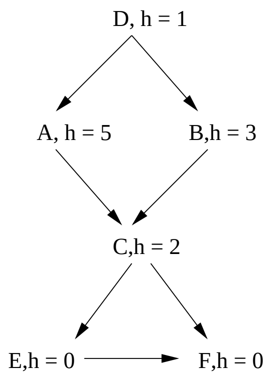
# Université Paris 7 - Master 1 Informatique - Intelligence Artificielle

Examen du 18 janvier 2005 - Durée : 2 heures

**Informations :** Tous les documents reliés sont autorisés. Le barème est donné à titre indicatif et peut être modifié.

**Exercice 1** A* (**4 points**)
Considérez l'espace de recherche suivant (D est le début et F l'état qu'on veut atteindre):

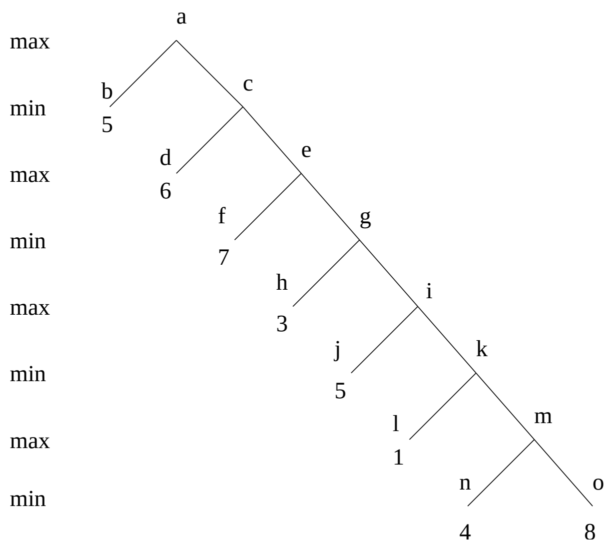



Pour chaque nœud est indiquée la valeur de l'heuristique $h$. On veut récupérer le coût de chaque arc entre deux nœuds. Pour cela nous disposons d'une trace de l'algorithme A*. Pour chaque pas de l'algorithme est indiquée la liste des nœuds encore à traiter avec leur valeur $f = g + h$. Si un nœud peut apparaître deux fois avec deux valeurs de f différentes, on conserve seulement celui avec la meilleure (la plus petite) valeur de $f$.

```
[(D, f = 1)]
[(B, f = 7), (A, f = 8)]
[(A, f = 8), (C, f = 10)]
[(C, f = 10)]
[(E, f = 12), (F, f = 15)]
[(F, f = 14)]
```

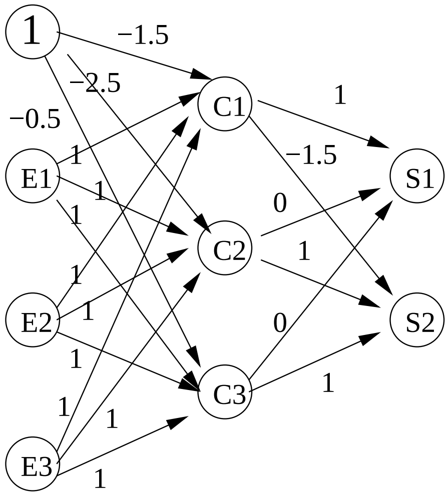
Utilisez cette trace et votre connaissance du fonctionnement de A* pour calculer les coûts de tous les arcs. Détaillez. Est-ce que $h$ est admissible ?

**Exercice 2** Jeux (4 points)
Considérez l'arbre de jeux suivant. La racine est un nœud max.




- Appliquez l'algorithme minimax sur cet arbre.
- Appliquez l'algorithme SSS* sur cet arbre.
- Est-ce qu'on peut donner un arbre de jeux (pas forcément complet) avec 1000 feuilles, pour lequel l'algorithme $\alpha$-$\beta$ ne considère que 2 feuilles ? Donnez schématiquement un tel arbre ou montrez que ce n'est pas possible.

**Exercice 3** Réseaux de neurones (**3 points** )
Considérez le réseau de neurones suivant:




Ce réseau contient des perceptrons linéaires avec seuil 0. La fonction d'activation est la fonction de Heaviside (Si $\vec{w}\vec{x} > 0$ alors 1 sinon 0). Les poids sont indiqués sur les arcs. On suppose que les entrées $E1, E2, E3$ sont 0 ou 1. On remarque que les poids des entrées (sauf pour l'entrée fixe 1) sont tous 1. Ce qui signifie que seulement la connaissance du **nombre** d'entrées qui sont à 1 suffit pour décrire les sorties des neurones cachés $C1, C2, C3$.

- Décrivez ce que calculent les neurones $C1, C2, C3$ en fonction du **nombre** d'entrées à 1.
- De la même façon décrire ce que calculent les neurones de sortie $S1, S2$

**Exercice 4** Réseaux de neurones (**5 points**)
On considère un "perceptron" avec entrées $x_1, x_2, \ldots, x_n$ et poids $w_1, w_2, \ldots, w_n$. La sortie $o$ est donnée par

$$o = w_1(x_1 + x_1^3) + w_2(x_2 + x_2^3) \cdots + w_n(x_n + x_n^3)$$

- Donnez un algorithme d'apprentissage basé sur la descente du gradient pour un tel "perceptron". Indication : Définissez d'abord une fonction d'erreur standard. Expliquez votre façon d'obtenir l'algorithme.

On considère un tel "perceptron" avec 2 entrées $x_1$ et $x_2$, les poids $w_1 = 1$ et $w_2 = -1$. L'échantillon d'apprentissage consiste en un vecteur $x_1 = 1, x_2 = 1$ et la sortie désirée est 1.

- Détaillez le premier pas de votre algorithme.

**Exercice 5** Apprentissage (**4 points**)

- Construisez au fur et à mesure l'automate obtenu par l'algorithme (ZR) d'apprentissage pour les langages réversibles sur l'échantillon $\{aba, bc, abc, aaba, babba\}$.
- Soit $P$ un échantillon positif à partir duquel on a construit avec l'algorithme ZR un automate $A$. Un nouveau mot $w$ est ajouté à l'échantillon. On ne veut pas refaire toute l'algorithme.
  - Comment pourrait-on "ajouter" le mot $w$ en utilisant $A$ ? Donnez un algorithme qui étant donné un automate déterministe réversible $A$ et un mot $w$ construit un automate déterministe réversible $A'$ qui accepte $w$ et qui satisfait la propriété $L(A) \subseteq L(A')$.
  - Construisez d'abord l'automate pour l'échantillon $\{aba\}$. Ajoutez en suite **un par un** les mots $bc$, $abc$, $aaba$, $babba$ en utilisant votre algorithme (qui est donc utilisé 4 fois).
  - Est-ce que le résultat est le même que le résultat de ZR sur l'échantillon $\{aba, bc, abc, aaba, babba\}$ ?