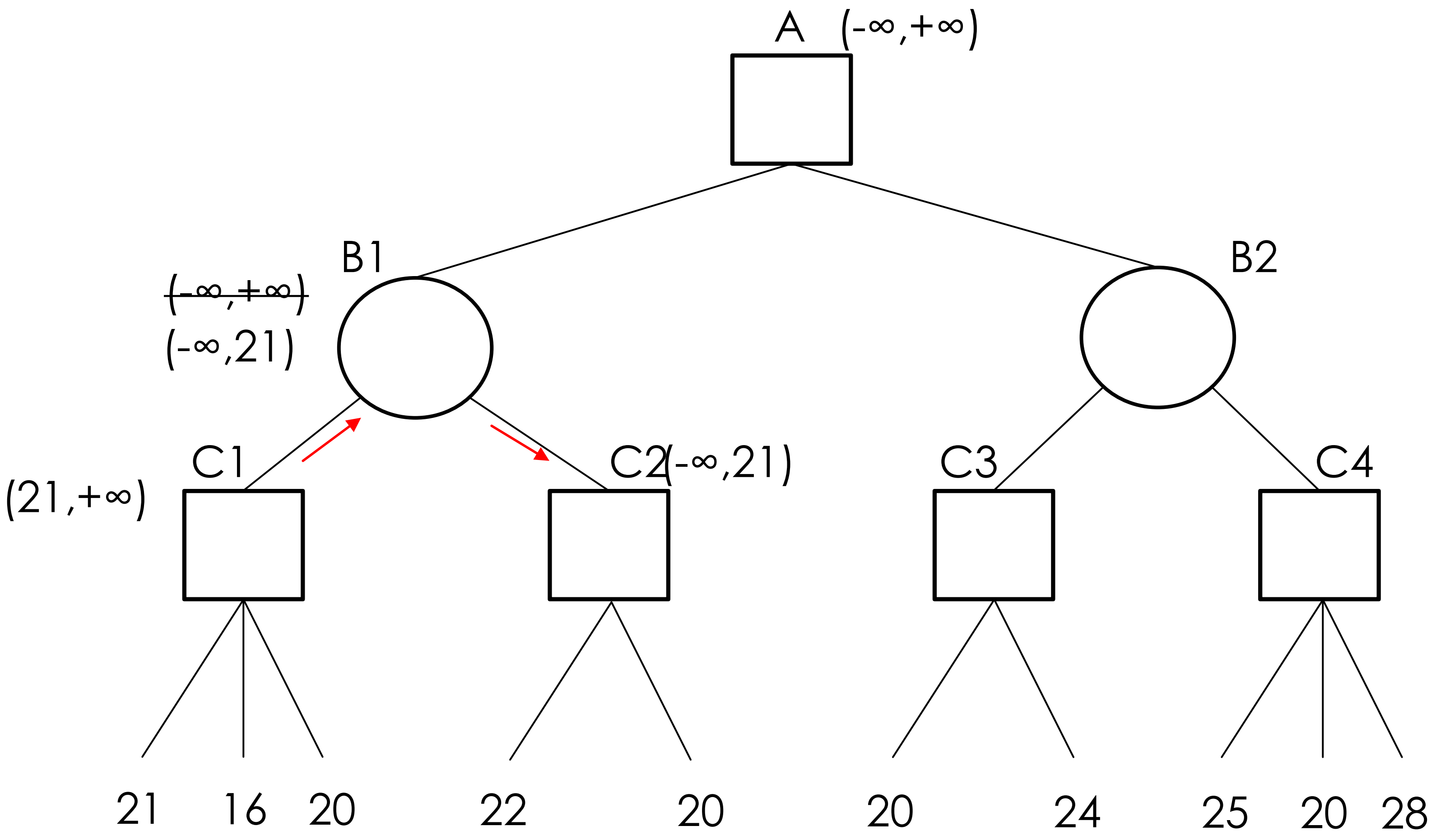
function MAX-VALUE(*state*, α, β) **returns** *a utility value*
  if TERMINAL-TEST(*state*) **then return** UTILITY(*state*)
  $v \leftarrow -\infty$
  for each a **in** ACTIONS(*state*) **do**
    $v \leftarrow$ MAX(v, MIN-VALUE(RESULT(s, a), α, β))
    if $v \geq \beta$ **then return** v
    $\alpha \leftarrow$ MAX(α, v)
  return v

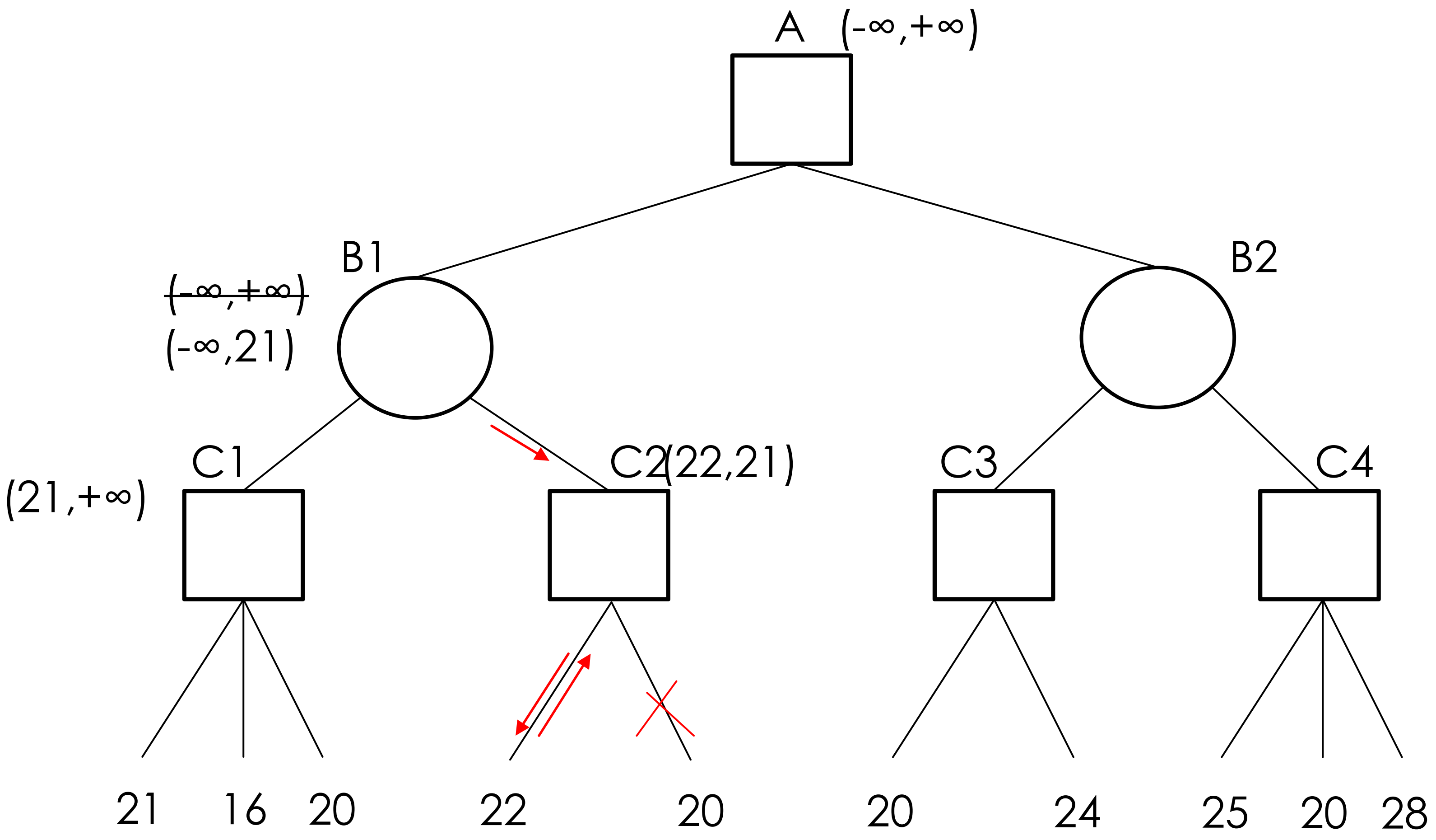
---

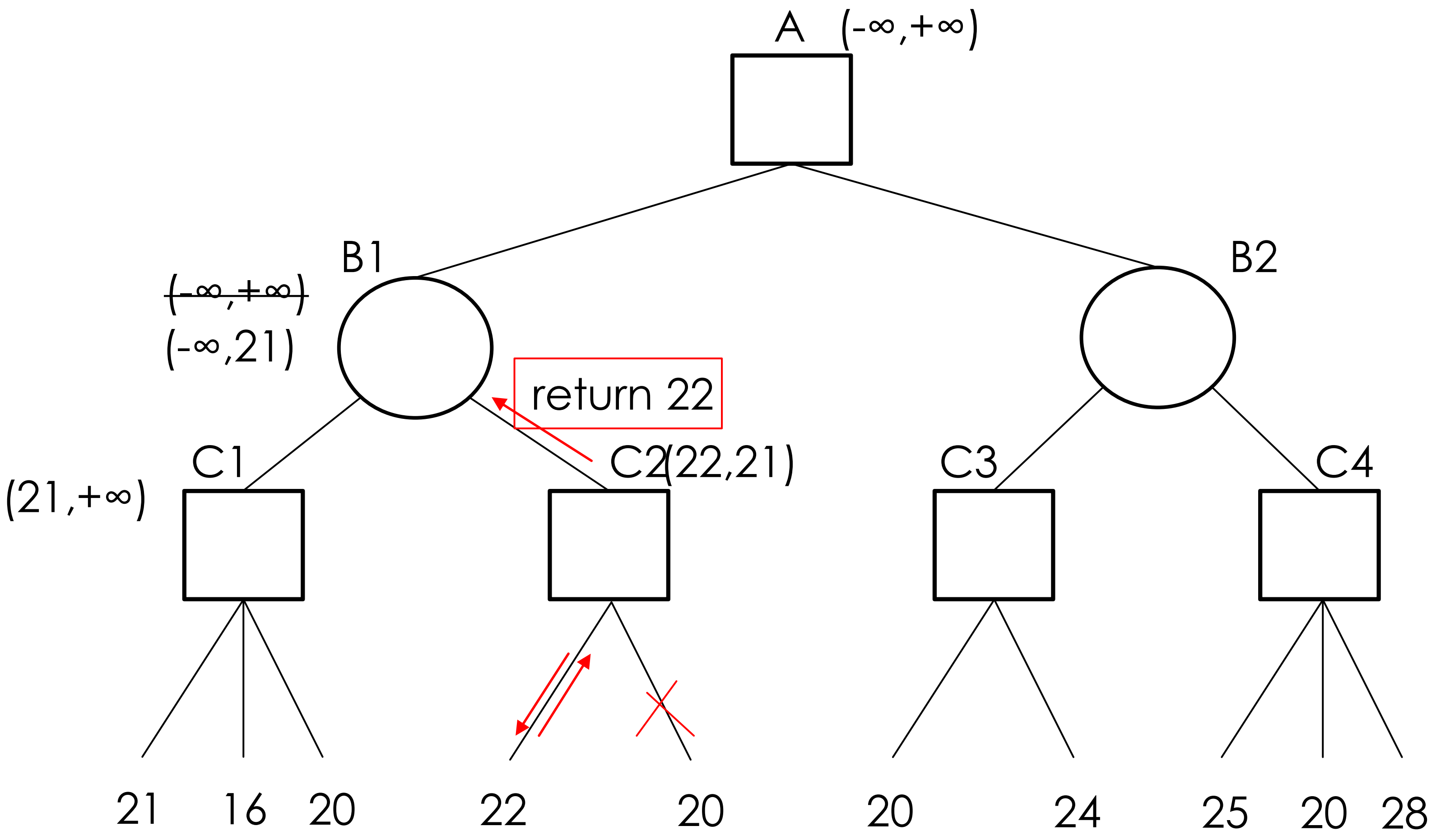
function MIN-VALUE(*state*, α, β) **returns** *a utility value*
  if TERMINAL-TEST(*state*) **then return** UTILITY(*state*)
  $v \leftarrow +\infty$
  for each a **in** ACTIONS(*state*) **do**
    $v \leftarrow$ MIN(v, MAX-VALUE(RESULT(s, a) , α, β))
    if $v \leq \alpha$ **then return** v
    $\beta \leftarrow$ MIN(β, v)
  return v

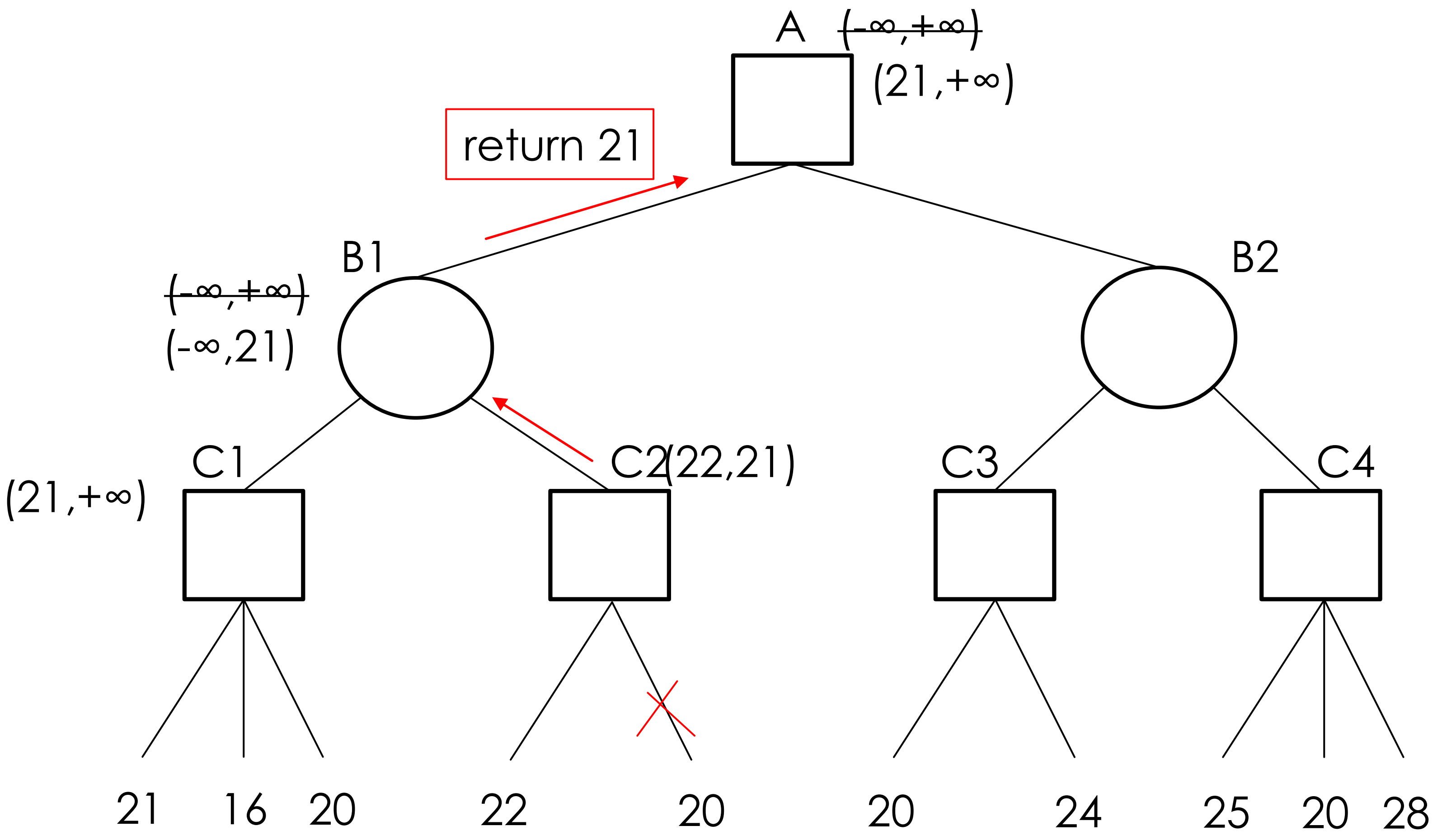
Question 2 Analysis

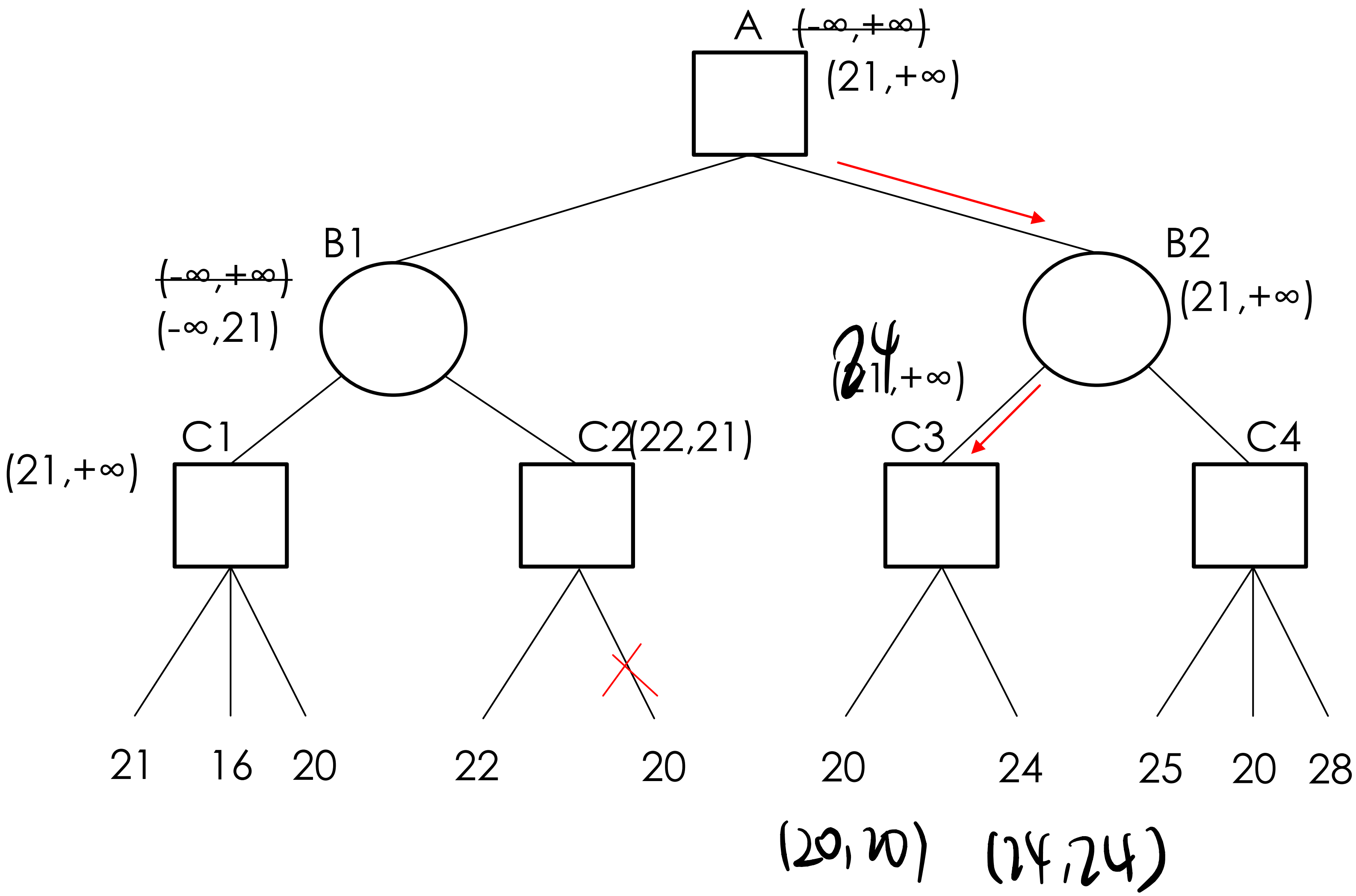


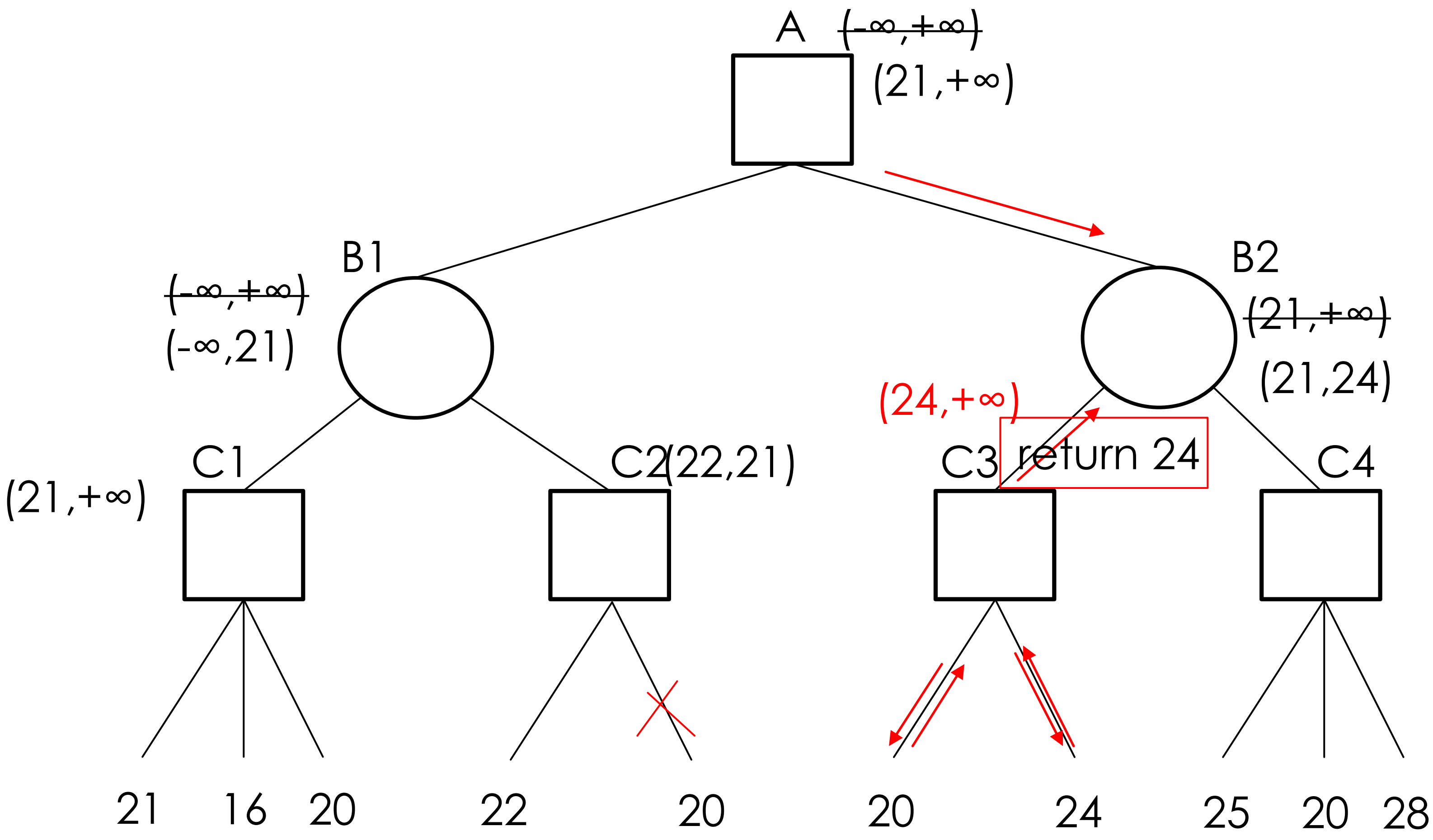

A (-∞,+∞)
B1
~~(-∞,+∞)~~
(-∞,21)
B2
C1
(21,+∞)
C2(-∞,21)
C3
C4
21 16 20
22 20
20 24
25 20 28


A (-∞,+∞)
B1
~~(-∞,+∞)~~
(-∞,21)
B2
C1
(21,+∞)
C2(22,21)
C3
C4
21
16
20
22
20
20
24
25
20
28


A (-∞,+∞)
B1
~~(-∞,+∞)~~
(-∞,21)
B2
return 22
C1
(21,+∞)
C2(22,21)
C3
C4
21
16
20
22
20
20
24
25
20
28


A
~~(-∞,+∞)~~
(21,+∞)
return 21
B1
~~(-∞,+∞)~~
(-∞,21)
B2
C1
(21,+∞)
C2
(22,21)
C3
C4
21
16
20
22
20
20
24
25
20
28


A
~~(-∞,+∞)~~
(21,+∞)
B1
~~(-∞,+∞)~~
(-∞,21)
B2
(21,+∞)
24
(21,+∞)
C1
(21,+∞)
C2(22,21)
C3
C4
21
16
20
22
20
20
24
25
20
28
(20,20)
(24,24)


A
~~(-∞,+∞)~~
(21,+∞)
B1
~~(-∞,+∞)~~
(-∞,21)
B2
~~(21,+∞)~~
(21,24)
(24,+∞)
return 24
C1
(21,+∞)
C2
(22,21)
C3
C4
21
16
20
22
20
20
24
25
20
28

Propositional Logic Review

YAO ZHAO

Key Points

- Transformation to Conjunctive Normal Form (CNF)
- SAT Problem → PDLL Algorithm

Transformation in to CNF: review

$B_{1,1} \Leftrightarrow (P_{1,2} \vee P_{2,1})$

1. Eliminate $\Leftrightarrow$, replacing $\alpha \Leftrightarrow \beta$ with $(\alpha \Rightarrow \beta) \wedge (\beta \Rightarrow \alpha)$.

   $(B_{1,1} \Rightarrow (P_{1,2} \vee P_{2,1})) \wedge ((P_{1,2} \vee P_{2,1}) \Rightarrow B_{1,1})$

2. Eliminate $\Rightarrow$, replacing $\alpha \Rightarrow \beta$ with $\neg\alpha \vee \beta$.

   $(\neg B_{1,1} \vee P_{1,2} \vee P_{2,1}) \wedge (\neg(P_{1,2} \vee P_{2,1}) \vee B_{1,1})$

3. Move $\neg$ inwards using de Morgan's rules and double-negation:

   $(\neg B_{1,1} \vee P_{1,2} \vee P_{2,1}) \wedge ((\neg P_{1,2} \wedge \neg P_{2,1}) \vee B_{1,1})$

4. Apply distributivity law ($\vee$ over $\wedge$) and flatten:

   $(\neg B_{1,1} \vee P_{1,2} \vee P_{2,1}) \wedge (\neg P_{1,2} \vee B_{1,1}) \wedge (\neg P_{2,1} \vee B_{1,1})$

The focus of logical expression

- The transformation to conjunctive normal form (CNF)
- The great significance of the conjunction paradigm is that it makes it possible to define a search space for a logical problem. Naturally, the logical problem can be transformed into a problem solved by backtracking.

SAT Problem

SAT:(Boolean) Satisfiability Problem

For example: Is it possible the following expression is true? Which model can make it being true?

```
~P11 & (B11 <=> (P12 | P21)) & (B21 <=>(P11 | P22 | P31)) &  ~B11 & B21
```

A model is an assignment of truth-value to variables making the above expression true.
A possible assignment is
[P11: False, B11: False, P12: False, P21: False, B21: True, P31: True]

Backtracking can be used to find the above assignment of truth-value to variable.

联想CSP中变量
以及对变量赋值的过程

DPLL Review

```
function DPLL-SATISFIABLE?(s) returns true or false
  inputs: s, a sentence in propositional logic

  clauses ← the set of clauses in the CNF representation of s
  symbols ← a list of the proposition symbols in s
  return DPLL(clauses, symbols, { })
```

```
function DPLL(clauses, symbols, model) returns true or false

  if every clause in clauses is true in model then return true
  if some clause in clauses is false in model then return false
  P, value ← FIND-PURE-SYMBOL(symbols, clauses, model)
  if P is non-null then return DPLL(clauses, symbols – P, model ∪ {P=value})
  P, value ← FIND-UNIT-CLAUSE(clauses, model)
  if P is non-null then return DPLL(clauses, symbols – P, model ∪ {P=value})
  P ← FIRST(symbols); rest ← REST(symbols)
  return DPLL(clauses, rest, model ∪ {P=true}) or
         DPLL(clauses, rest, model ∪ {P=false}))
```

Backtracking in DPLL

- If there is a clause returning false, then return false as a whole.
- If all clauses are determined to be true in the model, return true as a whole.
- Look for pure symbols first: For example, if there is only A in all clauses, you can directly assign A to true; if there is only ¬B in all clauses, you can directly assign B to false. We can ignore all clauses determined to be true with the built model.
- Find the unit clause in priority: clause with only one literal
  - a clause can also be considered as a unit clause if all other literals are assigned a value except one literal .

    E.g, A is a unit clause, and ¬A is also a unit clause.

    A has been assigned a value of false, A | ¬B is also a unit clause since B must be false here. When B is false, A | B | C is also a unit clause, because C must be true.

    The process is a bit like constraint propagation, called unit propagation.
- If none of the above symbols were found, look for the next symbol. Try all possible assignments of the symbol and recursively call the next layer.

Forward checking in CSP BTS

Minimum Remaining Values heuristic

Normal backtracking

DPLL Example

Transform into CNF and extract all conjunction sub-clauses

```
~P11 & (B11 <=> (P12 | P21)) & (B21 <=>(P11 | P22 | P31)) &  ~B11 & B21
```

~P11

~P12 | B11

~P21 | B11

~P22 | B21

~P31 | B21

~P11 | B21

~B11

B21

clauses

P12 | P21 | ~B11

P11 | P22 | P31 | ~B21

symbols:

P31, P22, B21, P12, P21, P11, B11

Extract each clause of the conjunction paradigm.
The entire expression is true if each clause is true.

DPLL Example(initial)

Put priority in selecting pure literal and unit clauses

~P11

~P12 | B11

~P21 | B11

~P22 | B21

~P31 | B21

~P11 | B21

~B11

B21

unknown clauses

P12 | P21 | ~B11

P11 | P22 | P31 | ~B21

symbols:

P31, P22, B21, P12, P21, P11, B11

model:

{}

DPLL Example

Put priority in selecting pure literal and unit clauses

Pure symbols?

Unit Clause

~P11

~P12 | B11

~P21 | B11

~P22 | B21

~P31 | B21

~~~P11 | B21~~

true

~B11

B21

unknown clauses

P12 | P21 | ~B11

P11 | P22 | P31 | ~B21

P22 | P31 | ~B21

symbols:

P31, P22, B21, P12, P21, ~~P11,~~ B11

model:

{P11:False}
~~~

DPLL Example

Unit Clause

~P11

~P12 | B11

~P21 | B11

~P22 | B21

~P31 | B21

~B11

B21

~P11 | B21

~P12

~P21

unknown clauses

P12 | P21 | ~B11

True

P11 | P22 | P31 | ~B21

P22 | P31 | ~B21

symbols:

P31, P22, B21, P12, P21, ~~P11, B11~~

model:

{P11:False, B11:False}

DPLL Example

~P11

~P11 | B21

~B11

P12 | P21 | ~B11

~P12 | B11 → ~P12

~P21 | B11 → ~P21

~P22 | B21 → True

~P31 | B21 → True

Unit Clause

B21

unknown clauses

P11 | P22 | P31 | ~B21

P22 | P31 | ~B21 → P22 | P31

symbols:

P31, P22, ~~B21~~, P12, P21, ~~P11, B11~~

model:

{P11:False, B11:False, B21:True}

DPLL Example

~P11

~P11 | B21

~B11

P12 | P21 | ~B11

~P22 | B21

~P31 | B21

B21

Unit Clause

unknown clauses

~P12 | B11

~P12

~P21 | B11

~P21

P11 | P22 | P31 | ~B21

P22 | P31 | ~B21

P22 | P31

symbols:

P31, P22, ~~B21~~, ~~P12~~, P21, ~~P11, B11~~

model:

{P11:False, B11:False, B21:True， P12:False}

DPLL Example

~P12 | B11

~P11

~P11 | B21

~B11

P12 | P21 | ~B11

~P22 | B21

~P31 | B21

B21

Unit Clause

~P21 | B11

~P21

unknown clauses

P11 | P22 | P31 | ~B21

P22 | P31 | ~B21

P22 | P31

symbols:

P31, P22, ~~B21~~, ~~P12~~, ~~P21~~, ~~P11, B11~~

model:

{P11:False, B11:False, B21:True,P12:False,P21:False}

DPLL Example

No pure literals and no unit clauses. Select a literal and assign it to be true

~P11

~P11 | B21

~P12 | B11

~B11

~P21 | B11

P12 | P21 | ~B11

~P22 | B21

~P31 | B21

B21

No Unit Clause, choose the first symbol in symbols.

unknown clauses

P11 | P22 | P31 | ~B21

P22 | P31 | ~B21 → P22 | P31

True

symbols:

Choose and set true

P31, ~~P22~~, ~~B21~~, ~~P12~~, ~~P21~~, ~~P11~~, ~~B11~~

model:

{P11:False, B11:False, B21:True,P12:False,P21:False,P22:True}

DPLL Example

No clauses and return model

~P11

~P11 | B21

~P12 | B11

~P21 | B11

~B11

P12 | P21 | ~B11

~P22 | B21

~P31 | B21

B21

P11 | P22 | P31| ~B21

unknown clauses

symbols:

P31, ~~P22~~, ~~B21~~, ~~P12~~, ~~P21~~, ~~P11, B11~~

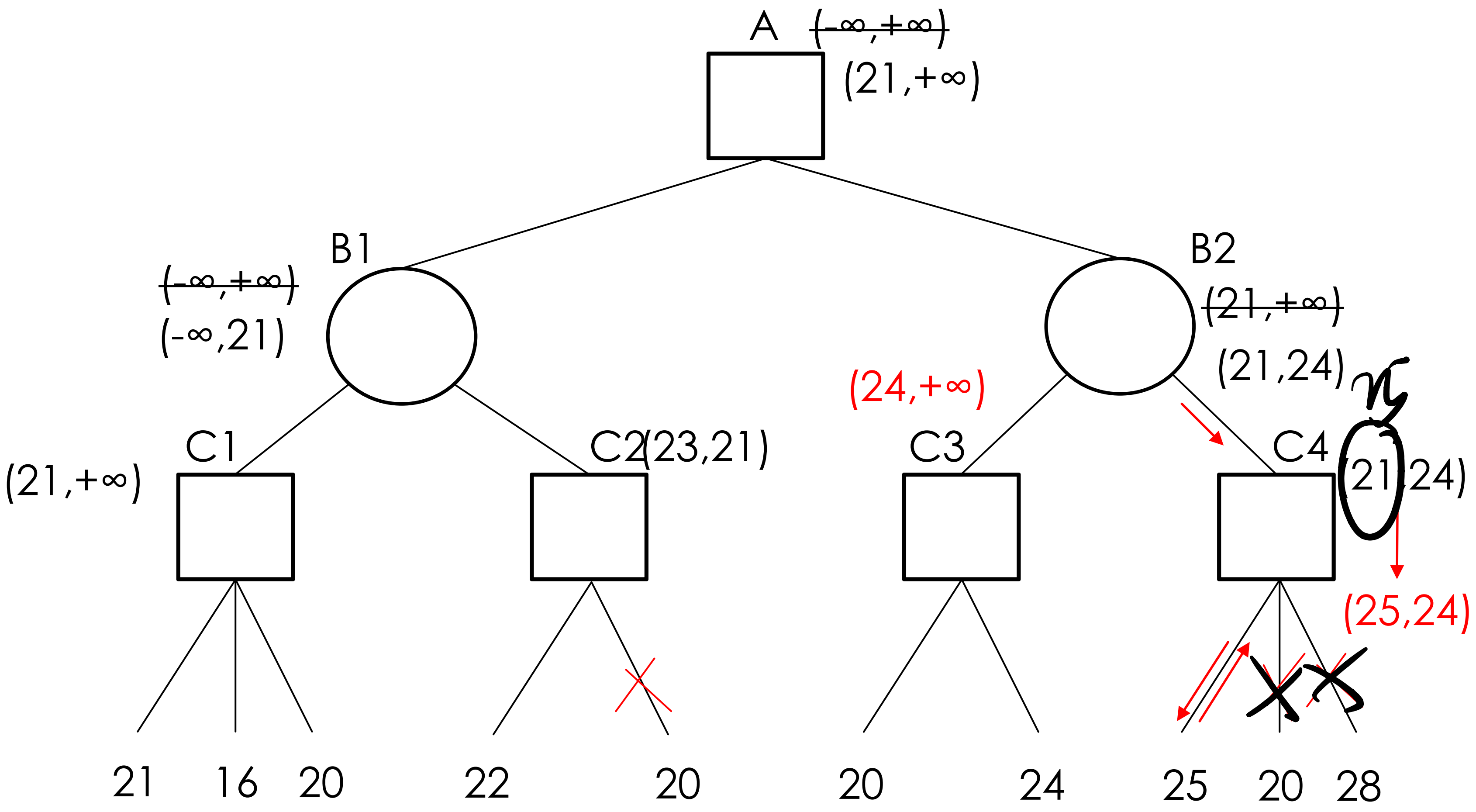
A solution

model:

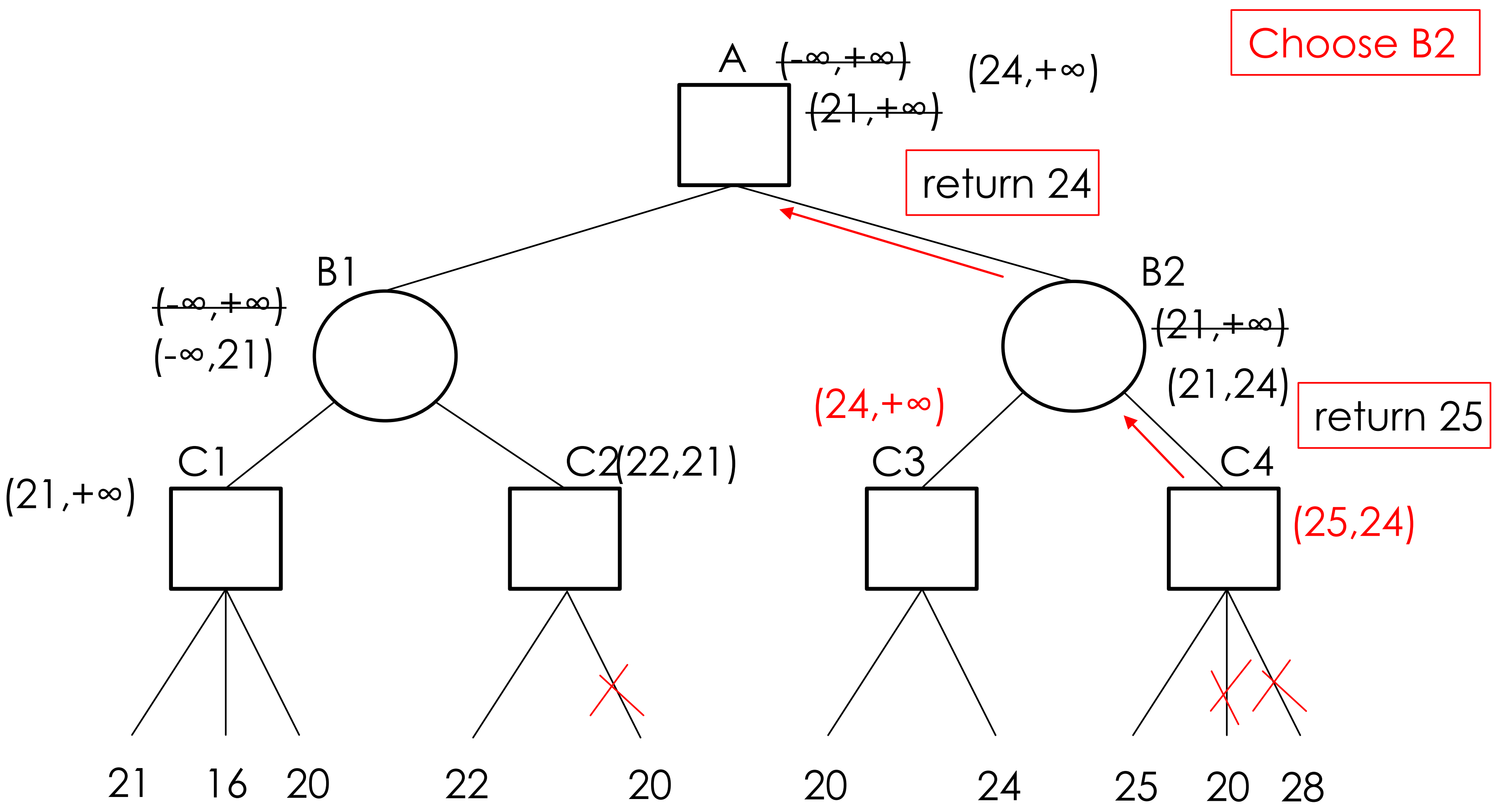
{P11:False, B11:False, B21:True,P12:False,P21:False,P22:True}

Summary

- The above example is relatively simple, but you can understand two points:
  - Early termination of backtracking, such as the last remaining symbol P31, but its value has not affected the result.
  - Preemptive selection of pure symbols and unit clauses, for the role of pruning , the process of the unit’s communication.
- For large problems, there are more directions for optimization.

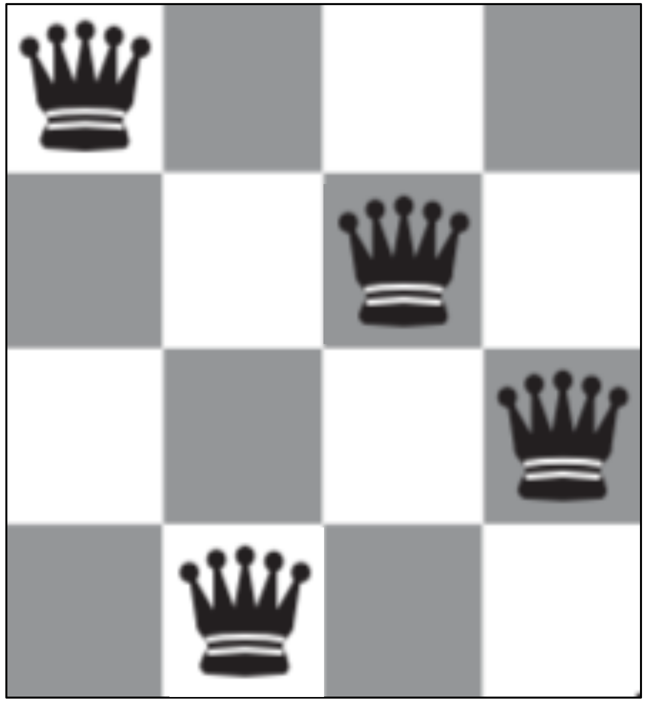


A
(-∞,+∞)
(21,+∞)
B1
(-∞,+∞)
(-∞,21)
B2
(21,+∞)
(21,24)
(24,+∞)
C1
(21,+∞)
C2
(23,21)
C3
C4
(21,24)
(25,24)
21
16
20
22
20
20
24
25
20
28


Choose B2
A
~~(-∞,+∞)~~
~~(21,+∞)~~
(24,+∞)
return 24
B1
~~(-∞,+∞)~~
(-∞,21)
B2
~~(21,+∞)~~
(21,24)
return 25
(24,+∞)
C1
(21,+∞)
C2
(22,21)
C3
C4
(25,24)
21
16
20
22
20
20
24
25
20
28

Question 3

Here is the initial status of a 4-queens problem. Using min-conflicts algorithm to solve this problem. Draw every step and tell why? (Finding one solution with the minimal number of steps can get full marks)



Min-Conflicts Algorithm

function MIN-CONFLICTS(csp, max_steps) **returns** a solution or failure
  inputs: csp, a constraint satisfaction problem
    max_steps, the number of steps allowed before giving up

  $current \leftarrow$ an initial complete assignment for csp
  for i = 1 to max_steps **do**
    if $current$ is a solution for csp **then return** $current$
    $var \leftarrow$ a randomly chosen conflicted variable from csp.VARIABLES
    $value \leftarrow$ the value v for var that minimizes CONFLICTS(var, v, $current$, csp)
    set $var = value$ in $current$
  return $failure$

Question 3 Answer

Step1:choose queen no.2

Q		1	
		Q	
		3	Q
	Q	2	

Q		Q	
			Q
	Q		

Step2:choose queen no.0

Q		Q	
0			
3			Q
1	Q		

		Q	
Q			
			Q
	Q		

Step3：no conflict queen, exit

Question 4

- You want to go to Germany as a tourist and visit the 9 cities Berlin, München, Frankfurt, Nürnberg, Hamburg, Stuttgart, Dresden, Eisenach, and Weimar (each at most once).Only Berlin, München, and Frankfurt have international airports, so you can **arrive in Germany and leave Germany ONLY from one of these three cities.**In Germany, you want to travel by train. All the 9 cities are directly connected by train. Each train connection has a **travel time (duration) and a rating**. The worst rating is 1 star, which means that the train ride is uninteresting and in an uncomfortable train. The best rating is 5 stars and it stands for comfortable train trips that go through a nice scenery.

How can you make your train trips **both as short and as pleasant as possible, AND feasible**?

Write something for each of the points below. If an element is not needed, say so!

- (1)What is a solution space in this context?
- (2)How would you represent such solutions internally for the optimization algorithms? In other words, how would you encode them and which search space would you use?
- (3)Define the objective function(s) and whether they are maximized or minimized. (a textual description or short pseudo-code is enough)
- (4)Define the search operations that you would employ. It is sufficient to shortly describe what they do.

Question 4 Answer

- (1)The solution space is the permutation of 9 cities in which the first city and the last city should be one among Berlin, München and Frankfurt. Permutation means one city only appear once.
- (2)A solution can be represented as a permutation of numbers from 1 to 9, each number denotes a city. The first number and last number in one solution should be one among 1, 2, 3.
- (3)There are two objective functions:

  total travel time (distance), to be minimized

  total travel rating, to be maximized
- (4)Possible search operations:

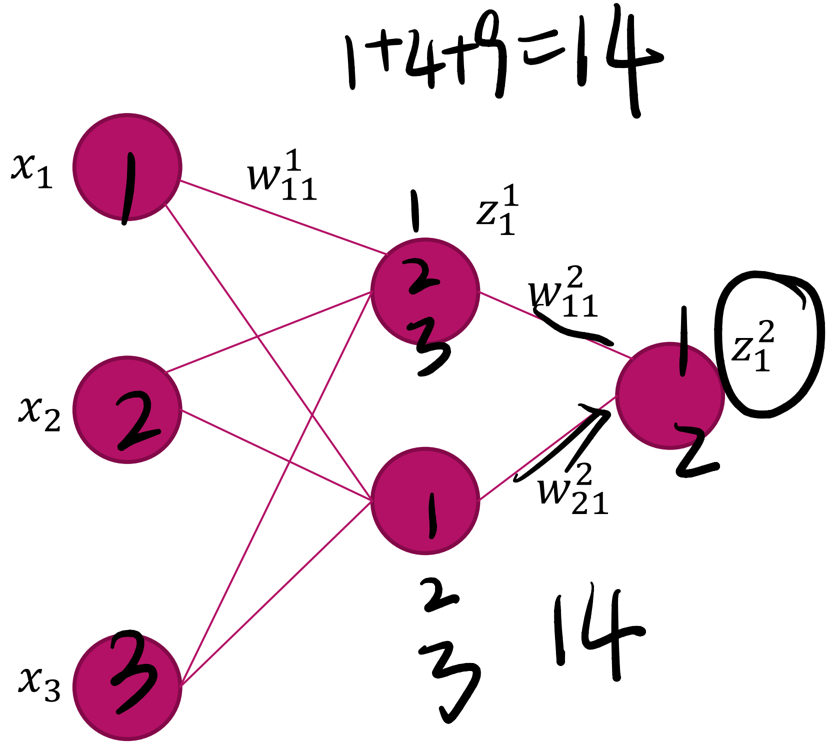
  Initialization: initialize lists of 1-9 and then randomly shuffle.

  Mutation: randomly choose two elements in one solution and exchange them.

  Repair: If the first element in one solution is not 1, 2, or 3, exchange the first element with the first appearing 1, 2, or 3 in the list. If the last element in one solution is not 1, 2, or 3, exchange the last element with the last appearing 1, 2 or 3 in the list.

Question 5

- Assume x = $[1\ 2\ 3]^T$, y = 3. $w^1 = \begin{bmatrix} 1 & 1 \\ 2 & 2 \\ 3 & 3 \end{bmatrix}$, $w^2 = \begin{bmatrix} 1 \\ 2 \end{bmatrix}$. Activity function for hidden layer and output layer is Relu: $f(x) = \max(0, x)$.
- (1)Please calculate the z_1^2.
- (2)We want to minimize $E(z_1^2) = (y - z_1^2)^2/2$, using gradient descent. Let the step size be $\alpha = 0.001$, which w^1 and w^2 will be at after ONE iteration of gradient descent?



Question 5 Answer

$W_1 = W_1 - \alpha \cdot \frac{\partial E}{\partial W_1}$

- (1) $z_1^1 = z_2^1$ = f(1+4+9) = 14

z_1^2 = f(14+28) = 42

$z_j = f(a_j)$

$a_j = W_i x$

- (2)

$f(x) = \max(0, x)$

$f'(x) = \begin{cases} if\ x > 0, f'(x) = 1 \\ \quad else\ f'(x) = 0 \end{cases}$

$\frac{\partial E}{\partial w_{ij}^{(2)}} = \frac{\partial E}{\partial z_j^{(2)}} \frac{\partial (z_j^{(2)})}{\partial a_j^{(2)}} \frac{\partial (a_j^{(2)})}{\partial w_{ij}^{(2)}}$ = -$(y - z_1^2)$*1*$(z_i^{(1)})$

$\frac{\partial E}{\partial w_{ij}^{(1)}} = \frac{\partial E}{\partial z_1^{(2)}} \frac{\partial (z_1^{(2)})}{\partial a_1^{(2)}} \frac{\partial (a_1^{(2)})}{\partial z_j^{(1)}} \frac{\partial (z_j^{(1)})}{\partial (a_j^{(1)})} \frac{\partial (a_j^{(1)})}{\partial w_{ij}^{(1)}}$ = -$(y - z_1^2)$*1* $w_{j1}^{(2)} * 1 * x_i$

$w^2 = w^2$ -0.001* $\frac{\partial E}{\partial w_{ij}^{(2)}}$ = $\begin{bmatrix}1\\2\end{bmatrix}$ – 0.001*(-1)(3-42) * $\begin{bmatrix}14\\14\end{bmatrix}$ = $\begin{bmatrix}1\\2\end{bmatrix}$ - $\begin{bmatrix}0.546\\0.546\end{bmatrix}$ = $\begin{bmatrix}0.454\\1.454\end{bmatrix}$

$w^1 = w^1$ - 0.001* $\frac{\partial E}{\partial w^{(1)}}$ = = $\begin{bmatrix}1 & 1\\2 & 2\\3 & 3\end{bmatrix} - 0.001 * (-1) *$(3-42) *$\begin{bmatrix}1\\2\\3\end{bmatrix}$ *[1 2] = $\begin{bmatrix}1 & 1\\2 & 2\\3 & 3\end{bmatrix}$ - $\begin{bmatrix}0.039 & 0.078\\0.078 & 0.156\\0.117 & 0.234\end{bmatrix}$= $\begin{bmatrix}0.961 & 0.922\\1.922 & 1.844\\2.883 & 2.766\end{bmatrix}$

Question 6

- Consider a support vector machine (SVM) with decision boundary $w^T x + b = 0$ for a 3D feature space. The weight vector is $w = [3\ 2\ 1]^T$ and $b = 2$.

(1) Which of the following points will be classified *incorrectly* by this SVM in training process?

(a) $x_1 = [0\ 0\ 0]^T$, $y_1 = -1$

(b) $x_2 = [1\ 1\ 1]^T$, $y_2 = +1$

(c) $x_3 = [-1.88\ \ -2.99\ \ \ 11]^T$, $y_3 = +1$

(2) If the above w and b were obtained with the above three points, what is the range of the slack variables ξ_i?

Hint: the constraints are $y_i(w^T x_i + b) \geq 1 - \xi_i (\xi_i \geq 0)$.

Question 6 Answer

- (1)

(a) y_1 *(3*0+2*0+1*0+2)= -2 < 1 *incorrect*

(b) y_2*(3*1+2*1+1*1+2) = 8 ≥ 1 *correct*

(c) y_3*(3*(-1.88)+2*(-2.99)+1*11+2) = 1.38 ≥ 1 *correct*

(2) $\xi_1 \geq 3$ $\xi_2 \geq 0$ $\xi_3 \geq 0$