Heuristiques

D.E ZEGOUR
École Supérieure d'Informatique
ESI

Heuristiques

Sommaire

A. Introduction

B. Recherche d ’une branche

. Hill Climbing

. Beam Search

. Best First Search

C. Recherche d'une branche optimale

. Branch And Bound

. Branch and Bound avec sous estimations

. Branch and Bound avec programmation dynamique

. A*

Heuristiques

Introduction

L'exploration systématique consiste à parcourir un graphe pour la recherche des solutions à des problèmes.

Le graphe peut être très grand et donc l'exploration ne peut se faire.

Une heuristique est une technique qui améliore l'efficacité d'un processus de recherche avec sacrifice des meilleures solutions

Pour des problèmes d'optimisation où la recherche d'une solution exacte(optimale) est difficile(coût exponentiel), on peut se contenter d'une solution satisfaisante donnée par une heuristique avec un coût plus faible.

Heuristiques

Introduction

Certaines heuristiques sont polyvalentes (elles donnent d'assez bons résultats pour une large gamme de problèmes)

D'autres sont spécifiques à chaque type de problème.

Dans le backtracking, dans les jeux de stratégie (jeu d'échec), on a déjà utilisé les heuristiques.

Les heuristiques peuvent donner des solutions optimales, ce qui semble paradoxal (Algorithme A*).

Heuristiques

Algorithmes de recherche utilisant les heuristiques

1. Procédures simples utilisées pour trouver des branches (pas forcément les plus courtes)

Hill Climbing
Beam Search
Best First search

Elle dérivent de Depht First Search et Breadth First Search

2. Procédures plus complexes qui recherchent les plus courtes branches.

Donnent une importance primaire au coût de la branche traversée.

Branch And Bound
Branch and Bound avec sous estimations
Branch and Bound avec programmation dynamique
*A**

Recherches guidées : Au niveau de chaque nœud, on a une information pour la recherche (coût, distance, ...)

Heuristiques

Exemple considéré

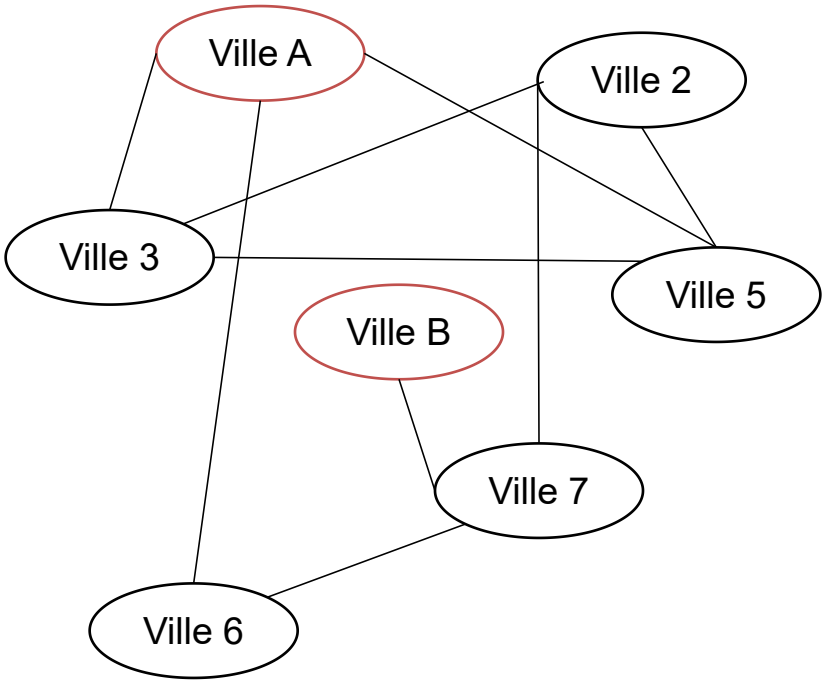
Nous sommes dans une ville A et nous voulons rejoindre une ville B.

Pour chaque ville traversée (y compris A), il y a un sous ensemble de villes pour lesquelles il y a une route.

Nous considérons les distances kilométriques entre les villes traversées

Estimation de la distance restante : la distance directe (vol d'oiseau)

Supposition (dans les exemples) : au delà de 110Km, il n ya plus de chemin entre A et B



Heuristiques

Escalade d'une montagne (alpiniste)
- en fonction des distances des trous
- Si Echec, retourne en arrière pour explorer un autre chemin

Hill Climbing

C'est une recherche en profondeur.

A chaque étape, on sélectionne le nœud avec distance (ou coût) minimale.

C'est une amélioration de DFS(Depht First Search)

Utilise une pile (Explore tous les noeuds noeuds du niveau i avant de revenir aux noeuds du niveau i-1)

1. Empiler le nœud racine

2. Si la pile est vide, recherche sans succès

3. Dépiler un élément e, s'il est égal à l'élément recherché, l'algorithme se termine avec succès, autrement

4. *Trier les fils de e non visités, s'ils existent, par l'estimation de la distance restante*, ensuite les empiler.

5. Allera 2

Heuristiques

Les numéros en rouge désignent l'ordre des villes visitées.
Les valeurs sous les nœuds désignent les estimations des distances restantes.

Hill Climbing

Pile : A

Pile : Q (70) R(80) N(90)

Pile : M(120) S(170) R(80) N(90)

Pile : S(170) R(80) N(90)

Pile : R(80) N(90)

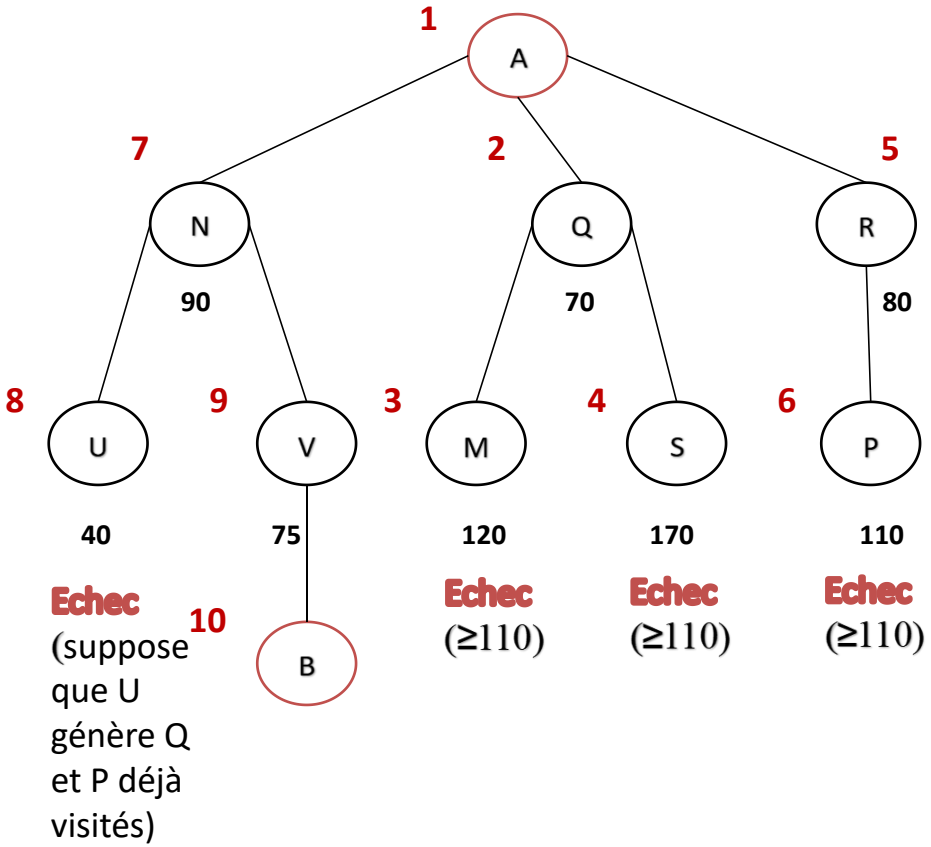
Pile : P(110) N(90)

Pile : N(90)

Pile : U(40) V(75)

Pile : V(75)

Pile : B



Heuristiques

Beam search(w)

Idem que *Hill Climbing* sauf que pour chaque niveau seulement les w "bons" premiers nœuds sont explorés.

Beam : Rayon (ou faisceau lumineux)

Si w = 1, Steepest Hill climbing (Hill Climbing raide ou rigide)

Remarque

Si w=1 et et choix du noeud aléatoire : Random Hill Climbing (Pas d'estimation)

Heuristiques

Best First Search

Utilise une liste ordonnée (ou une file d'attente avec priorité)

1. Mettre le noeud racine dans une liste.

2. Si la liste est vide, recherche sans succès.

3. Si Premier(L) est égal à l'élément recherché , l'algorithme se termine avec succès. Autrement

4. Supprimer le premier élément.
Pour chaque fils non visité de cet élément :
- calculer l'*estimation de la distance restante*
- placer le dans la liste

5. Allera 2

Heuristiques

Les numéros en rouge désignent l'ordre des villes visitées.
Les valeurs sous les nœuds désignent les estimations des distances restantes.

Best First Search

Liste : A

Liste : Q(70), R(80), N(90)

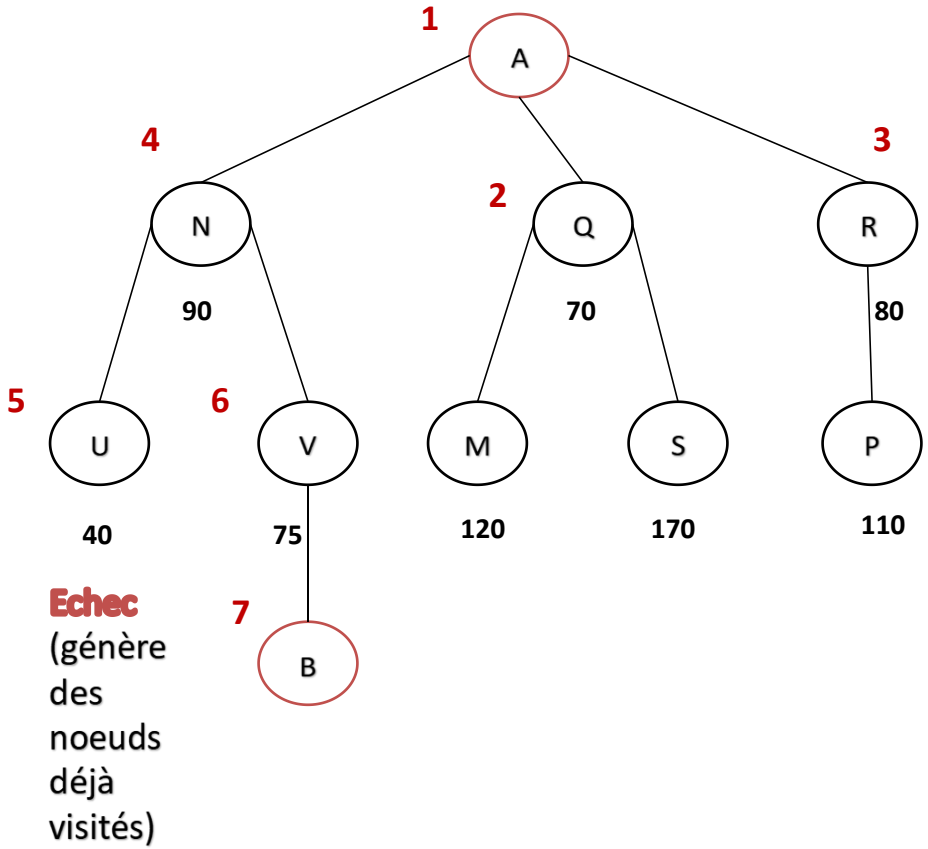
Liste : R(80), N(90), M(120), S(170)

Liste : N(90), P(110), M(120), S(170)

Liste :U(40), V(75), P(110), M(120), S(170)

Liste :V(75), P(110), M(120), S(170)

Liste : B(0), P(110), M(120), S(170)



Best First Search trouve toujours un bon chemin vers le but.(Avec la supposition qu'on fait une bonne estimation de la distance restante).

Heuristiques

Recherche de la meilleure branche

Utilisées quand on donne une importance primaire au coût de la branche.

Pour les méthodes précédentes, on ne tient pas compte du passif. Ici, il est tenu compte du passif. On attribue donc un coût à toute la branche parcourue depuis la racine.

- Accepter les éléments déjà visités
- Ne pas accepter les nouveaux élements formant des cycles

Heuristiques

Branch and Bound

Tenir compte du passif (attribuer une valeur à chaque branche)

Utilise une liste ordonnée(ou file d'attente avec priorité)

1. Placer le nœud racine de longueur 0 dans la liste.

2. Répéter jusqu'à ce que liste vide ou nœud recherché trouvé :
a) Si la première branche contient le nœud recherché, fin avec succès.
c) Sinon
- Supprimer la branche de la liste et former des branches nouvelles en étendant la branche supprimée d'une étape.
- Calculer les coût cumulés des branches et les ajouter dans la liste

3. Autrement " pas d'élément"

Heuristiques

Les numéros en rouge désignent l'ordre des villes visitées.
Les valeurs soulignées désignent les cumuls des distances parcourues.

Branch and Bound

Liste : A

Liste : AR(80), AN(90), AQ(170)

Liste : AN(90), ARP(110), AQ(170)

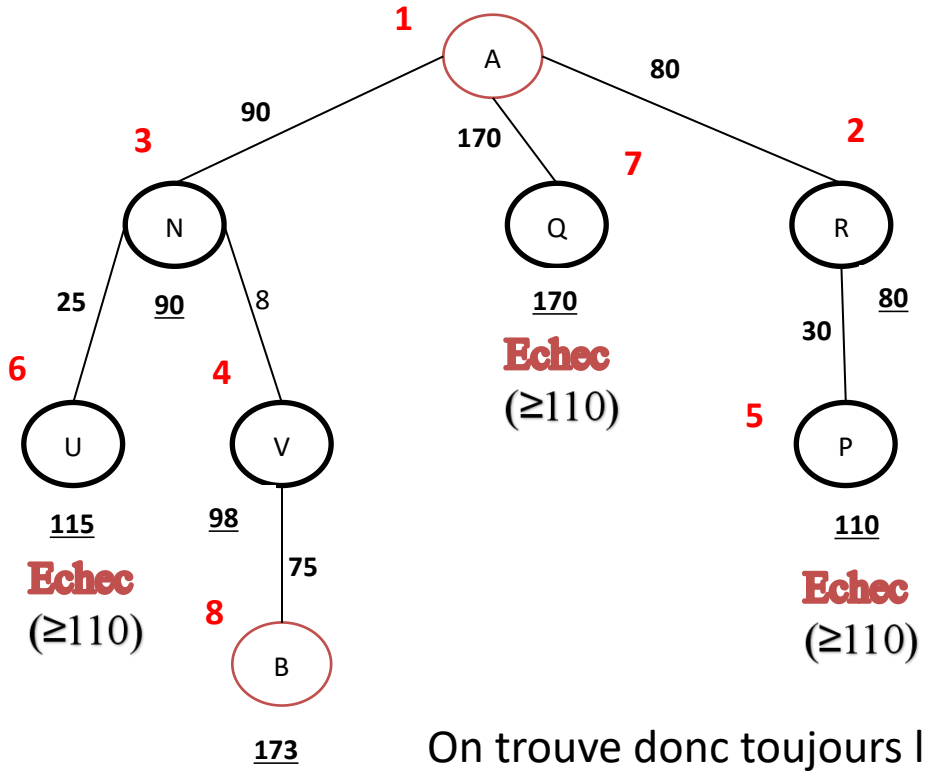
Liste : ANV(98), ARP(110), ANU(115), AQ(170)

Liste : ARP(110), ANU(115), AQ(170), ANVB(173)

Liste : ANU(115), AQ(170), ANVB(173)

Liste : AQ(170), ANVB(173)

Liste : ANVB(173)



- Pas d'estimation sur la distance restante
- L'heuristique : choix d'une branche arbitraire qui n'est pas la meilleure.

On trouve donc toujours la solution optimale.

Heuristiques

Branch and Bound avec sous estimation

Tenir compte du passif (attribuer une valeur à chaque branche)

Utilise une liste ordonnée (ou file d'attente avec priorité)

1. Placer le nœud racine de longueur 0 dans la liste.

2. Répéter jusqu'à ce que liste vide ou nœud recherché trouvé :
 a) Si la première branche contient le nœud recherché, fin avec succès.
 c) Sinon
 - Supprimer la branche de la liste et former des branches nouvelles en étendant la branche supprimée d'une étape.
 - Calculer la somme du coût cumulé et l'estimation de la distance restante des branches et les ajouter dans la liste

3. Autrement " fin avec échec"

Heuristiques

Les numéros en rouge désignent l'ordre des villes visitées.
Les valeurs soulignées désignent les cumuls des distances parcourues + estimation de la distance restante

Branch and Bound avec sous estimation

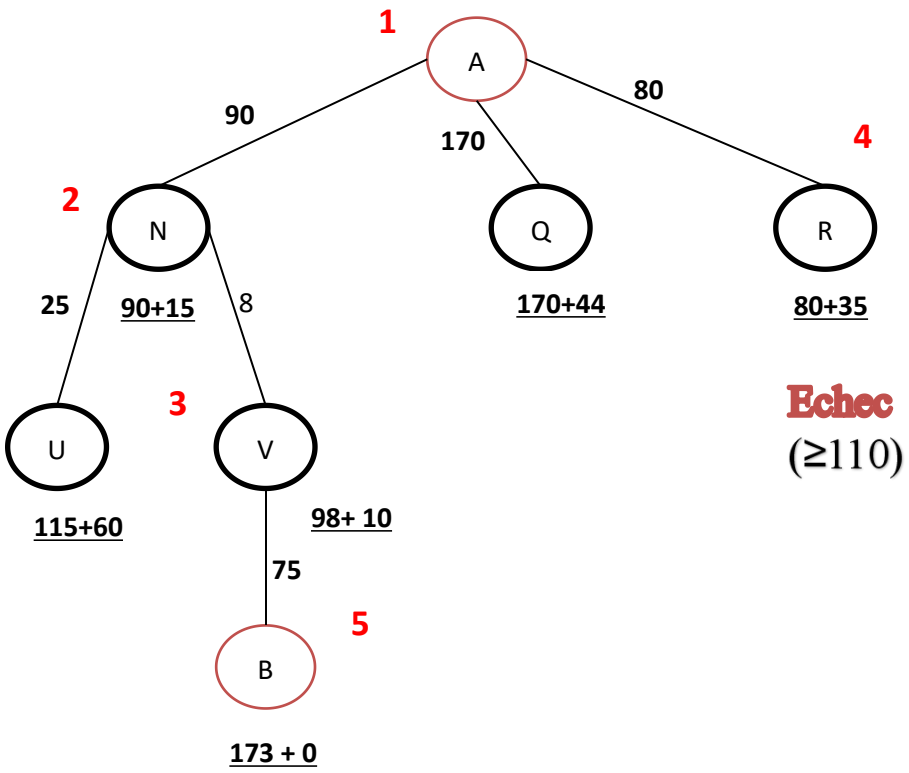
Liste : A

Liste : AN(105),AR(115),AQ(214)

Liste : ANV(108), AR(115), ANU(175), AQ(214)

Liste : AR(115), ANVB(173), ANU(175), AQ(214)

Liste : ANVB(173), ANU(175), AQ(214)



Heuristiques

Branch and Bound avec programmation dynamique

Tenir compte du passif
(attribuer une valeur à chaque branche)

Utilise une liste ordonnée
(ou file d'attente avec priorité)

1. Placer le nœud racine de longueur 0 dans la liste.

2. Répéter jusqu'à ce que liste vide ou nœud recherché trouvé :
a) Si la première branche contient le nœud recherché, fin avec succès.
c) Sinon
- Supprimer la branche de la liste et former des branches nouvelles en étendant la branche supprimée d'une étape.
- calculer les coûts cumulés des branches et les ajouter dans la liste
- Si deux ou plusieurs branches possèdent un nœud en commun, éliminer toutes ces branches sauf celle avec un coût minimal.

3. Autrement " pas d'élément"

Heuristiques

Les numéros en rouge désignent l'ordre des villes visitées.
Les valeurs soulignées désignent les cumuls des distances parcourues

Branch and Bound avec programmation dynamique

Liste : A

Liste : AR(80),AN(90), AQ(170)

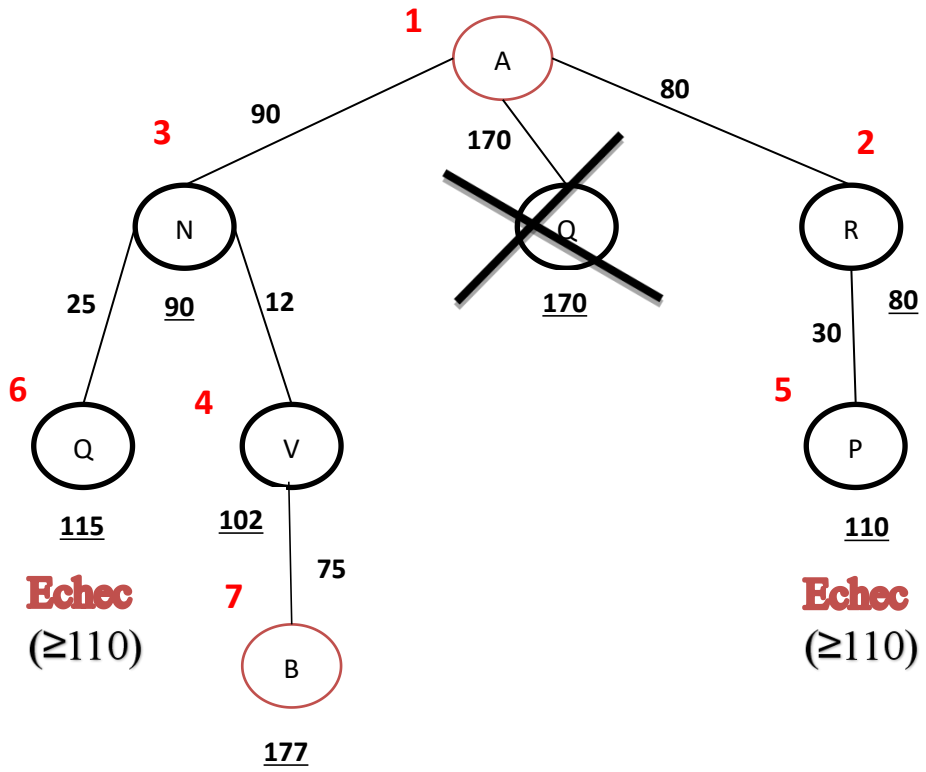
Liste : AN(90),ARP(110), AQ(170)

Liste : ANV(102),ARP(110), ANQ(115)

Liste : ARP(110), ANQ(115), ANVB(177)

Liste : ANQ(115), ANVB(177)

Liste : ANVB(177)



Heuristiques

Branch and Bound avec sous estimation et programmation dynamique = A*

C'est branch and bound amélioré.

Il combine l'estimation de la distance restante avec la programmation dynamique.

1. Placer le nœud racine de longueur 0 dans la liste.

2. Répéter jusqu'à ce que liste vide ou nœud recherché trouvé :
a) Si la première branche contient le nœud recherché, fin avec succès.
c) Sinon
- Supprimer la branche de la liste et former des branches nouvelles en étendant la branche supprimée d'une étape.
- calculer la somme du coût cumulé et l'estimation de la distance restante des branches et les ajouter dans la liste
- Si deux ou plusieurs branches possèdent un nœud en commun, éliminer toutes ces branches sauf celle avec un coût minimal.

3. Autrement " pas d'élément"

Heuristiques

Les numéros en rouge désignent l'ordre des villes visitées.
Les valeurs soulignées désignent les cumuls des distances parcourues + estimation de la distance restante

Branch and Bound avec sous estimation et programmation dynamique = A*

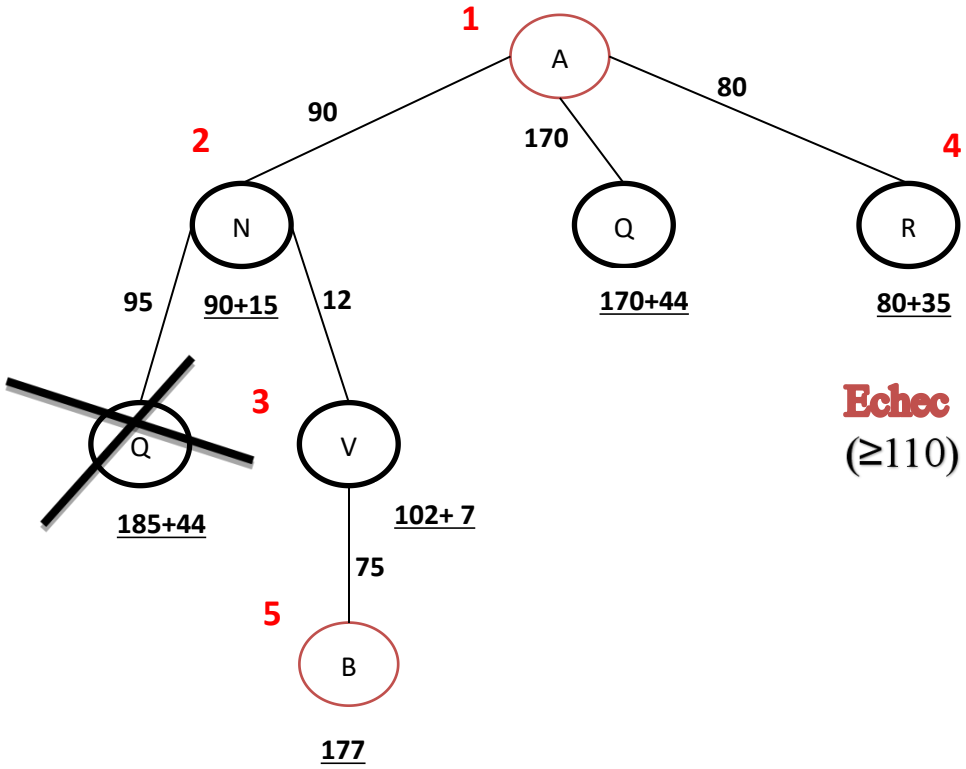
Liste : A

Liste : AN(105),AR(115),AQ(214)

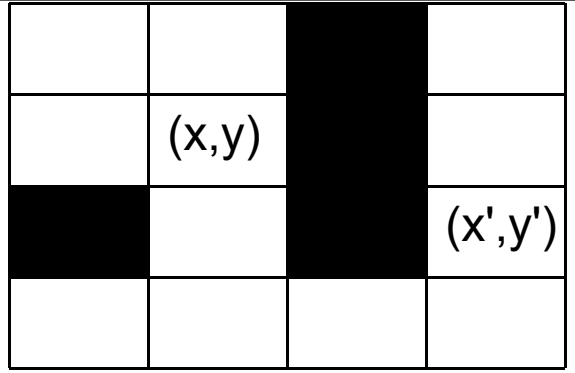
Liste : ANV(109), AR(115), AQ(214)

Liste : AR(115), ANVB(177), AQ(214),

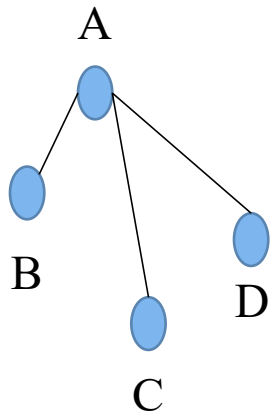
Liste : ANVB(177), AQ(214),



Heuristiques



1	2	3
8		4
7	6	5



Autres exemples

Labyrinthe.

L'estimation de la distance est donnée par la formule Racine carrée($(x'-x)^2 + (y'-y)^2$) sachant que (x, y) sont les coordonnées de la position courante et (x', y') les coordonnées de la position finale (sortie). On cumulera les distances des branches parcourues.

Jeu de taquin

L'estimation peut être le nombre de pièces mal placées. On cumulera les pas parcourus.

Problème du Voyageur de Commerce

L'estimation peut être la distance en vol d'oiseau. On cumulera les distances parcourues.