TD1 : Représentation et Résolution de problèmes par recherche Solution

Exercice: 1

Donner un état initial, un test de l'état final, une fonction successeur et une fonction de coût à chacun des problèmes suivants. Choisissez une formulation suffisamment précise pour qu'elle puisse être implémentée.

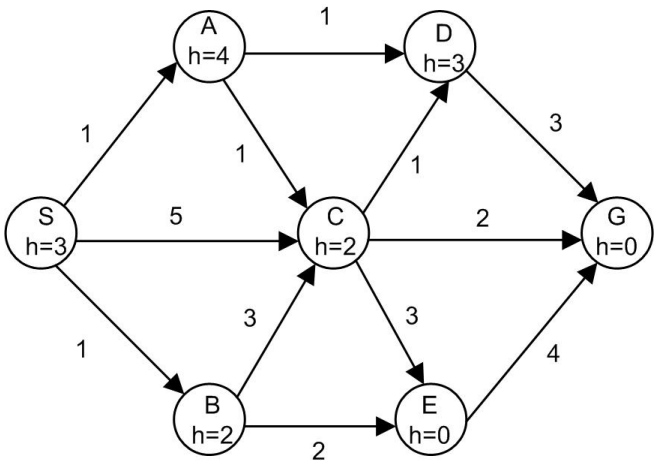
A. Un singe mesurant 90 cm se trouve dans une pièce dans laquelle des bananes sont suspendues à un plafond de 2.7 m de hauteur. Le singe aimerait bien avoir les bananes. La pièce contient deux caisses de 90 cm de hauteur qu'il est possible d'empiler, de déplacer et d'escalader.

a. Un état est décrit par la position (x1,y1,z1) de la caisse 1, de la position (x2, y2,z2) de la caisse 2 et de la position (x3, y3,z3) du singe :
   E=[x1, y1,z1, x2, y2,z2, x3, y3,z3],
   tq x1,x2,x3 ∈ {a1, a2, a, 0}
   et y1, y2 , y3 ∈ {b1, b2, b, 0} et
   z1, z2, z3 ∈ {0, 90, 180, 270}
b. L'état initial est : [a1,b1,c1,a2,b2,c2, a,b,c], avec (a, b,c)≠(0,0,270) position des bananes
c. L'état final est : [0,0,z1,0,0,z2,0,0,270]
d. Fonction SUCCESSEUR (E) : application d'un des opérateurs suivants
   i. Singe-Deplace-caisse1-vers-caisse2([x1,y1,0,x2,y2 ,0,x1,y1,0]) :
      retourner [x2,y2,0,x2,y2 ,0,x2,y2,0]
   ii. Singe-Deplace-caisse2-vers-caisse1([x1,y1,0,x2,y2 ,0,x2,y2,0]) :
      retourner [x1,y1,0,x1,y1 ,0,x1,y1,0]
   iii. Singe-Deplace-caisse1-vers-bananes([x1,y1,0,x2,y2 ,z2,x1,y1,0]) :
      retourner [0,0,0,x2,y2 ,z2,0,0,0]
   iv. Singe-Deplace-caisse2-vers-bananes([x1,y1,z1,x2,y2 ,0,x2,y2,0]) :
      retourner [x1,y1,z1,0,0,0,0,0,0]
   v. Singe-Empile-caisse2-sur-caisse1([x1,y1,0,x1,y1 ,0,x1,y1,0]) :
      retourner [x1,y1,0,x1,y1 ,90,x1,y1,0]
   vi. Singe-Empile-caisse1-sur-caisse2([x2,y2,0,x2,y2 ,0,x2,y2,0]) :
      retourner [x2,y2,90,x2,y2 ,0,x2,y2,0]
   vii. Singe-Escalade-caisse-1([x1,y1,z1,x2,y2 ,z2,x1,y1,z1]) :
      retourner [x1,y1,z1,x2,y2 ,z2,x1,y1, (z1+90)]
   viii. Singe-Escalade-caisse-2([x1,y1,z1,x2,y2 ,z2,x2,y2,z2]) :
      retourner [x1,y1,z1,x2,y2 ,z2,x2,y2, (z2+90)]
   ix. Singe-se-deplace-vers-caisse-1([x1,y1,z1,x2,y2 ,z2,x3,y3,0]) :
      retourner [x1,y1,z1,x2,y2 ,z2,x1,y1,0]
   x. Singe-se-deplace-vers-caisse-2([x1,y1,z1,x2,y2 ,z2,x3,y3,0]) :
      retourner [x1,y1,z1,x2,y2 ,z2,x2,y2,0]
e. F(nœud)= distance parcourue par singe,

B. considerer le problème de transporter N personnes à travers une rivière, où chaque personne a un certain poids. Tout le monde se trouve à la rive droite de la rivière et toutes les personnes doivent être transportées à la rive gauche. Supposez qu'il existe qu'une seule barque d'une capacité de WB kilos.

Exercice: 2

Considérer le graphe suivant :



Tracer l'exécution de A* pour le graphe ci-dessous et énumérer les nœuds successifs dans la liste OUVERTS.
Donner le chemin menant au but trouvé par A*.
Dans le graphe ci-dessus, S est l'état initial, G est l'état final, chaque arc est étiqueté par un coût correspondant à la valeur de la fonction g et chaque nœud est étiqueté par la valeur de h.

Dans la liste OUVERTS chaque élément doit être sous la forme (s,a) où s est l'état actuel et a est la valeur de la fonction $f = g + h$ où g est le coût du chemin de l'état initial au nœud s et h est la fonction heuristique étiquetant le nœud s.

Open	**Closed**
[(S, 3, null)]	[ ]
[(B,3,S),(A,5,S),(C,7,S)]	[(S, 3, null)]
[(E,3,B),**(C,6,B)**,(A,5,S)]	[(B,3,S),(S, 3, null)]
[(C,6,B),(A,5,S),(G,7,E)]	[(E,3,B),(B,3,S),(S, 3, null)]

Open	Closed
[(A,5,S),**(G,6,C),**(D,8,C)]	[(C,6,B),(E,3,B),(B,3,S),(S, 3, null)]
[**(C,4,A)**,(D,5,A),(G,6,C)]	[(A,5,S),(E,3,B),(B,3,S),(S, 3, null)]
[**(G,4,C),**(D,5,A)]	[(D,5,A),(A,5,S),**(C,4,A)**,(E,3,B),(B,3,S),(S, 3, null)]
Solution : [**(G,4,C), (C,4,A),(A,5,S)**]	

Exercice 3 :

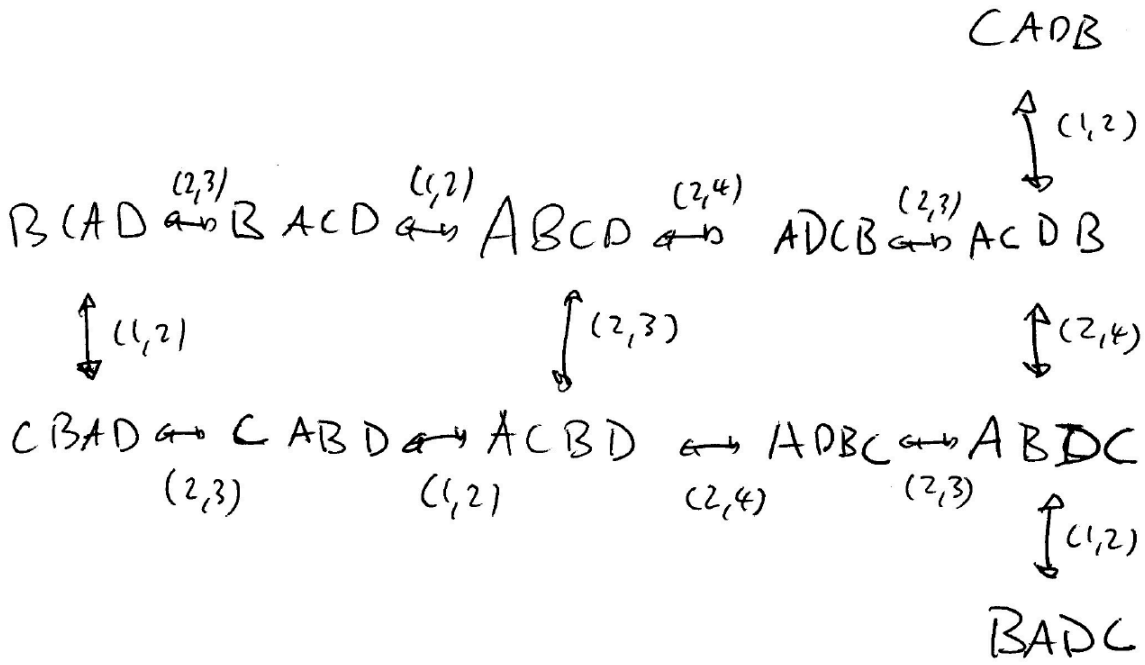
Nous considérons un monde avec 4 pions (A,B,C,D) non superposables. Ils peuvent être arrangés dans n'importe quel ordre, sauf A qui ne peut pas être plus à droite que D. Par exemple, ABCD et CBAD sont deux états possibles du monde, tandis que DCBA et CDAB ne sont pas possibles. Le monde peut être manipulé par une action de la forme échange(x, y) qui échange les pions des positions x et y. Par exemple échange(1, 2) transforme BCAD dans CBAD. Seules les actions échange(1, 2), échange(2, 3) et échange(2, 4) sont autorisées. Ils donnent un successeur uniquement si la situation atteinte est possible.

1. Dessinez le graphe d'états.

2. On suppose que l'´etat de départ est ADBC et l'´etat que l'on veut atteindre est CBAD. On suppose que chaque action coute 1. Donnez une "bonne" heuristique h admissible (mais aussi différente de 0 pour les noeuds non-finaux) pour ce problème. Le principe de l'heuristique devrait être suffisamment général pour pouvoir s'appliquer à des problèmes similaires.

3. Appliquez la recherche gloutonne (du meilleur premier) avec votre heuristique. Si vous n'avez pas trouvé d'heuristique, utilisez l'heuristique h = 0. Ne considérez pas les noeuds déjà développés. En cas d'égalité, choisissez un noeud à développer au hasard.

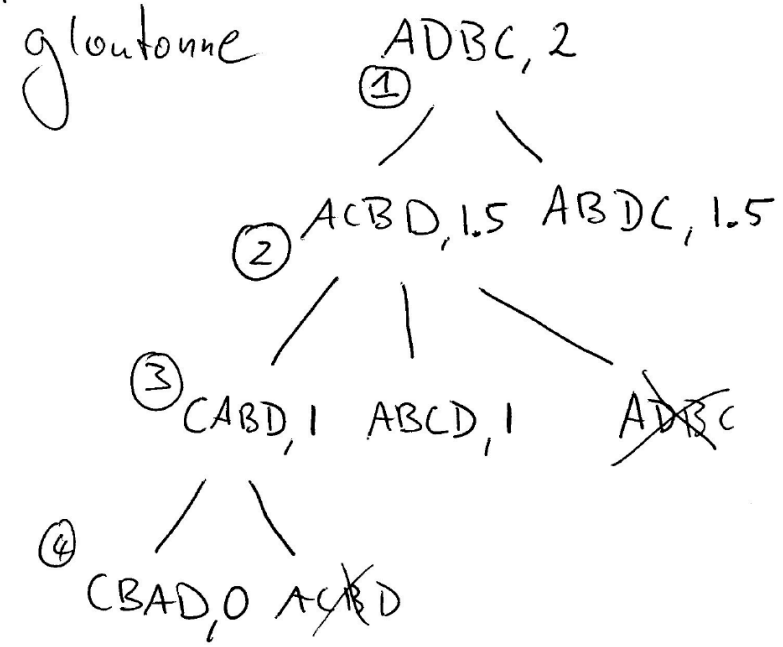
4. Appliquez la recherche A* avec votre heuristique. Si vous n'avez pas trouvé d'heuristique, utilisez l'heuristique h = 0. Ne considérez pas les noeuds déjà développés. En cas d'égalité choisissez un noeud à développer au hasard.



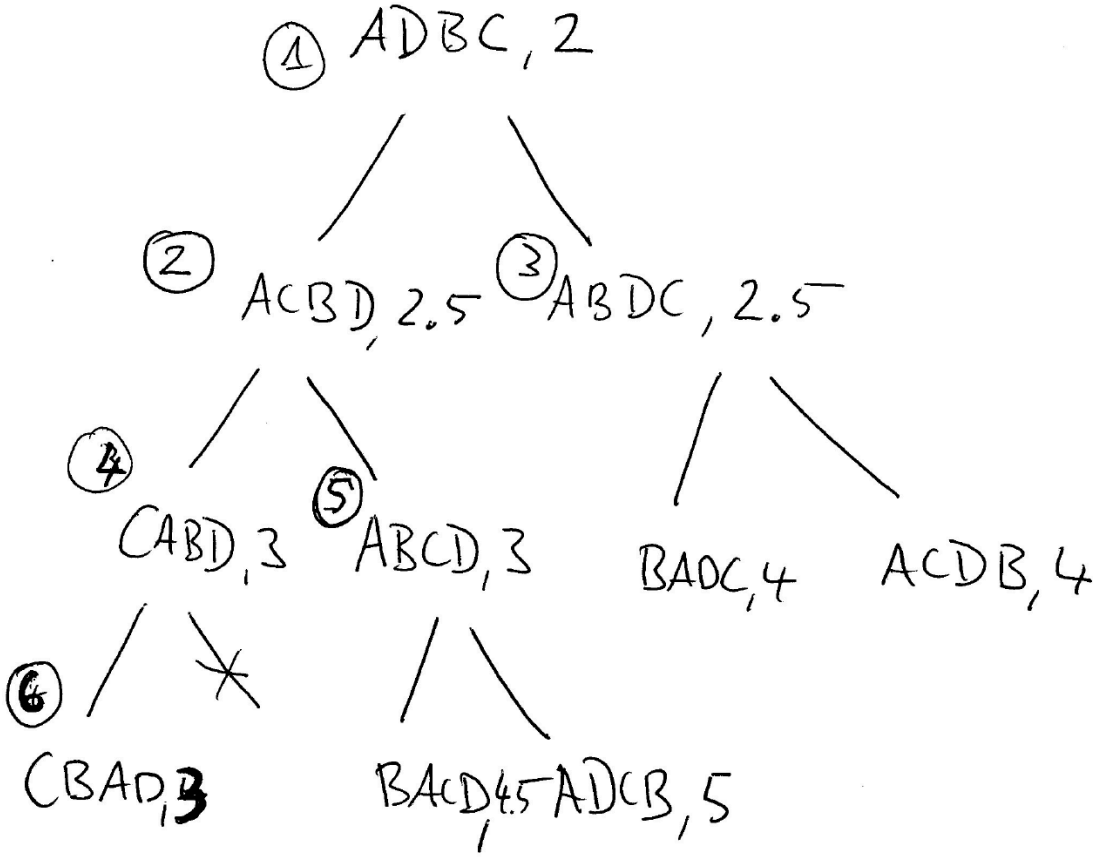
h : Le nombre de pions mal placés par rapport à CBAD (l'état final)

~~2~~

Recherche gloutonne

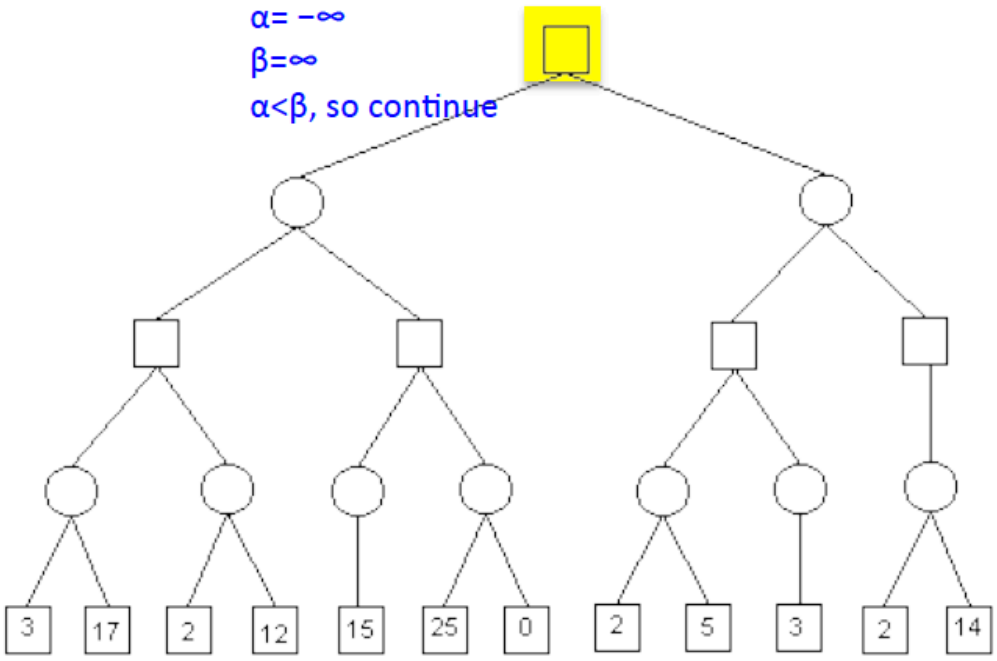


Recherche A^*

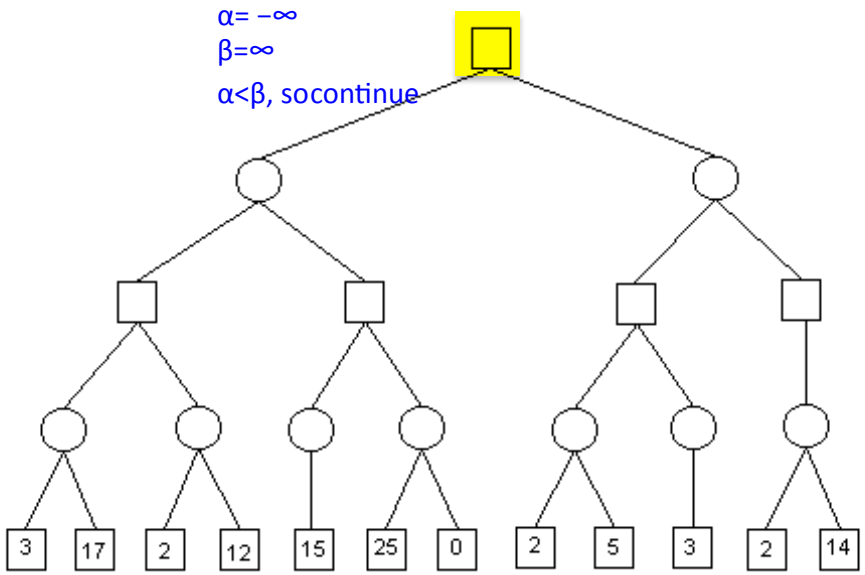

1 ADBC, 2
2 ACBD, 2.5
3 ABDC, 2.5
4 CABD, 3
5 ABCD, 3
BADC, 4
ACDB, 4
6 CBAD, 3
BACD, 4.5
ADCB, 5

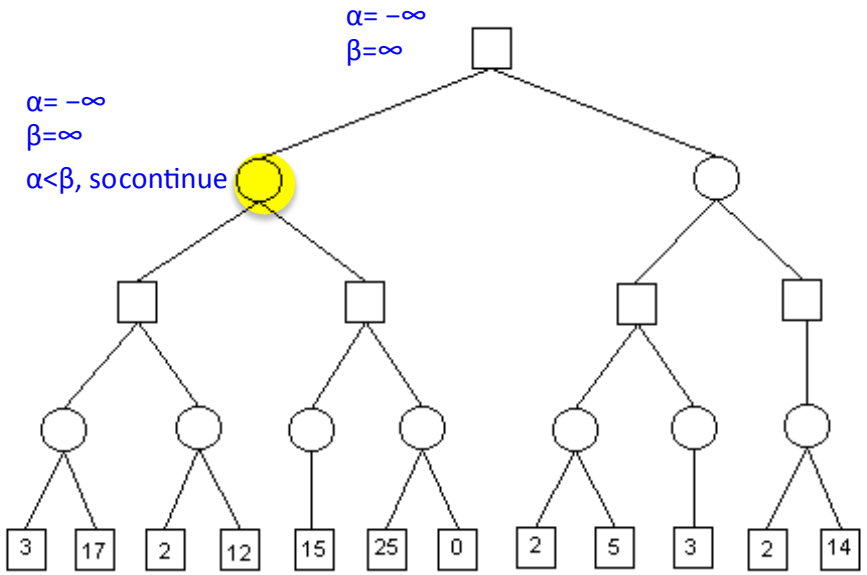
Exercice 4 :

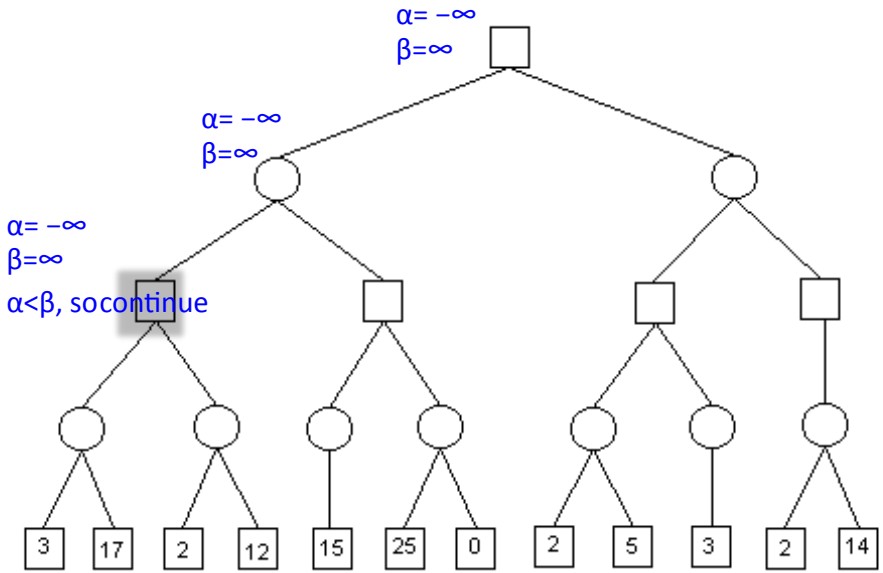
Considérez l'arbre de jeu suivant

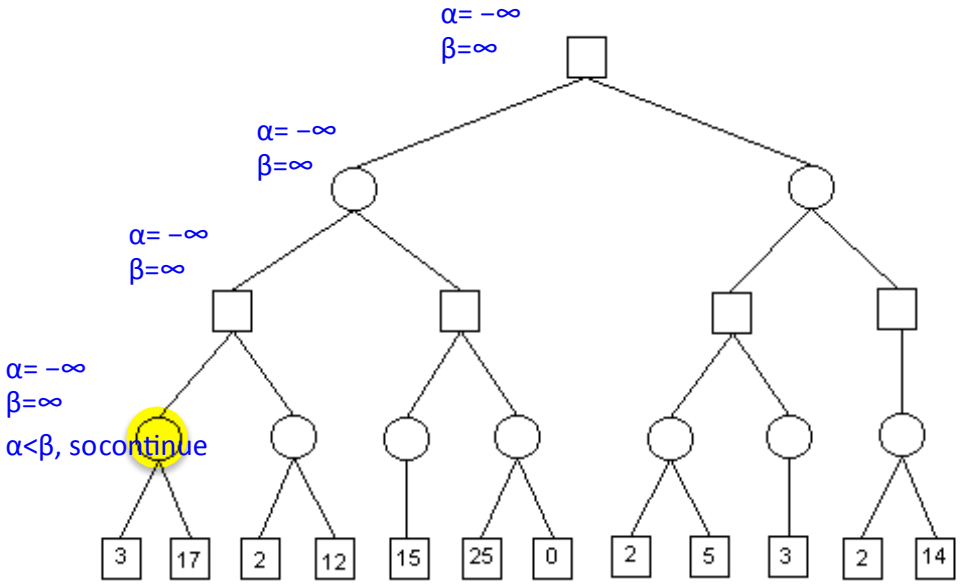


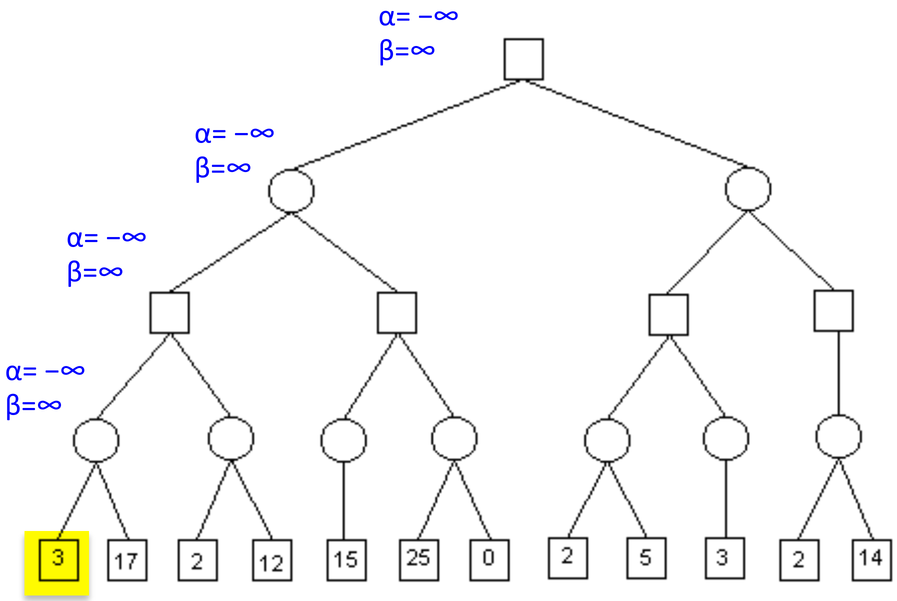
1. Donner le résultat suite à une exploration avec algorithme MINMAX ?
2. Donner le résultat suite à une exploration avec algorithme alpha-beta ?
3. Ordonner les descendants des noeuds à la profondeur 3 par ordre croissant et donner le résultat par une exploration avec algorithme alpha-beta ? Que conclure ?

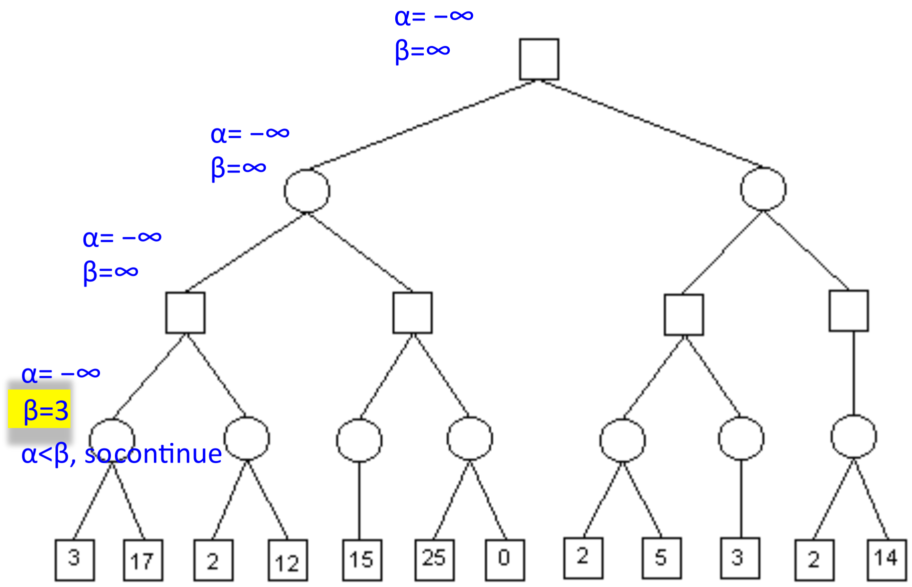

α= −∞
β=∞
α<β, socontinue
3
17
2
12
15
25
0
2
5
3
2
14

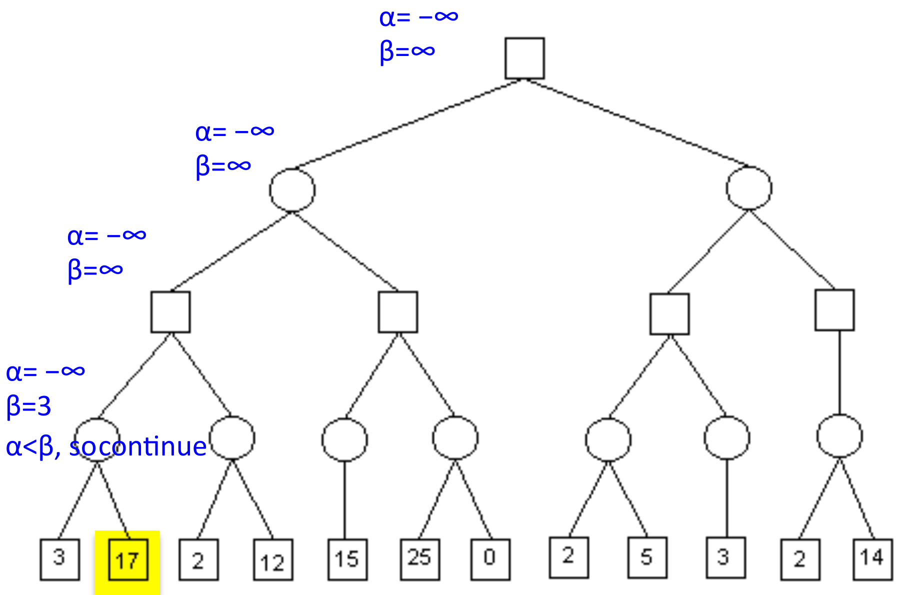

α= −∞
β=∞
α= −∞
β=∞
α<β, socontinue
3
17
2
12
15
25
0
2
5
3
2
14

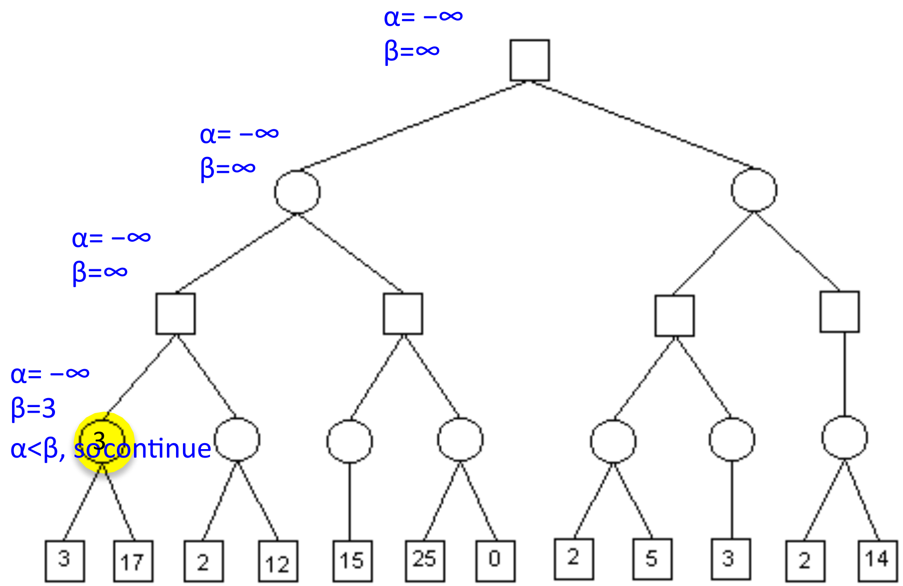

α= −∞
β=∞
α= −∞
β=∞
α= −∞
β=∞
α<β, socontinue
3
17
2
12
15
25
0
2
5
3
2
14

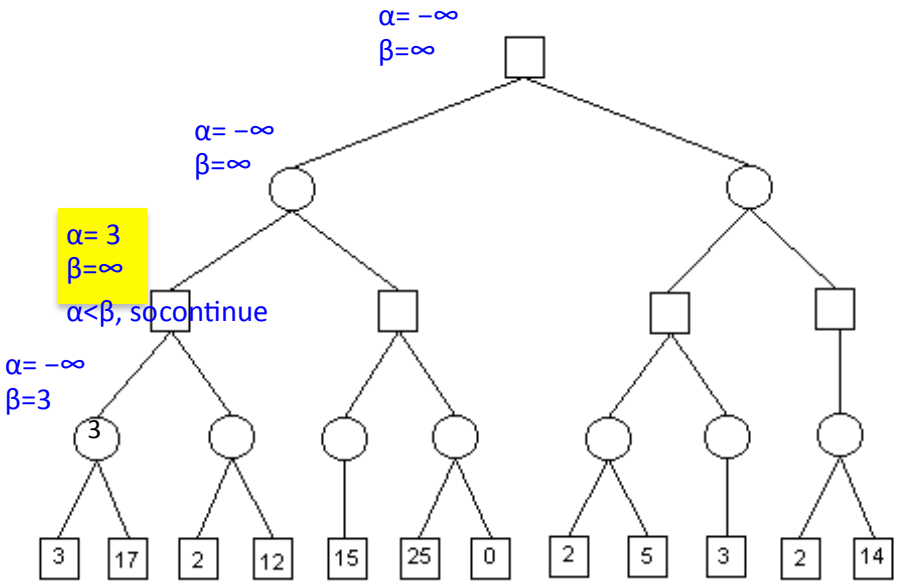

α= −∞
β=∞
α= −∞
β=∞
α= −∞
β=∞
α= −∞
β=∞
α<β, socontinue
3
17
2
12
15
25
0
2
5
3
2
14

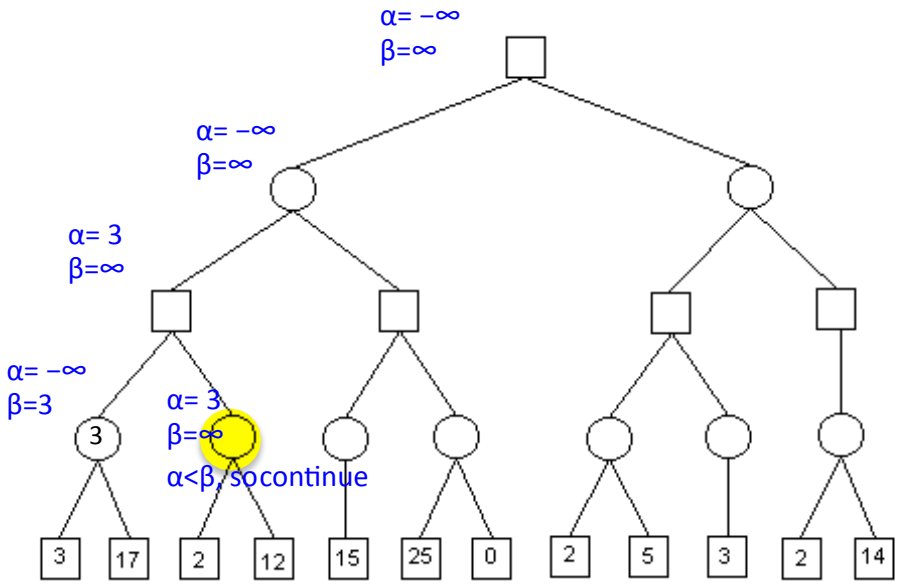

α= −∞
β=∞
α= −∞
β=∞
α= −∞
β=∞
α= −∞
β=∞
3 17 2 12 15 25 0 2 5 3 2 14

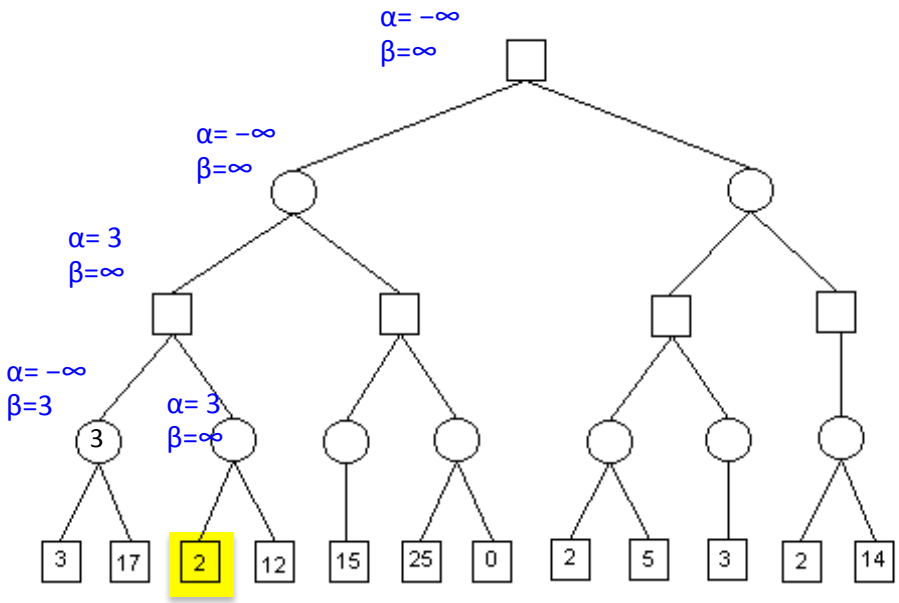

α= −∞
β=∞
α= −∞
β=∞
α= −∞
β=∞
α= −∞
β=3
α<β, so continue
3 17 2 12 15 25 0 2 5 3 2 14

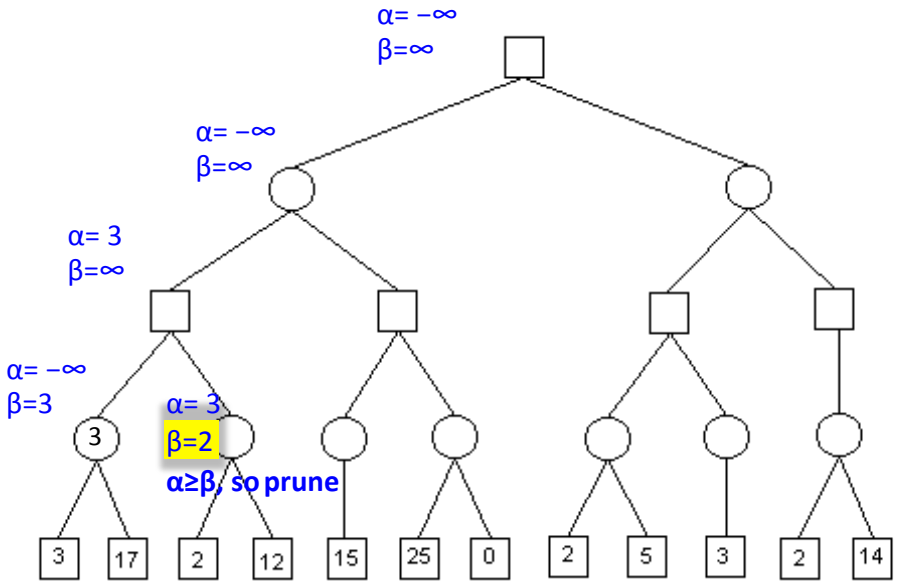

α= −∞
β=∞
α= −∞
β=∞
α= −∞
β=∞
α= −∞
β=3
α<β, so continue
3 17 2 12 15 25 0 2 5 3 2 14

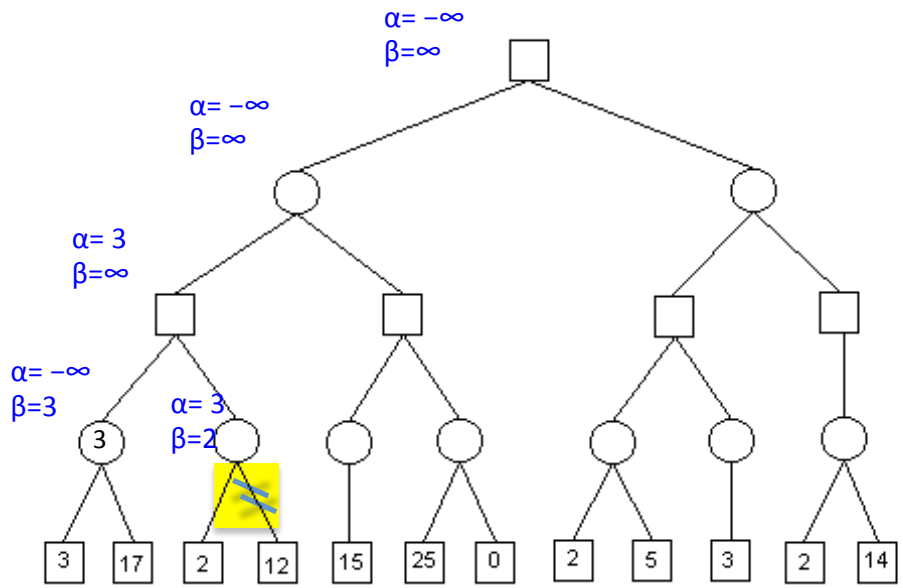

α= −∞
β=∞
α= −∞
β=∞
α= −∞
β=∞
α= −∞
β=3
α<β, so continue
3
3 17 2 12 15 25 0 2 5 3 2 14

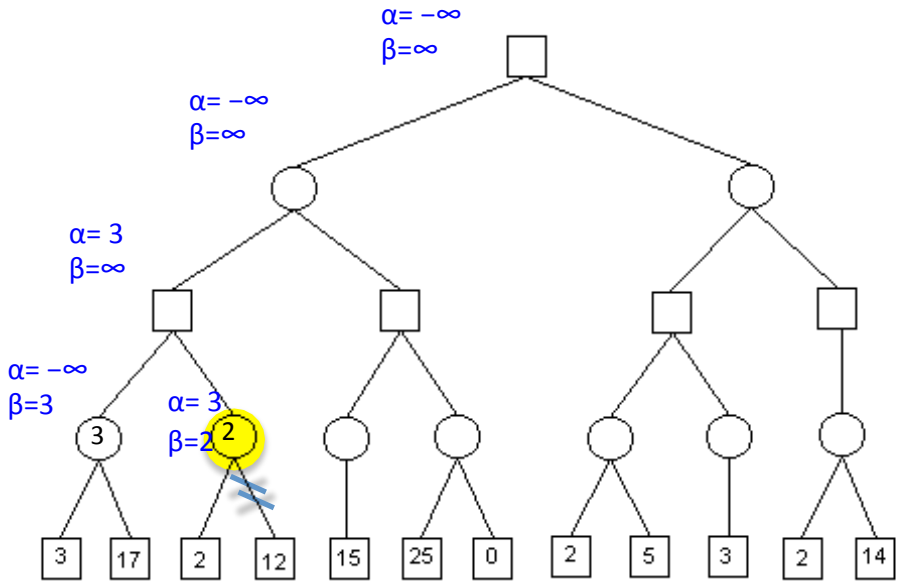

α= −∞
β=∞
α= −∞
β=∞
α= 3
β=∞
α<β, so continue
α= −∞
β=3
3
3
17
2
12
15
25
0
2
5
3
2
14

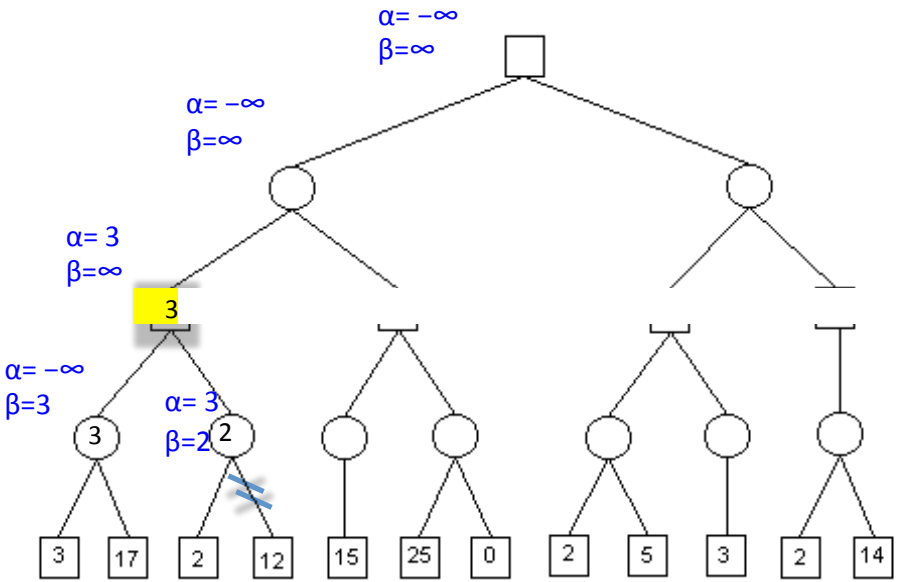

α= −∞
β=∞
α= −∞
β=∞
α= 3
β=∞
α= −∞
β=3
3
α= 3
β=∞
α<β, so continue
3
17
2
12
15
25
0
2
5
3
2
14

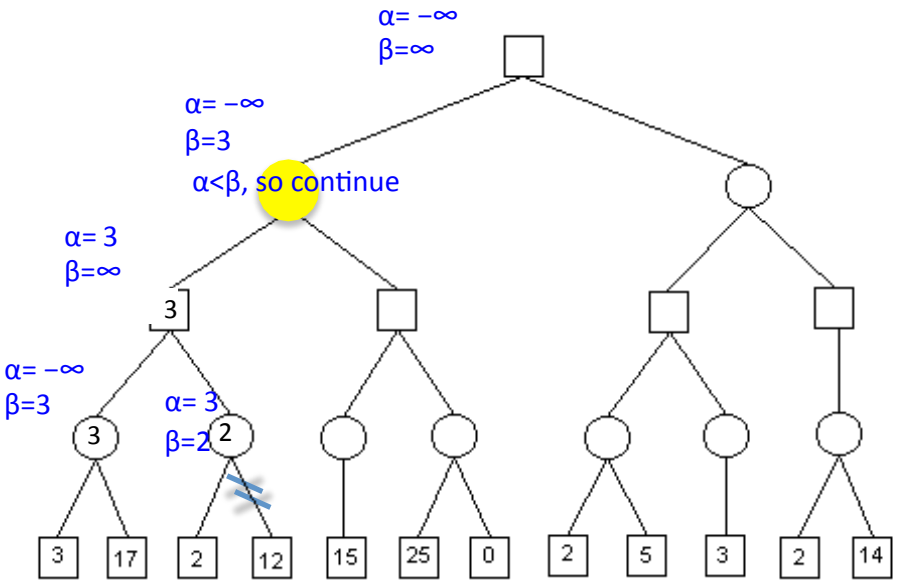

α= −∞
β=∞
α= −∞
β=∞
α= 3
β=∞
α= −∞
β=3
3
α= 3
β=∞
3
17
2
12
15
25
0
2
5
3
2
14

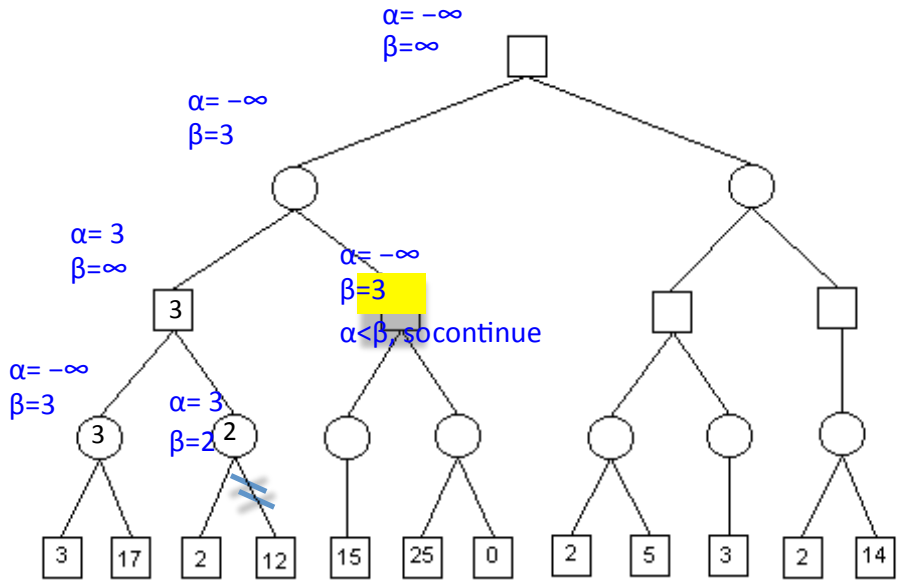

α= −∞
β=∞
α= −∞
β=∞
α= 3
β=∞
α= −∞
β=3
3
α= 3
β=2
α≥β, so prune
3
17
2
12
15
25
0
2
5
3
2
14

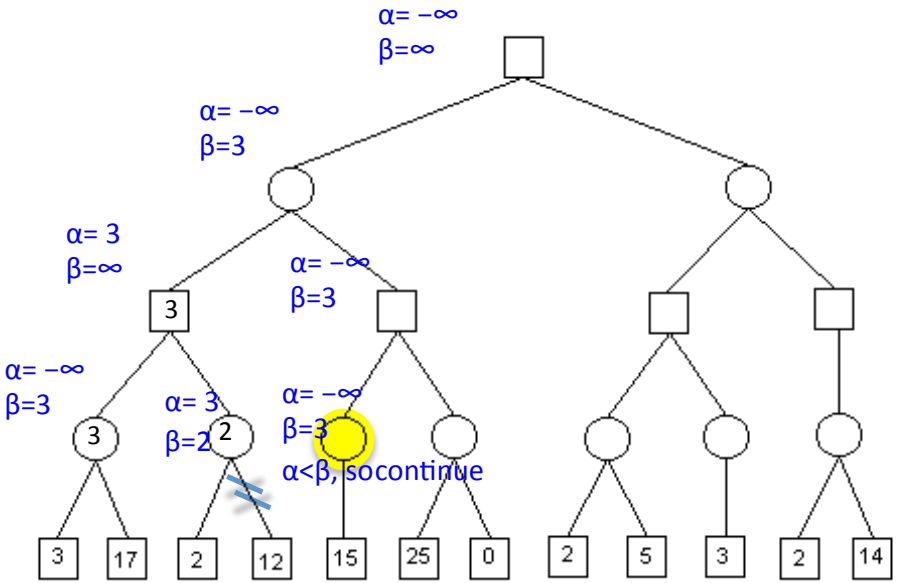

α= −∞
β=∞
α= −∞
β=∞
α= 3
β=∞
α= −∞
β=3
α= 3
β=2
3
3 17 2 12 15 25 0 2 5 3 2 14

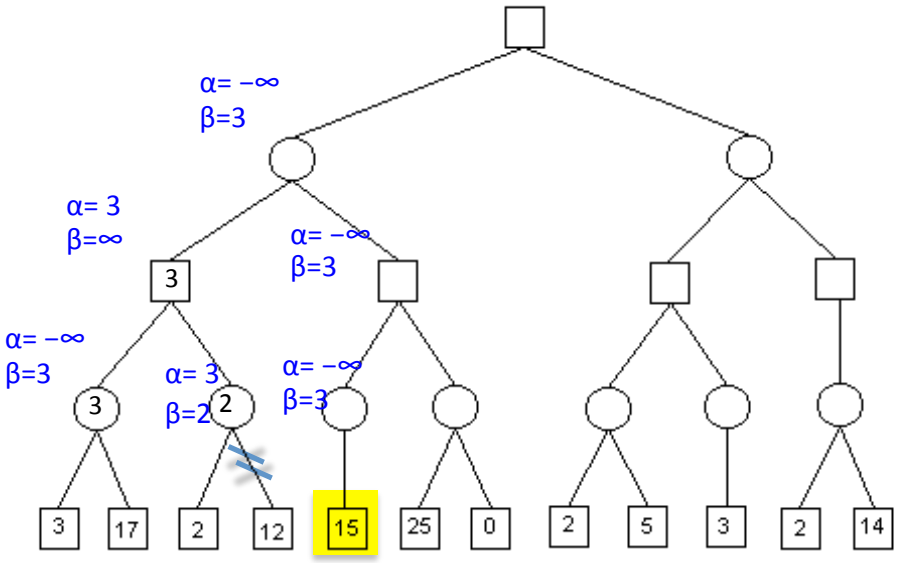

α= −∞
β=∞
α= −∞
β=∞
α= 3
β=∞
α= −∞
β=3
α= 3
β=2
3
2
3 17 2 12 15 25 0 2 5 3 2 14

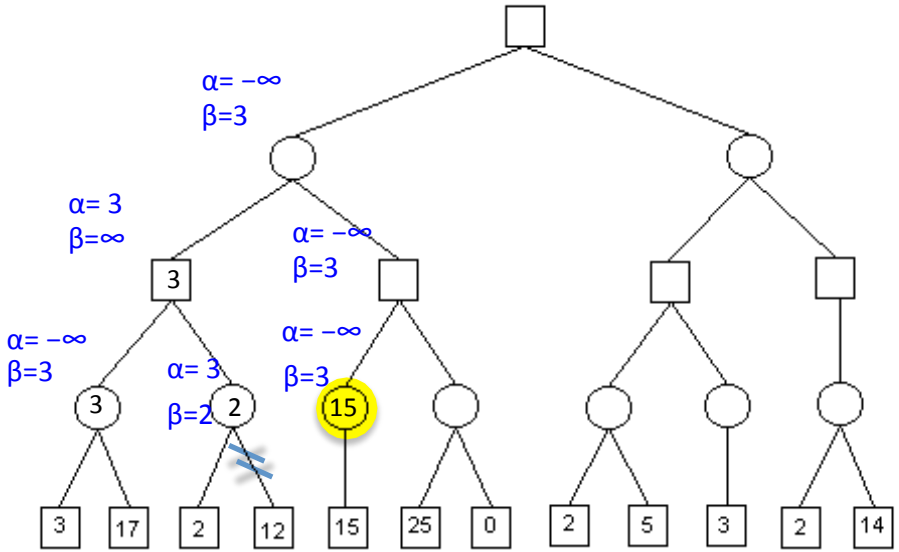

α= −∞
β=∞
α= −∞
β=∞
α= 3
β=∞
3
α= −∞
β=3
α= 3
β=2
3
2
3 17 2 12 15 25 0 2 5 3 2 14

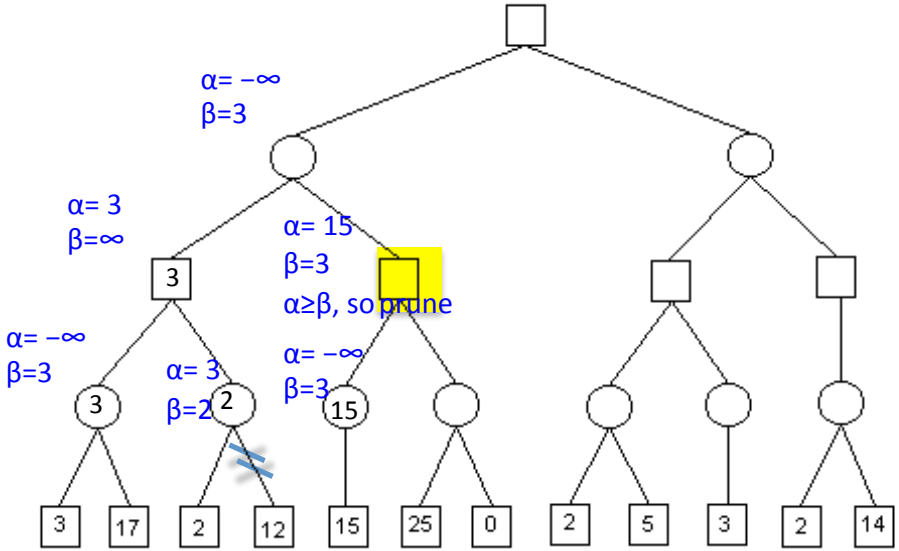

α= −∞
β=∞
α= −∞
β=3
α<β, so continue
α= 3
β=∞
3
α= −∞
β=3
α= 3
β=2
3
2
3 17 2 12 15 25 0 2 5 3 2 14

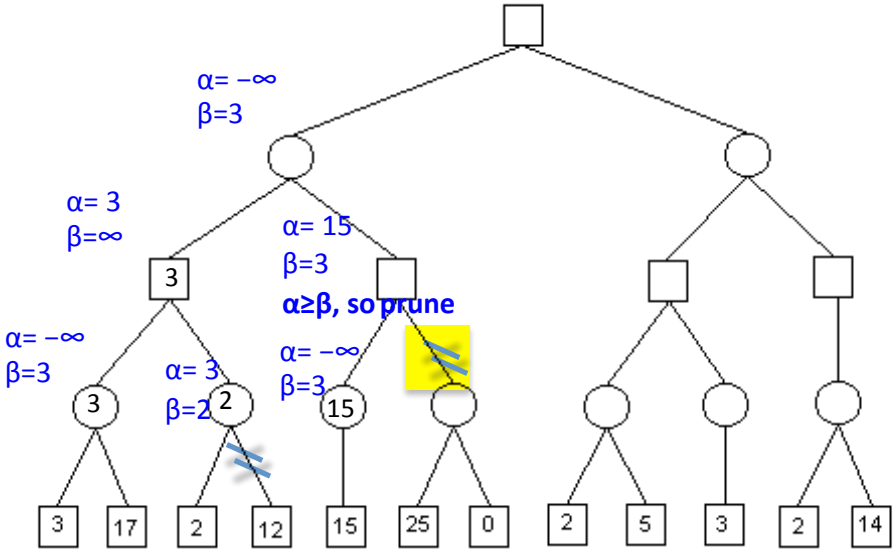

α= −∞
β=∞
α= −∞
β=3
α= 3
β=∞
α= −∞
β=3
α<β, socontinue
α= −∞
β=3
α= 3
β=2
3
17
2
12
15
25
0
2
5
3
2
14

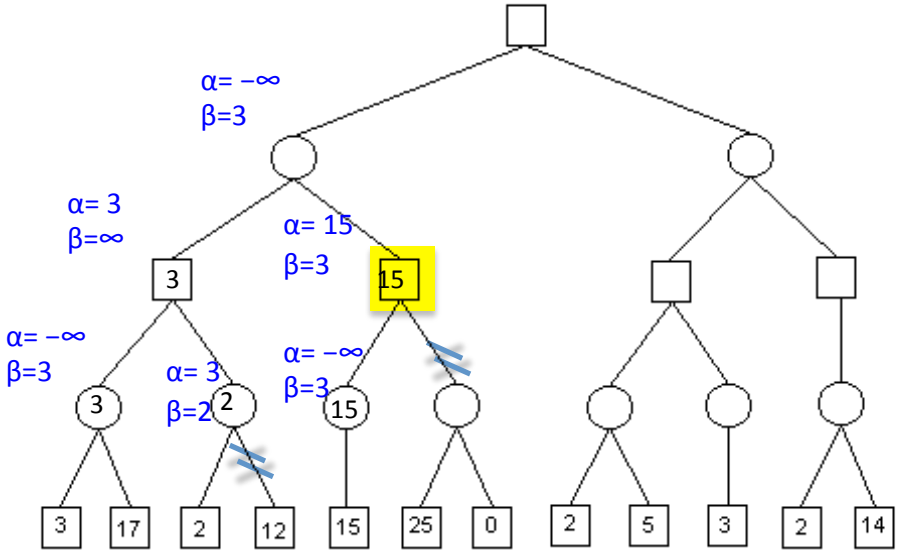

α= −∞
β=∞
α= −∞
β=3
α= 3
β=∞
α= −∞
β=3
α<β, socontinue
α= −∞
β=3
α= 3
β=2
3
17
2
12
15
25
0
2
5
3
2
14

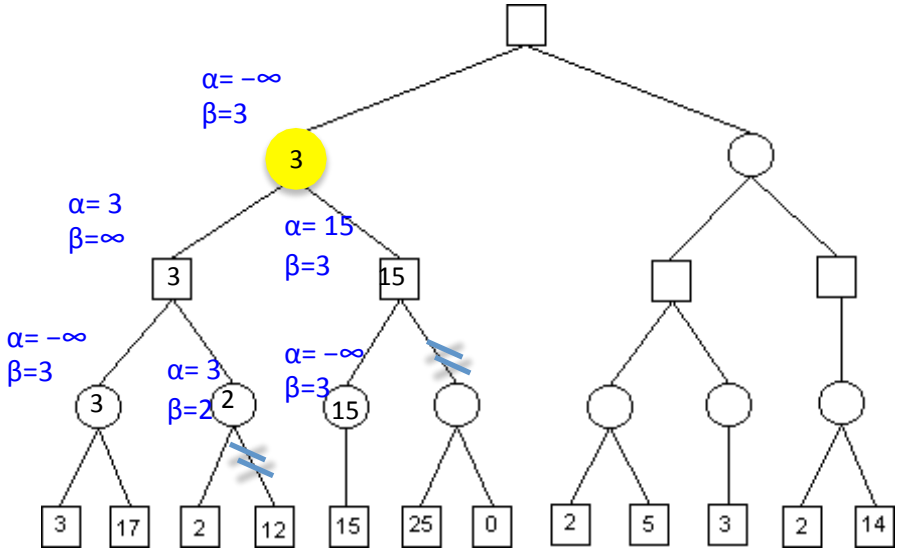

α= −∞
β=3
α= 3
β=∞
α= −∞
β=3
α= −∞
β=3
α= 3
β=2
3
17
2
12
15
25
0
2
5
3
2
14

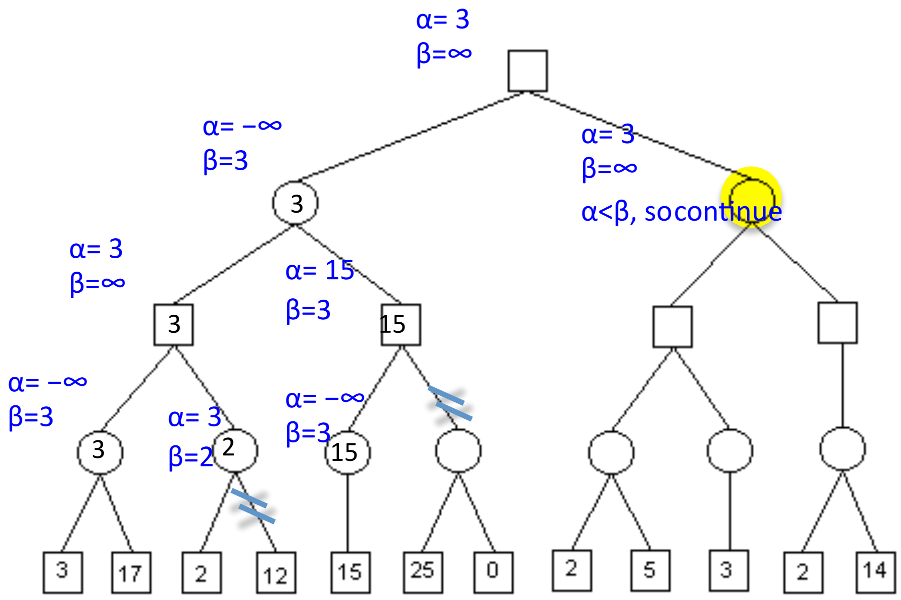

α= −∞
β=3
α= 3
β=∞
α= −∞
β=3
α= −∞
β=3
α= 3
β=2
15
3
17
2
12
15
25
0
2
5
3
2
14

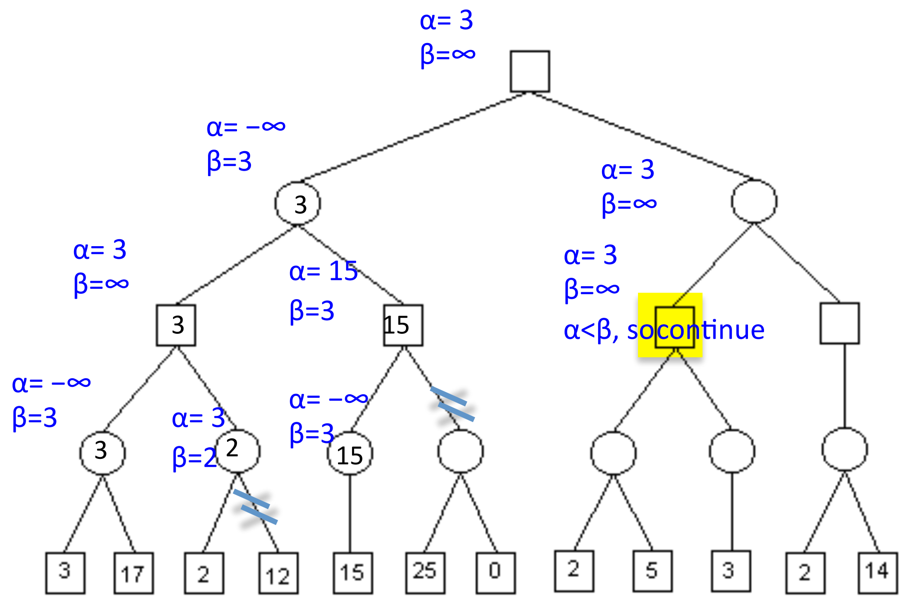

α= −∞
β=3
α= 3
β=∞
α= 15
β=3
α≥β, so prune
α= −∞
β=3
α= 3
β=2
α= −∞
β=3
3
2
15
3
17
2
12
15
25
0
2
5
3
2
14

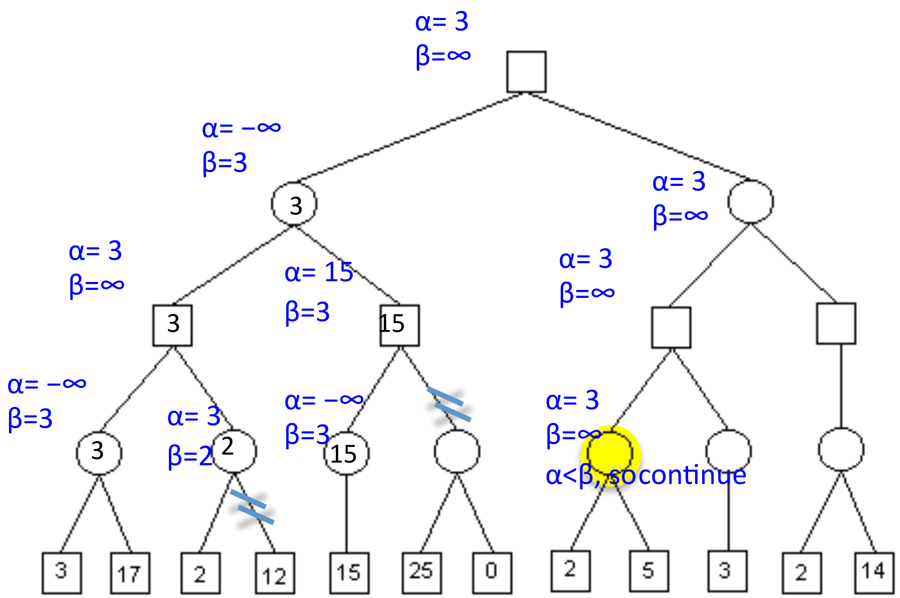

α= −∞
β=3
α= 3
β=∞
α= 15
β=3
α≥β, so prune
α= −∞
β=3
α= 3
β=2
α= −∞
β=3
3
2
15
3
17
2
12
15
25
0
2
5
3
2
14

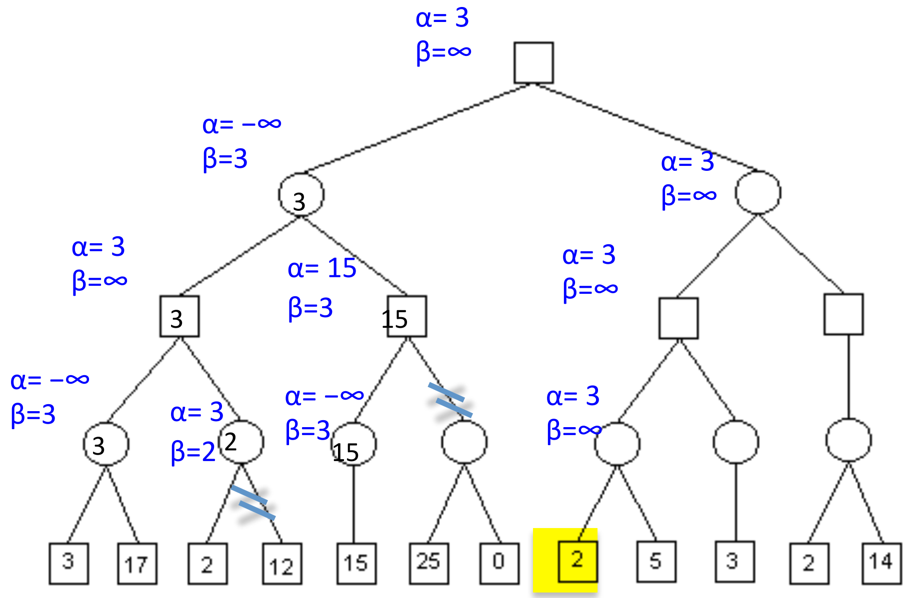

α= −∞
β=3
α= 3
β=∞
α= 15
β=3
15
α= −∞
β=3
α= 3
β=2
α= −∞
β=3
3
2
15
3
17
2
12
15
25
0
2
5
3
2
14

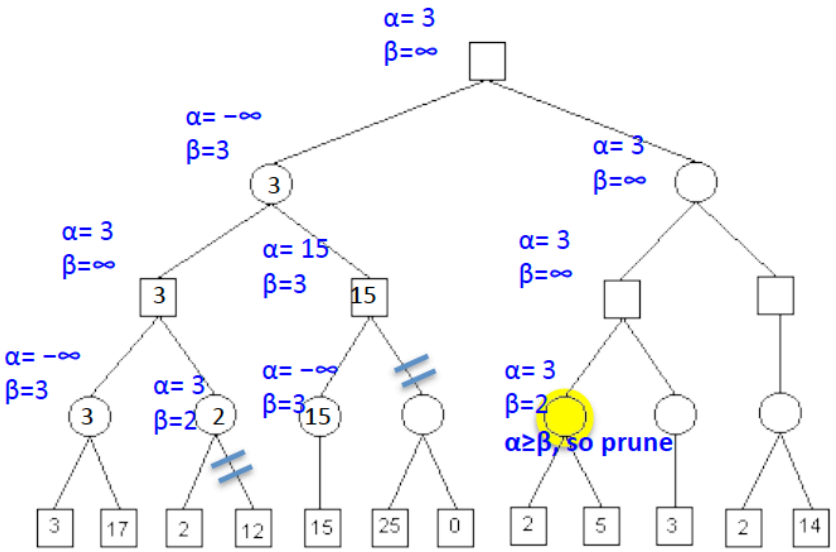

α= −∞
β=3
3
α= 3
β=∞
α= 15
β=3
15
α= −∞
β=3
α= 3
β=2
α= −∞
β=3
3
2
15
3
17
2
12
15
25
0
2
5
3
2
14

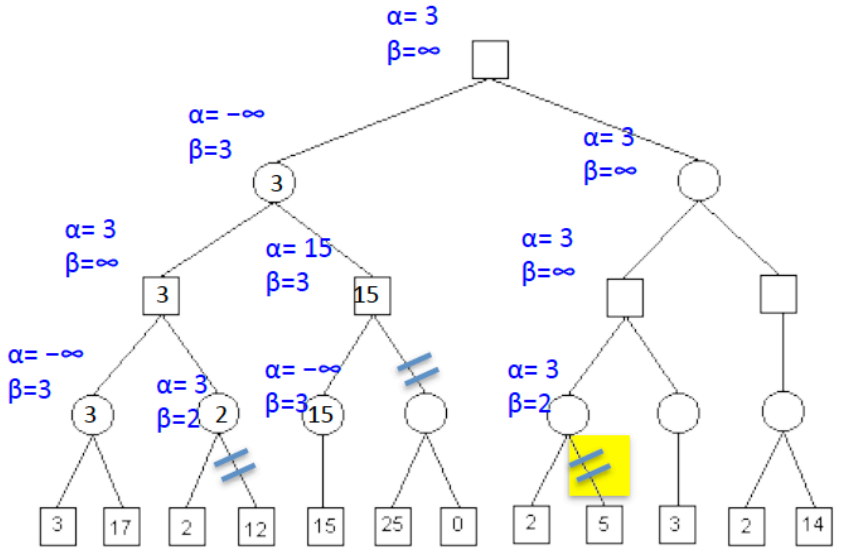

α= 3
β=∞
α= −∞
β=3
α= 3
β=∞
α<β, socontinue
α= 3
β=∞
α= 15
β=3
α= −∞
β=3
α= 3
β=2
α= −∞
β=3
3
3
15
3
2
15
3
17
2
12
15
25
0
2
5
3
2
14

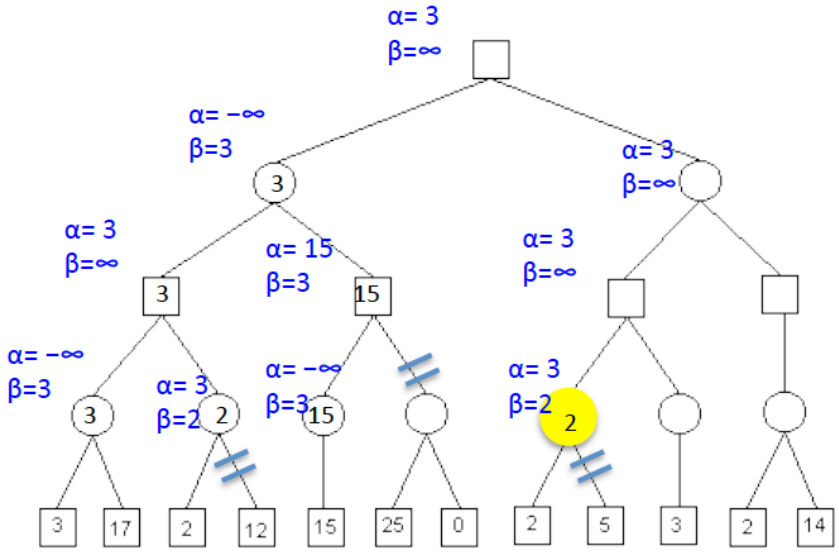

α= 3
β=∞
α= −∞
β=3
α= 3
β=∞
α= 3
β=∞
α= 15
β=3
α= 3
β=∞
α<β, socontinue
α= −∞
β=3
α= 3
β=2
α= −∞
β=3
3
3
15
3
2
15
3
17
2
12
15
25
0
2
5
3
2
14

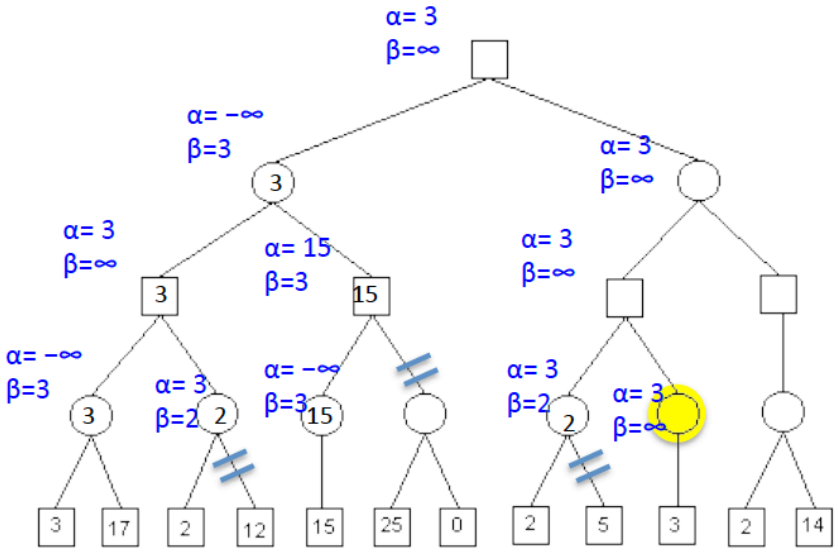

α= 3
β=∞
α= −∞
β=3
α= 3
β=∞
α= 3
β=∞
α= 15
β=3
α= 3
β=∞
α= −∞
β=3
α= 3
β=2
α= −∞
β=3
α= 3
β=∞
α<β, socontinue
3
3
15
3
2
15
3
17
2
12
15
25
0
2
5
3
2
14

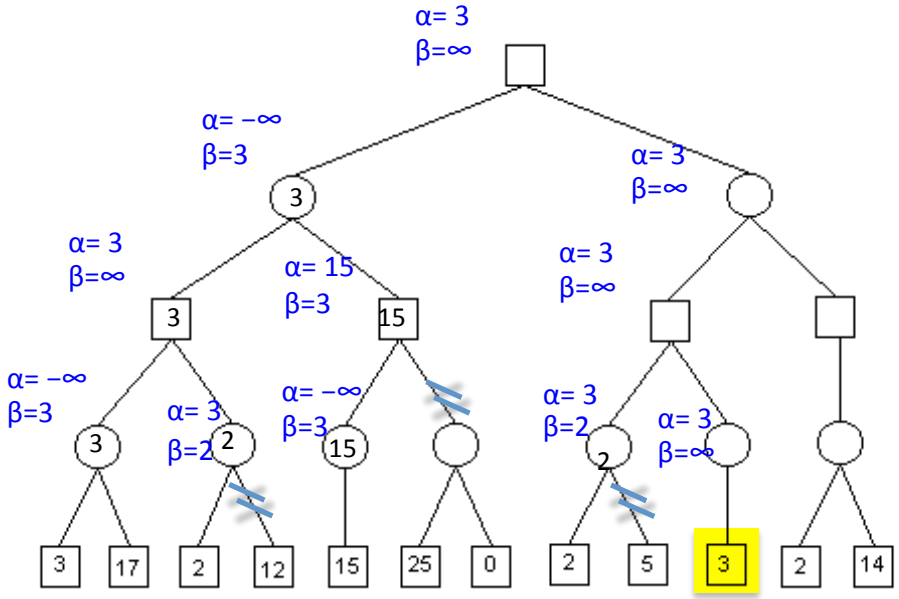

α= 3
β=∞
α= −∞
β=3
α= 3
β=∞
α= 3
β=∞
α= 15
β=3
α= 3
β=∞
α= −∞
β=3
α= 3
β=2
α= −∞
β=3
α= 3
β=∞
3
3
15
3
2
15
3
17
2
12
15
25
0
2
5
3
2
14

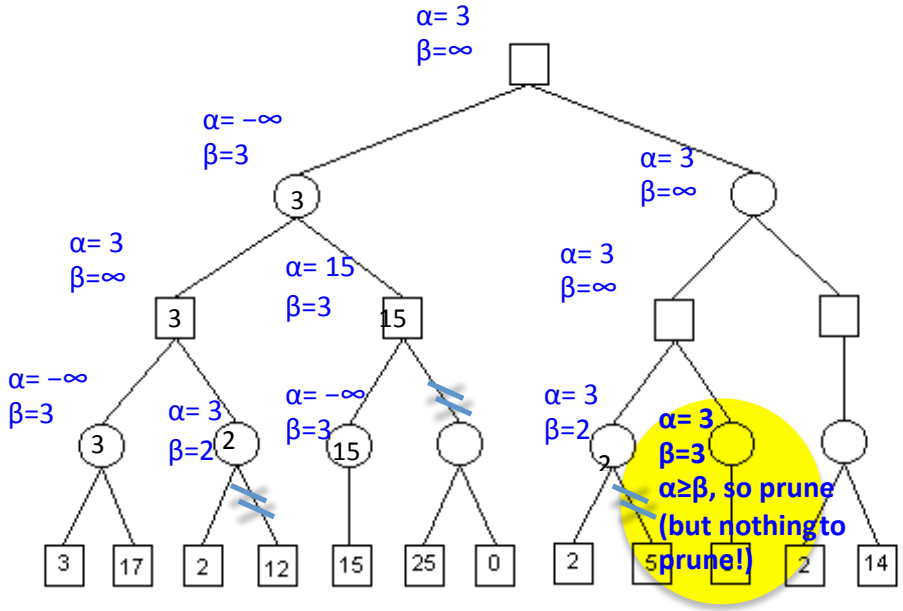

α= 3
β=∞
α= −∞
β=3
α= 3
β=∞
α= 3
β=∞
α= 15
β=3
α= 3
β=∞
α= −∞
β=3
α= 3
β=2
α= −∞
β=3
α= 3
β=2
α≥β, so prune
3
17
2
12
15
25
0
2
5
3
2
14

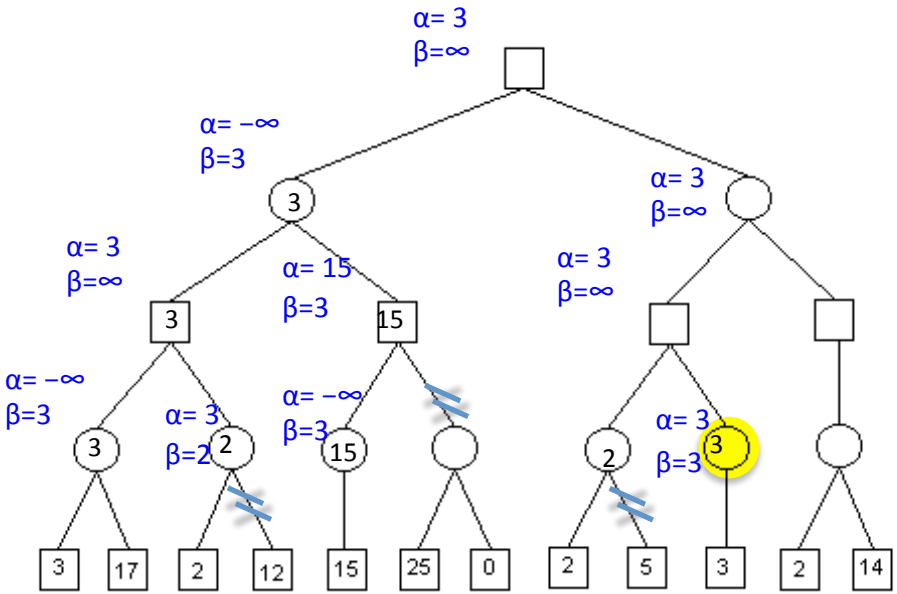

α= 3
β=∞
α= −∞
β=3
α= 3
β=∞
α= 3
β=∞
α= 15
β=3
α= 3
β=∞
α= −∞
β=3
α= 3
β=2
α= −∞
β=3
α= 3
β=2
3
17
2
12
15
25
0
2
5
3
2
14

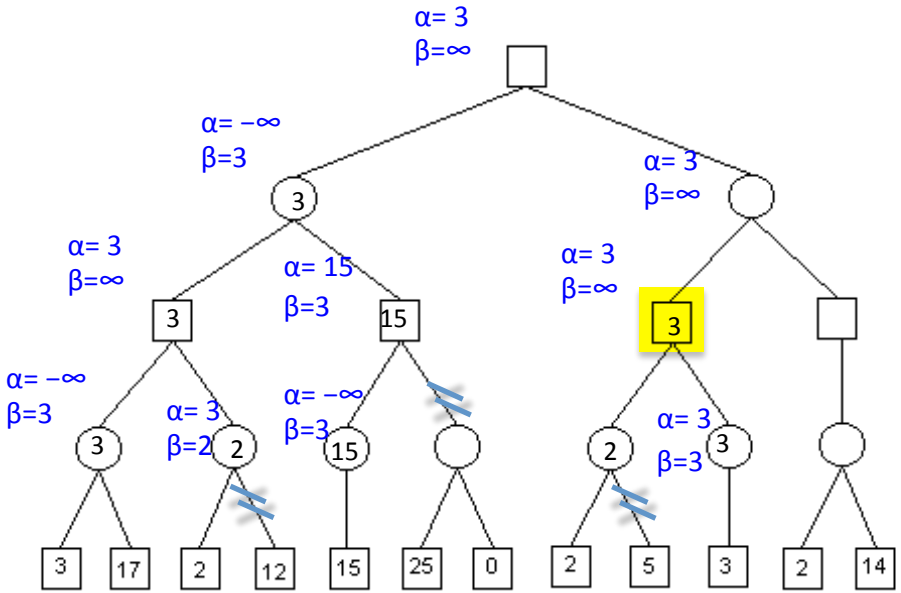

α= 3
β=∞
α= −∞
β=3
α= 3
β=∞
α= 3
β=∞
α= 15
β=3
α= 3
β=∞
α= −∞
β=3
α= 3
β=2
α= −∞
β=3
α= 3
β=2
2
3
17
2
12
15
25
0
2
5
3
2
14

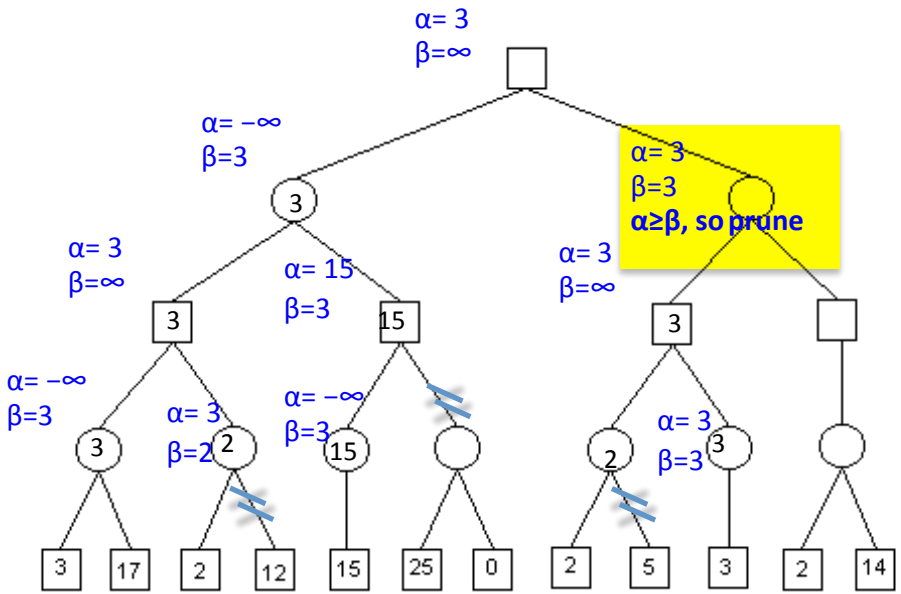

α= 3
β=∞
α= −∞
β=3
α= 3
β=∞
α= 3
β=∞
α= 15
β=3
α= 3
β=∞
α= −∞
β=3
α= 3
β=2
α= −∞
β=3
α= 3
β=2
α= 3
β=∞
3
17
2
12
15
25
0
2
5
3
2
14

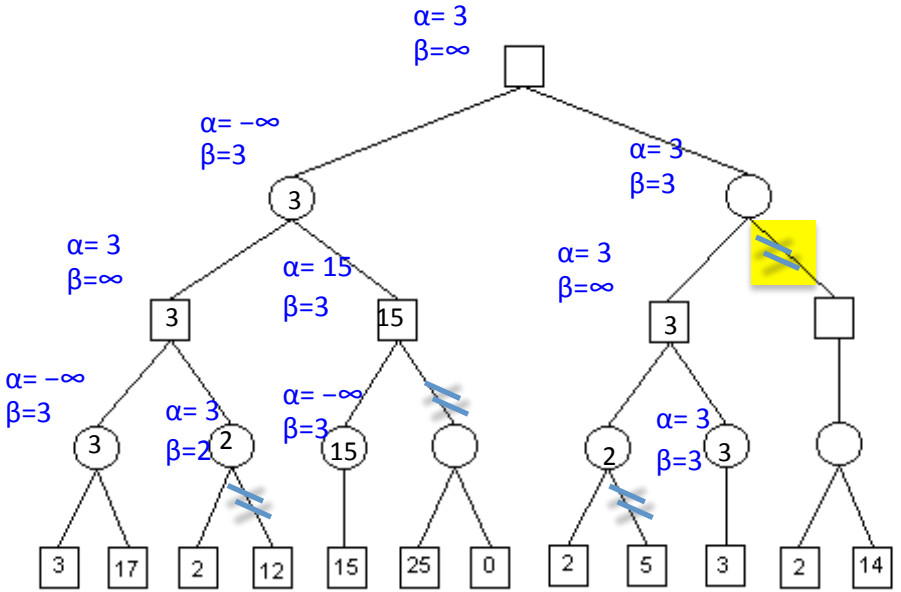

α= 3
β=∞
α= −∞
β=3
α= 3
β=∞
α= 3
β=∞
α= 15
β=3
α= 3
β=∞
α= −∞
β=3
α= 3
β=2
α= −∞
β=3
α= 3
β=2
α= 3
β=∞
3 17 2 12 15 25 0 2 5 3 2 14

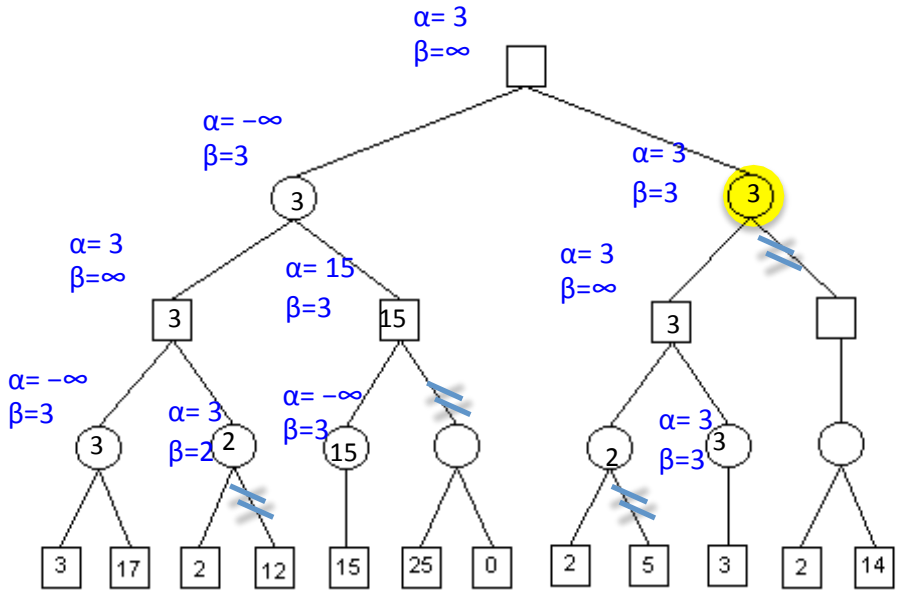

α= 3
β=∞
α= −∞
β=3
α= 3
β=∞
α= 3
β=∞
α= 15
β=3
α= 3
β=∞
α= −∞
β=3
α= 3
β=2
α= −∞
β=3
α= 3
β=2
α= 3
β=3
α≥β, so prune
(but nothing to
prune!)
3 17 2 12 15 25 0 2 5 2 14

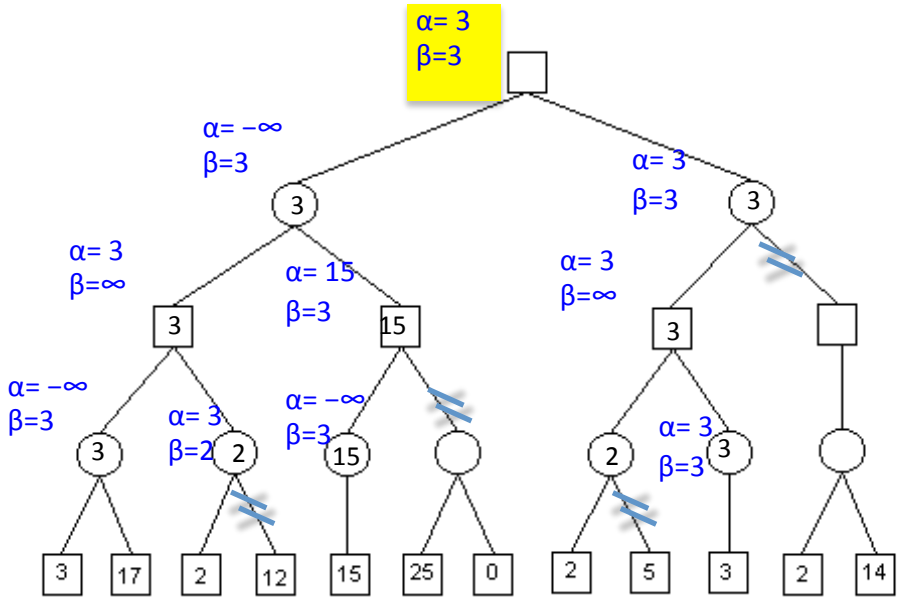

α= 3
β=∞
α= −∞
β=3
α= 3
β=∞
α= 3
β=∞
α= 15
β=3
α= 3
β=∞
α= −∞
β=3
α= 3
β=2
α= −∞
β=3
α= 3
β=3
3 17 2 12 15 25 0 2 5 3 2 14


α= 3
β=∞
α= −∞
β=3
α= 3
β=∞
α= 3
β=∞
α= 15
β=3
α= 3
β=∞
α= −∞
β=3
α= 3
β=2
α= −∞
β=3
α= 3
β=3
3 17 2 12 15 25 0 2 5 3 2 14


α= 3
β=∞
α= −∞
β=3
α= 3
β=∞
α= 15
β=3
α= 3
β=∞
α= 3
β=3
α≥β, so prune
α= −∞
β=3
α= 3
β=2
α= −∞
β=3
α= 3
β=3
3
17
2
12
15
25
0
2
5
3
2
14


α= 3
β=∞
α= −∞
β=3
α= 3
β=∞
α= 15
β=3
α= 3
β=∞
α= −∞
β=3
α= 3
β=2
α= −∞
β=3
α= 3
β=3
3
17
2
12
15
25
0
2
5
3
2
14


α= 3
β=∞
α= −∞
β=3
α= 3
β=3
α= 3
β=∞
α= 15
β=3
α= 3
β=∞
α= −∞
β=3
α= 3
β=2
α= −∞
β=3
α= 3
β=3
3
17
2
12
15
25
0
2
5
3
2
14


α= 3
β=3
α= −∞
β=3
α= 3
β=3
α= 3
β=∞
α= 15
β=3
α= 3
β=∞
α= −∞
β=3
α= 3
β=2
α= −∞
β=3
α= 3
β=3
3
17
2
12
15
25
0
2
5
3
2
14

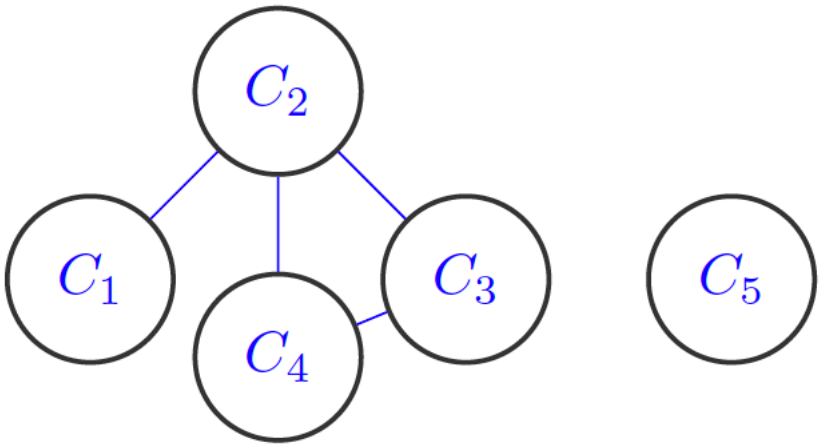
Exercice 5 :

Vous êtes en charge de l'emploi du temps pour des cours en informatique les Lundi, Mercredi et Vendredi. Il y a 5 classes et 5 professeurs pour enseigner ces classes pendant ces 3 jours.
Chaque professeur ne peut enseigner qu'une classe à la fois. les cours sont :

1.Cours 1 - Intro à la Programmation: de 8:00-9:00am
2. Cours 2 - Intro à l'Intelligence Artificielle : de 8:30-9:30am
3. Cours 3 - Natural Language Processing: de 9:00-10:00am
4. Cours 4 - vision par ordinateur 9:00-10:00am
5. Cours 5 - Machine Learning: de 10:30-11:30am
Les professeurs sont:
1. Professeur A, qualifié pour enseigner cours 1, 2, and 5.
2. Professeur B, qualifié pour enseigner cours 3, 4, and 5.
3. Professeur C, qualifié pour enseigner cours 1, 3, and 4.

1. Formuler ce problème comme un CSP dans lequel il y a une variable par classe, et identifier leurs domaines (après avoir fixé les contraintes unaires), et les contraintes binaires. Les contraintes doivent être spécifié formellement et de manière précise.

Variables	Domains (or unary constraints)	Binary Constraints
C_1	{A, C}	$C_1 \neq C_2$
C_2	{A}	$C_2 \neq C_3$
C_3	{B, C}	$C_2 \neq C_4$
C_4	{B, C}	$C_3 \neq C_4$
C_5	{A, B}	



Exercice 6 :

Une compagnie fabriquant des voitures a trois lignes de production : la ligne A, la ligne B et la ligne C. La compagnie fabrique trois sortes de voitures : une voiture sportive, une voiture familiale et une voiture tout-terrain. Chaque ligne de production fonctionne durant 4 périodes de 2 heures dans une journée : de 8h à 10, de 10 à 12h, de 12h à 14h et de 14h à 16h. La construction d'une voiture dans une ligne nécessite une de ces périodes de 2 heures complète. De plus :

– La ligne A peut fabriquer la voiture sportive et la voiture familiale seulement.
– La ligne B peut fabriquer la voiture tout-terrain seulement.
– La ligne C peut fabriquer la voiture familiale et la voiture tout-terrain seulement.

Supposons que la compagnie reçoive une commande d'un client pour 2 voitures sportives, 3 voitures familiales et 5 voitures tout-terrain. De plus, on exige que toutes les voitures familiales et toutes les voitures tout-terrain aient été assemblées avant 14h.

Formulez ce problème comme un problème de satisfaction de contraintes. Identifiez (1) toutes les variables du problème, (2) le domaine de chacune des variables et (3) l'ensemble des contraintes à satisfaire.

solution:
Variables : F1, F2, F3, S1 , S2, T1, T2, T3, T4, T5 (les types de voitures)

Domaines : toutes les variables ont comme domaine {A1, A2, A3, A4, B1, B2, B3, B4, C1, C2, C3, C4} (les 4 périodes de 2 heures de production, pour chaque ligne)

Contraintes :

$Fi \neq Sk$, $Fi \neq Tk$ et $Si \neq Tk$ pour tout i et k (deux voitures ne peuvent pas être construites
$Fi \neq Fk$, $Si \neq Sk$ et $Ti \neq Tk$ pour tout i et $k \neq i$ sur la même ligne en même temps)

$Fi \neq A4$, $Fi \neq B4$, $Fi \neq C4$ pour tout i (les voitures familiales et tout-terrain doivent
$Ti \neq A4$, $Ti \neq B4$, $Ti \neq C4$ pour tout i doivent être construites avant 14h)

$Fi \neq Bk$, $Si \neq Bk$, $Si \neq Ck$ et $Ti \neq Ak$ pour tout i et k (contraintes tenant compte que certaines lignes ne produisent pas certaines voitures)

Note : il y a plusieurs autres façons de décrire ce problème. On aurait pu définir les domaines de variables de façon à s'assurer que les contraintes de production avant 14h et sur les types de voiture par ligne soient satisfaites. On aurait alors eu moins de contraintes à spécifier explicitement.

On aurait aussi pu prendre une approche totalement différente et utiliser les périodes comme variables et les types de voiture comme valeurs, comme on l'a fait en classe (ceci donnerait lieu à des contraintes différentes…)

Exercice 7 :

Soit le problème CSP suivant :

(a) Dom[X] = {1, 2, 3, 4}
(b) Dom[Y] = {1, 2, 3, 4}
(c) Dom[Z] = {1, 2, 3, 4}
(d) Dom[W] = {1, 2, 3, 4, 5}

et 3 contraintes:

(a) C1(X, Y,Z) est satisfaite si $X = Y + Z$
(b) C2(X,W) est satisfaite si $W > X$
(c) C3(X, Y,Z,W) est satisfaite si n $W = X + Z + Y$

utiliser l'algorithme AC-3 avec ces contraintes et donner les valeurs des variables consistantes.

(a) $Dom[X] = \{1, 2, 3, 4\}$
(b) $Dom[Y] = \{1, 2, 3, 4\}$
(c) $Dom[Z] = \{1, 2, 3, 4\}$
(d) $Dom[W] = \{1, 2, 3, 4, 5\}$

And 3 constraints:

(a) $C_1(X, Y, Z)$ which is satisfied only when $X = Y + Z$
(b) $C_2(X, W)$ which is satisfied only when $W > X$
(c) $C_3(X, Y, Z, W)$ which is satisfied only when $W = X + Z + Y$

Enforce GAC on these constraints, and give the resultant GAC consistent variable domains.

- All constraints put on GAC queue.
- Process C_3 first.
  X = 1 (X=1, Y=1, Z=1, W=3)
  X = 2 (X=2, Y=1, Z=1, W=4)
  X = 3 (X=3, Y=1, Z=1, W=5)
  X = 4 – Inconsistent.
  Dom(X) = {1, 2, 3}
  similarly
  Dom(Y) = {1, 2, 3}
  Dom(Z) = {1, 2, 3}

W = 1 – inconsistent
W = 2 – inconsistent
W = 3 – same support as X=1
W = 4 – same support as X = 2
W= 5 – same support as X = 3

Dom(W) = {3, 4, 5}

All domains pruned, but all other constraints already on GAC queue

(a) $Dom[X] = \{1, 2, 3, 4\}$
(b) $Dom[Y] = \{1, 2, 3, 4\}$
(c) $Dom[Z] = \{1, 2, 3, 4\}$
(d) $Dom[W] = \{1, 2, 3, 4, 5\}$

And 3 constraints:

(a) $C_1(X, Y, Z)$ which is satisfied only when $X = Y + Z$
(b) $C_2(X, W)$ which is satisfied only when $W > X$
(c) $C_3(X, Y, Z, W)$ which is satisfied only when $W = X + Z + Y$

Enforce GAC on these constraints, and give the resultant GAC consistent variable domains.

- Process C_2 next current domains:
  Dom(X) = {2, 3}
  Dom(W) = {4,5}

X = 2 – {X=2, W=4}
X = 3 – {X=3, W=4}

W = 4 – found support
W = 5 – {X=3, W=5}

No Domains pruned.
Nothing added to queue

Queue Empty

GAC finished.

GAC domains:
X = {2,3}
Z = {1, 2}
Y = {1, 2}
W = {4.5}

(a) $Dom[X] = \{1, 2, 3, 4\}$
(b) $Dom[Y] = \{1, 2, 3, 4\}$
(c) $Dom[Z] = \{1, 2, 3, 4\}$
(d) $Dom[W] = \{1, 2, 3, 4, 5\}$

And 3 constraints:

(a) $C_1(X, Y, Z)$ which is satisfied only when $X = Y + Z$
(b) $C_2(X, W)$ which is satisfied only when $W > X$
(c) $C_3(X, Y, Z, W)$ which is satisfied only when $W = X + Z + Y$

Enforce GAC on these constraints, and give the resultant GAC consistent variable domains.

- Note GAC enforce does not find a solution
  To find a solution we must use do search while enforcing GAC.

- Branch on X.
  X = 2
  GAC(C_1) ➔ Y = 1, Z=1
  GAC(C_2) ➔ no changes
  GAC(C_3) ➔ W = 4
  This is a solution.

- Branch on X = 3
  GAC(C_1) ➔ no changes
  GAC(C_2) ➔ no changes
  GAC(C_3) ➔ Prune W=4
  Prune Y = 2
  Prune Z = 2
  Current Domains
  X={3}, Y={1}, Z={1}, W={5}
  GAC(C_1) ➔ Prune Y={1} DWO

NOTE No solution with X=3 but
X=3 not pruned by GAC enforce.

(a) $Dom[X] = \{1, 2, 3, 4\}$
(b) $Dom[Y] = \{1, 2, 3, 4\}$
(c) $Dom[Z] = \{1, 2, 3, 4\}$
(d) $Dom[W] = \{1, 2, 3, 4, 5\}$

And 3 constraints:

(a) $C_1(X, Y, Z)$ which is satisfied only when $X = Y + Z$
(b) $C_2(X, W)$ which is satisfied only when $W > X$
(c) $C_3(X, Y, Z, W)$ which is satisfied only when $W = X + Z + Y$

Enforce GAC on these constraints, and give the resultant GAC consistent variable domains.

- Process C_2 next
  Currently
  Dom(X) = {1, 2, 3}
  Dom(W) = {3, 4, 5}

  X = 1 (X=1, W=3)
  X = 2 (X=2, W=3)
  X = 3 (X=3, W=4)

  W=3, W=4 found supports already

W=5 (X=1, W=5)

No domains pruned. Nothing added to GAC Queue

(a) $Dom[X] = \{1, 2, 3, 4\}$
(b) $Dom[Y] = \{1, 2, 3, 4\}$
(c) $Dom[Z] = \{1, 2, 3, 4\}$
(d) $Dom[W] = \{1, 2, 3, 4, 5\}$

And 3 constraints:

(a) $C_1(X, Y, Z)$ which is satisfied only when $X = Y + Z$
(b) $C_2(X, W)$ which is satisfied only when $W > X$
(c) $C_3(X, Y, Z, W)$ which is satisfied only when $W = X + Z + Y$

Enforce GAC on these constraints, and give the resultant GAC consistent variable domains.

- Process C_1 next

At this stage
Dom(X) = Dom(Y) = Dom(Z)
= {1, 2, 3}

X = 1 – inconsistent
X = 2 – (X=2, Y=1, Z=1)
X = 3 – (X=3, Y=1, Z=2)

Y = 1 – same support as X=2
Y = 2 – (X=3, Y=2, Z=1)
Y = 3 – inconsistent

Z = 1 – same support as X=2
Z = 2 – same support as X=3
Z = 3 – inconsistent

Updated domains
X = {2,3}
Y = {1,2}
Z = {1,2}

Put C_2 and C_3 back onto GAC queue

(a) $Dom[X] = \{1, 2, 3, 4\}$
(b) $Dom[Y] = \{1, 2, 3, 4\}$
(c) $Dom[Z] = \{1, 2, 3, 4\}$
(d) $Dom[W] = \{1, 2, 3, 4, 5\}$

And 3 constraints:

(a) $C_1(X, Y, Z)$ which is satisfied only when $X = Y + Z$
(b) $C_2(X, W)$ which is satisfied only when $W > X$
(c) $C_3(X, Y, Z, W)$ which is satisfied only when $W = X + Z + Y$

Enforce GAC on these constraints, and give the resultant GAC consistent variable domains.

- Process C_3 next current domains:
  Dom(X) = {2, 3}
  Dom(Y) = {1, 2}
  Dom(Z) = {1, 2}
  Dom(W) = {3,4,5}

X = 2 – {X=2, W=4, Y=1, Z=1}
X = 3 – {X=3, W=5, Y=1, Z=1}

Y = 1 – found support
Y = 2 – {X=2, W=5, Y=2, Z=1}

Z = 1 – found support
Z = 2 – {X=2, W=5, Y=1, Z=2}

W = 3 inconsistent
W = 4 – found support
W = 5 – found support

Pruned domains
W = {4, 5}

C_2 already on GAC queue