Guide Grand Oral NSI – Échecs : script, code Python commenté et questions jury

▶ SommaireSauter à une section
✓ L'essentiel

- Script minuté 10 minutes avec transitions rédigées et blocs de code Python commentés pour présentation orale.
- Minimax, alpha-bêta et AlphaZero expliqués avec analyse de coût et pseudo-code.
- 25 questions de jury classées du plus simple au plus technique, avec réponses complètes en langage oral.
- 3 problématiques au choix selon ton niveau : arbre de décision, optimisation alpha-bêta, IA par apprentissage.
- Fiche mémo notation O, vocabulaire NSI et méthodologie de l'épreuve (coefficient, posture, gestion du temps) à imprimer.

Ce **guide** rassemble **tout** ce qu'un élève de **terminale** spécialité **NSI** doit avoir en main pour réussir son **Grand Oral du bac** avec les échecs comme **sujet** : méthodologie de l'**épreuve**, **choix** de problématique parmi trois angles, script minuté de **10 minutes**, code Python commenté, **questions** de **jury** rédigées, anti-sèche et **conseils** de **posture** pour le **jour** J.

L'objectif : que tu n'aies plus à chercher d'**exemples**, plus à compiler des sources d'**internet** dispersées, plus à inventer ton plan dans l'urgence. **Tout** est là, exploitable directement.

> 📩 ***Comment utiliser ce guide ?*** *Télécharge le PDF via le bouton en haut de la page pour l'avoir hors-ligne. Lis-le sur écran ou impression selon ce qui te convient. Entraîne-toi à expliquer le code à voix haute ligne par ligne : c'est l'exercice clé pour l'**oral NSI**.*

Comprendre l'épreuve : Grand Oral en spécialité NSI

Coefficient, durée et déroulé minute par minute

Le **Grand Oral** est l'épreuve la mieux notée du **bac** général, à **coefficient 10**. Sa durée totale est de **40 minutes** :

Temps	Phase	Ce que tu fais	Ce que le jury observe
20 min	Préparation	On te donne deux **questions** issues de ton **programme** de spécialité. Tu choisis l'une des deux. Brouillon autorisé.	,
5 min	Exposé debout	Tu présentes ta **question** debout, sans notes (ou très peu).	**Posture**, voix, structure
10 min	Échange avec le **jury**	Le **jury** te questionne sur le **sujet**, ton plan, ton code.	Maîtrise, réactivité, ouverture
5 min	Projet d'**orientation**	Tu expliques comment ce **sujet** s'inscrit dans ton parcours post-bac.	Cohérence parcours

Sur les **10 minutes** d'exposé, tu joues 1 point coefficient par minute. C'est l'**épreuve** la plus rentable du **bac** au ratio temps de préparation / impact note.

Composition du jury et attendus

Le **jury** est composé de **deux** professeurs :

- L'un est issu de ta spécialité (typiquement **NSI** ou **mathématiques**)
- L'autre vient d'une discipline différente (lettres, histoire, philosophie, SVT…)

Le second n'est **pas** expert en **informatique**. Ton **oral** doit être compréhensible par un non-spécialiste : le code Python ne suffit pas, il faut le raconter en français clair. C'est le piège dans lequel tombent les candidats trop techniques.

Le rapport officiel du Bulletin officiel précise que le **jury** évalue quatre dimensions :

1. **Qualité orale** : élocution, **parole** posée, ton convaincu, gestes maîtrisés
2. **Maîtrise du sujet** : capacité à expliquer, justifier, nuancer
3. **Construction de l'argumentation** : problématique nette, plan annoncé, conclusion
4. **Cohérence avec ton orientation** : pourquoi ce **sujet** te prépare à tes études

Trois pièges à éviter absolument

- **Réciter par cœur** : le **jury** entend la mémorisation. Préfère un canevas que tu adaptes au moment.
- **Lire ses notes** : disqualifiant. Tu peux avoir un papier mais ne le regarde que pour vérifier une **donnée** chiffrée.
- **Dépasser le temps** : à 5 minutes pile, le **jury** coupe. Mieux vaut conclure 30 secondes en avance que d'être interrompu.

Choisir ta problématique

La problématique est ta colonne vertébrale. En **NSI**, elle doit montrer que tu maîtrises une notion ancrée dans le **programme** de **terminale** : récursivité, coût d'un calcul, arbres, ou intelligence artificielle.

Les trois angles recommandés

Angle A, Arbre de décision *(niveau accessible)*

> *« Comment un programme informatique peut-il prendre des décisions pour jouer aux échecs ? »*

Cet angle te fait raconter l'arbre des coups, la fonction d'évaluation, le **fonctionnement** de la récursivité. Bon **choix** si tu veux rester sur du solide et bien maîtrisé.

Angle B, Optimisation algorithmique *(niveau intermédiaire)*

> *« En quoi l'élagage alpha-bêta illustre-t-il l'optimisation d'une recherche arborescente ? »*

Plus technique, cet angle montre que tu sais analyser un coût en O(b^d) puis en O(b^(d/2)). C'est le **choix** privilégié quand tu maîtrises bien les **mathématiques** et que tu veux marquer des points sur le raisonnement.

Angle C, Intelligence artificielle et rupture épistémologique *(niveau avancé)*

> *« Pourquoi AlphaZero représente-t-il une rupture dans l'histoire de l'intelligence artificielle appliquée aux jeux de stratégie ? »*

Cet angle te place dans une réflexion plus large : opposition IA symbolique / IA connexionniste, **fonctionnement** des réseaux de neurones, **limites** de l'apprentissage automatique. À choisir si tu vises une école d'ingénieur ou une **mathématiques** spé.

Comment choisir ?

- **Si tu as Maths en spé complémentaire** : angle C est très différenciant.
- **Si tu vises une CPGE** ou une école d'ingénieur : angle B ou C.
- **Si tu veux jouer la sécurité** avec une excellente maîtrise : angle A.
- **Si tu débutes en Python** : angle A, mais maîtrise vraiment le code.

Autres idées de sujets Grand Oral en NSI

Pour situer le **choix** des échecs par rapport aux autres **sujets Grand Oral** possibles en **NSI**, voici un panorama des **idées de sujets** les plus solides :

Sujet	Forces principales	Limites	Difficulté
Sécurité informatique (RSA, chiffrement)	Actualité forte, liée à l'**internet** moderne	Niveau math élevé pour RSA, souvent traité	Moyenne
Cryptographie quantique	Sujet de pointe, impressionnant	Très exigeant, peu de candidats tiennent	Élevée
Réseaux sociaux et **algorithmes** de recommandation	Forte dimension de société	Peu de code à montrer, vite hors-programme	Moyenne
Systèmes embarqués (Raspberry Pi, IoT)	Très concret si projet matériel	Demande un projet à montrer le **jour** J	Moyenne
Échecs et IA *(notre choix)*	Combine algo + structures + IA, imaginaire culturel	Risque du récit pur si mal préparé	Faible avec ce guide
Bases de données massives (Lichess, GitHub)	Cas concret de big data, SQL réel	Demande de manier statistiques	Moyenne
Compression d'images / sons (PNG, MP3)	Math + structure	Sujet technique exigeant	Moyenne
Tri et **complexité** algorithmique	Cœur du programme	Considéré scolaire, difficile de **révolutionner**	Faible
Réseaux de neurones génératifs (ChatGPT, LLM)	Actualité brûlante	Risque de l'exposé non technique	Élevée
Cryptographie monétaire (Bitcoin, blockchain)	Économie + **informatique**	Sujet glissant : faux experts en circulation	Moyenne

Pourquoi les échecs gagnent

Trois critères distinguent un bon **sujet** d'un **sujet** moyen :

1. **Il contient du code visible** que tu peux décrire à voix haute
2. **Il a un enjeu** identifiable au-delà de la technique (société, science, philosophie)
3. **Il s'attache à un domaine** que le **jury** reconnaît instantanément

Les échecs cochent les trois. La **cryptographie** coche les deux premiers mais peu le troisième pour un **jury** lettres. Les **réseaux sociaux** cochent les deux derniers mais pas le premier.

Construire ta problématique : méthode pas à pas

Étape 1, Choisir un angle parmi les trois

Tu choisis A, B ou C selon ton niveau. Marque ce **choix** sur ton brouillon en gros caractères, n'en change plus.

Étape 2, Tester la problématique avec le « test du tiers »

Énonce ta problématique à un tiers (parent, ami, professeur d'une autre discipline). S'il dit « je vois ce que tu vas raconter », c'est bon. S'il dit « explique-moi mieux », c'est trop technique ou trop flou : reformule.

Étape 3, Vérifier qu'elle est *problématisable*

Une bonne problématique présente une **tension** : « comment » plutôt que « qu'est-ce que ». Elle suggère qu'il y a un débat ou un mécanisme à comprendre. « Qu'est-ce que minimax ? » est une mauvaise problématique (descriptive). « Comment minimax permet-il de jouer en **temps** raisonnable ? » est meilleure (causale).

Étape 4, Vérifier qu'elle s'aligne avec ton orientation

Si tu vises une école d'ingénieur, ta problématique doit pouvoir conduire au passage « voilà pourquoi je veux étudier l'**informatique** ». Si elle ne le permet pas, c'est qu'elle est mal posée.

Script minuté : problématique B (optimisation alpha-bêta)

Les transitions rédigées sont en italique. Le code Python est dans des blocs à expliquer oralement : tu n'as pas à le lire mot à mot, mais à l'expliquer en français clair.

⏱ 0:00–1:00, Introduction et problématique

> *« Bonjour. Je vais vous parler de la façon dont un ordinateur joue aux échecs, et plus précisément, comment il choisit son coup en un **temps** raisonnable.*
>
> *Les échecs ont environ 10^120 parties possibles : plus que d'atomes dans l'univers observable. Un programme qui explore tout est impossible. Ma problématique : en quoi l'élagage alpha-bêta illustre-t-il l'optimisation d'une recherche arborescente ?*
>
> *Je développerai en trois étapes : l'algorithme minimax de base, l'élagage alpha-bêta qui le rend efficace, puis les **limites** de cette approche et ce qu'AlphaZero a changé. »*

⏱ 1:00–4:00, Partie 1 : l'algorithme minimax

> *« L'idée de base : modéliser le jeu comme un arbre. Chaque nœud est une position, chaque branche un coup possible. Le joueur blanc cherche à maximiser son avantage, le noir à le minimiser : d'où le nom minimax.*
>
> *Voici le code en Python : »*

```
def minimax(position, profondeur, maximise):
    # Cas de base : profondeur atteinte ou partie terminée
    if profondeur == 0 or position.est_terminee():
        return evaluer(position)   # retourne un score numérique

    if maximise:  # tour des Blancs : on cherche le maximum
        meilleur = -infini
        for coup in position.coups_possibles():
            position.joue(coup)
            score = minimax(position, profondeur-1, False)
            position.annule(coup)
            meilleur = max(meilleur, score)
        return meilleur

    else:  # tour des Noirs : on cherche le minimum
        meilleur = +infini
        for coup in position.coups_possibles():
            position.joue(coup)
            score = minimax(position, profondeur-1, True)
            position.annule(coup)
            meilleur = min(meilleur, score)
        return meilleur
```

> *« Trois points clés : la récursivité (la fonction s'appelle elle-même), l'alternance maximise/minimise (les deux joueurs alternent), et la fonction* `evaluer()` *qui attribue un score à chaque position.*
>
> *Quel coût ? Avec b ≈ 35 coups possibles par position et d = 4 niveaux de profondeur, le minimax explore 35^4 ≈ 1,5 million de positions. Sur 6 niveaux : 35^6 ≈ 1,8 milliard. La croissance est exponentielle, O(b^d). »*

⏱ 4:00–7:00, Partie 2 : l'élagage alpha-bêta

> *« Le problème du minimax : il explore des branches inutiles. L'élagage alpha-bêta coupe ces branches sans changer le résultat.*
>
> *L'idée : si je sais déjà qu'une branche ne peut pas améliorer ma meilleure option connue, je l'ignore. »*

```
def alpha_beta(position, profondeur, alpha, beta, maximise):
    if profondeur == 0 or position.est_terminee():
        return evaluer(position)

    if maximise:
        meilleur = -infini
        for coup in position.coups_possibles():
            position.joue(coup)
            score = alpha_beta(position, profondeur-1, alpha, beta, False)
            position.annule(coup)
            meilleur = max(meilleur, score)
            alpha = max(alpha, meilleur)
            if beta <= alpha:     # ← coupure bêta
                break             # cette branche ne sera jamais choisie
        return meilleur

    else:
        meilleur = +infini
        for coup in position.coups_possibles():
            position.joue(coup)
            score = alpha_beta(position, profondeur-1, alpha, beta, True)
            position.annule(coup)
            meilleur = min(meilleur, score)
            beta = min(beta, meilleur)
            if beta <= alpha:     # ← coupure alpha
                break             # cette branche ne sera jamais choisie
        return meilleur
```

> *« `alpha` = meilleur score garanti pour les Blancs. `beta` = meilleur score garanti pour les Noirs. Quand `beta ≤ alpha`, il est inutile d'explorer davantage : les deux joueurs ne choisiraient jamais cette branche.*
>
> *Gain pratique : dans le meilleur cas (coups triés par ordre de qualité), alpha-bêta réduit le coût à O(b^(d/2)). Soit 35^2 = 1 225 positions pour d = 4 au lieu de 1,5 million. On passe de l'impossible au réalisable en quelques millisecondes. »*

⏱ 7:00–9:00, Partie 3 : limites et rupture AlphaZero

> *« L'élagage alpha-bêta est élégant, mais avec des **limites** structurelles.*
>
> ***Limite 1 : la fonction d'évaluation.*** `evaluer()` *est écrite par des humains. Elle encode les intuitions d'experts (valeur des pièces, contrôle du centre, sécurité du roi). Ces règles peuvent être incorrectes ou incomplètes.*
>
> ***Limite 2 : la profondeur.*** *Même avec alpha-bêta, la profondeur reste limitée à 20-30 coups sur les machines les plus puissantes. Au-delà, c'est impossible.*
>
> ***La rupture d'AlphaZero (2017) :*** *AlphaZero n'a pas de fonction d'évaluation écrite par des humains. Il apprend à jouer en jouant contre lui-même (apprentissage par renforcement) et développe sa propre évaluation à travers un réseau de neurones. En 4 heures d'entraînement autonome, AlphaZero a battu Stockfish, le meilleur programme minimax au monde.*
>
> *Ce n'est plus une optimisation : c'est un paradigme différent. Au lieu de chercher dans un arbre, il apprend à reconnaître des positions. »*

⏱ 9:00–10:00, Conclusion

Pour les autres angles : variantes de script

Si tu choisis l'**angle A (arbre de décision)** ou l'**angle C (rupture AlphaZero)**, voici les bascules à effectuer :

Variante angle A (arbre de décision)

- Remplace la partie 2 (alpha-bêta) par une **partie 2 dédiée à la fonction d'évaluation** : comment on attribue un score numérique à une position. Évoque les pondérations classiques : dame = 9, tour = 5, fou/cavalier = 3, pion = 1, plus bonus de position.
- Garde minimax en partie 1 mais simplifie le code (sans alpha-bêta).
- En partie 3, parle de Deep Blue (1997) plutôt qu'AlphaZero.

Variante angle C (rupture AlphaZero)

- Inverse la balance : 2 minutes sur minimax/alpha-bêta (rapidement), puis 5 minutes sur AlphaZero.
- Insiste sur les **systèmes** de réseaux résiduels, MCTS guidé par politique, apprentissage par renforcement (auto-jeu).
- Termine sur l'**ouverture** : AlphaZero → AlphaFold (médecine), MuZero (jeux sans règles).

Anticiper l'épreuve : 25 questions de jury rédigées

*Le **jury NSI** pose souvent des **questions** sur le code, le coût, et les distinctions algorithmiques. Prépare-toi à tracer une exécution à la main sur brouillon.*

Niveau 1, Comprendre le vocabulaire (questions de base)

Q1. Pourquoi avoir choisi les échecs pour votre Grand Oral NSI ?

> *« Les échecs concentrent plusieurs notions clés du **programme** de spé : la récursivité avec minimax, les arbres de décision, le coût algorithmique, et plus récemment l'apprentissage automatique avec AlphaZero. C'est un cas d'étude historiquement documenté (de Deep Blue en 1997 à AlphaZero en 2017) qui permet de voir l'évolution des paradigmes en intelligence artificielle. »*

Q2. Expliquez la récursivité dans minimax en une phrase.

> *« Minimax est récursif parce qu'il définit la valeur d'une position à partir des valeurs de ses positions enfants, et chaque position enfant est évaluée par le même minimax, jusqu'à atteindre une profondeur limite ou une position terminale. »*

Q3. Qu'est-ce qu'un arbre de jeu ?

> *« Un arbre de jeu est une structure de **données** où la racine représente la position actuelle, chaque nœud représente une position du jeu, et chaque arête représente un coup possible. Les feuilles sont soit des positions terminales (fin de partie), soit des nœuds où on atteint la profondeur limite et où la fonction d'évaluation s'applique. »*

Q4. Quelle est la différence entre un nœud MAX et un nœud MIN dans minimax ?

> *« Un nœud MAX correspond au tour du joueur qui veut maximiser son score : typiquement les Blancs. On y choisit le coup qui donne le score le plus élevé parmi les enfants. Un nœud MIN correspond au tour du joueur qui veut minimiser le score : les Noirs. L'alternance des nœuds MAX et MIN simule les deux joueurs qui prennent des décisions optimales à tour de rôle. »*

Q5. Calculez le coût de minimax pour b=30 coups et d=3 niveaux.

> *« O(b^d) = 30^3 = 27 000 positions à explorer. Avec alpha-bêta dans le meilleur cas : O(b^(d/2)) = 30^1,5 = 30 × √30 ≈ 30 × 5,5 ≈ 165 positions. Le gain est d'un facteur 163 : on passe de 27 000 à 165 évaluations. »*

Niveau 2, Comprendre les mécanismes (questions intermédiaires)

Q6. Expliquez la condition `if beta <= alpha : break` dans le code alpha-bêta.

> *« Cette condition est la coupure alpha-bêta. `alpha` est le meilleur score que les Blancs peuvent garantir sur la branche actuelle. `beta` est le meilleur score que les Noirs peuvent garantir. Si `beta ≤ alpha`, les Noirs n'accepteront jamais ce résultat : ils ont déjà une meilleure option ailleurs. Il est donc inutile d'explorer les coups restants sur cette branche : elle sera ignorée par le joueur noir quoi qu'il arrive. »*

Q7. Quelle est la différence entre le coût dans le meilleur cas et dans le pire cas pour alpha-bêta ?

> *« Dans le meilleur cas (coups triés du meilleur au pire), alpha-bêta atteint O(b^(d/2)) : il coupe environ la moitié des branches. Dans le pire cas (coups triés dans le mauvais ordre), il n'effectue aucune coupure et se réduit à minimax : O(b^d). En pratique, avec un tri heuristique des coups (par valeur de capture, par position connue), on est entre les deux : environ O(b^(3d/4)). »*

Q8. Qu'est-ce que la fonction d'évaluation et pourquoi est-elle cruciale ?

> *« La fonction d'évaluation attribue un score numérique à chaque position non terminale atteinte à la profondeur limite. Elle encode la connaissance humaine du jeu : valeur des pièces (dame = 9 points, tour = 5…), contrôle du centre, sécurité du roi, structure de pions. C'est le « jugement » du programme sur la qualité d'une position. Une mauvaise fonction d'évaluation produit un joueur faible même avec un algorithme parfait : c'est la **limite** principale de l'approche. »*

Q9. Pourquoi dit-on que l'algorithme minimax suppose un adversaire optimal ?

> *« Minimax suppose que l'adversaire joue toujours le meilleur coup possible : il minimise toujours. Si l'adversaire fait une erreur, minimax reste correct : la position résultante est encore meilleure pour nous. En revanche, minimax ne cherche pas à exploiter les erreurs de l'adversaire de façon proactive : il suppose simplement qu'elles n'arrivent pas. Cette hypothèse de jeu optimal est pessimiste mais sûre. »*

Q10. Comment Deep Blue (1997) différait-il de l'algorithme minimax pur ?

> *« Deep Blue utilisait alpha-bêta avec des optimisations massives : une bibliothèque de débuts (ouvertures connues) pour ne pas chercher dans les 20 premiers coups, une bibliothèque de finales (positions résolues), et une extension de recherche sélective (explorer plus profondément les positions tactiquement complexes). Il évaluait 200 millions de positions par seconde sur hardware dédié. Ce n'était pas du minimax pur, mais alpha-bêta hautement optimisé avec expertise humaine encodée. »*

Q11. Qu'est-ce que l'apprentissage par renforcement qu'AlphaZero utilise ?

> *« L'apprentissage par renforcement est un paradigme où l'agent apprend en interagissant avec son environnement et en recevant des récompenses. AlphaZero joue contre lui-même (self-play) : il reçoit +1 s'il gagne, -1 s'il perd, 0 pour une nulle. Le réseau de neurones ajuste ses paramètres pour maximiser la récompense cumulée. Après des millions de parties contre lui-même, le réseau apprend des stratégies que personne ne lui a enseignées. »*

Q12. Quelle est la structure du réseau de neurones dans AlphaZero ?

> *« AlphaZero utilise un réseau résiduel profond (ResNet) avec deux têtes de sortie. La tête de valeur retourne un nombre entre -1 et 1 : l'estimation de qui va gagner depuis cette position. La tête de politique retourne une distribution de probabilité sur les coups possibles : lesquels méritent d'être explorés. Ces deux sorties guident une recherche Monte Carlo Tree Search (MCTS) qui remplace le minimax classique. »*

Niveau 3, Nuances et limites (questions avancées)

Q13. Pourquoi AlphaZero est-il plus difficile à analyser qu'un programme minimax ?

> *« Un programme minimax est explicable : pour chaque coup choisi, on peut retracer l'arbre et voir les variations considérées. AlphaZero est une boîte noire : le réseau de neurones a des centaines de millions de paramètres, et on ne peut pas dire « pourquoi » il joue un coup en termes de règles. C'est le problème général de l'explicabilité des réseaux de neurones profonds : leur performance est remarquable mais leur raisonnement est opaque. »*

Q14. Peut-on appliquer l'algorithme minimax à d'autres jeux que les échecs ?

> *« Oui : à tout jeu à deux joueurs, information parfaite, sans hasard et à nombre fini de coups. Le morpion, les dames, le Go, le puissance 4, l'Othello. Le puissance 4 a été résolu par minimax en 1988 : les Blancs gagnent toujours avec un jeu parfait. Le morpion est trivial. Le Go a longtemps résisté (trop grand) jusqu'à AlphaGo. Les jeux avec hasard ou information cachée (poker) nécessitent des extensions du minimax. »*

Q15. Comment le tri des coups améliore-t-il alpha-bêta en pratique ?

> *« En explorant d'abord les coups les plus prometteurs (captures, coups qui donnent un bel avantage), on maximise les chances de trouver rapidement un bon* `alpha` *. Plus* `alpha` *est élevé tôt, plus on peut couper de branches. Un bon tri transforme un alpha-bêta au pire cas en un alpha-bêta proche du meilleur cas. En pratique, les programmes modernes utilisent des heuristiques comme « coups de tueur » (killer heuristic) et la table de transposition pour trier efficacement. »*

Q16. Qu'est-ce qu'une table de transposition en algorithmique des jeux ?

> *« Une table de transposition est une table de hachage qui mémorise les positions déjà évaluées avec leur score. Aux échecs, la même position peut être atteinte par des séquences de coups différentes ; sans table de transposition, on l'évalue plusieurs fois. La table évite ce recalcul : si la position est en mémoire, on retourne directement son score. C'est une application du principe de mémoïsation (ou programmation dynamique) à la recherche arborescente. »*

Q17. Pourquoi les bitboards accélèrent-ils les programmes d'échecs ?

> *« Un bitboard représente l'échiquier comme un entier de 64 bits, où chaque bit correspond à une case. Les opérations sur les bits (AND, OR, XOR, décalage) sont extrêmement rapides sur les processeurs modernes : une seule instruction CPU peut tester 64 cases simultanément. La génération des coups légaux, qui doit être faite des millions de fois par seconde, bénéficie massivement de cette représentation. C'est un* ***exemple*** *d'optimisation bas-niveau qui change l'ordre de grandeur des performances. »*

Q18. Stockfish est-il toujours de type alpha-bêta ou a-t-il adopté des réseaux de neurones ?

> *« Depuis Stockfish 12 (2020), Stockfish intègre NNUE (Efficiently Updatable Neural Network). C'est un réseau de neurones utilisé comme fonction d'évaluation, mais l'algorithme de recherche reste alpha-bêta. C'est une approche hybride : la structure de recherche classique avec une évaluation apprise par réseau de neurones. Résultat : Stockfish a gagné 100 à 150 points Elo d'un coup. Aujourd'hui, les deux approches (minimax+NNUE et pure NN comme Lc0) se disputent la première place. »*

Q19. Comment AlphaZero utilise-t-il Monte Carlo Tree Search ?

> *« AlphaZero combine le réseau de neurones avec MCTS. Le réseau guide quels nœuds explorer en priorité (la tête de politique donne des probabilités sur les coups). MCTS explore ces nœuds de façon stochastique, accumule des statistiques de victoire/défaite, et retourne le coup le plus visité. Le réseau est entraîné sur les résultats de ces recherches. C'est une boucle : le réseau guide MCTS, et MCTS produit des* ***données*** *pour améliorer le réseau. »*

Q20. Un algorithme parfait aux échecs est-il possible en principe ?

> *« En principe oui : le théorème de Zermelo (1913) garantit que les échecs ont un résultat optimal sous jeu parfait. Cet algorithme parfait serait un minimax à profondeur infinie : explorer tout l'arbre. En pratique, avec 10^120 feuilles, c'est physiquement impossible même pour toute la puissance de calcul de l'univers pendant toute son histoire. Les échecs ne seront jamais « résolus » comme les dames (2007) ou le morpion : leur taille les place hors de portée de la force brute. »*

Niveau 4, Ouvertures et liens transversaux

Q21. Les techniques d'IA des échecs s'appliquent-elles à d'autres domaines ?

> *« Oui, c'est l'un des intérêts majeurs. DeepMind a appliqué les principes d'AlphaZero à AlphaFold (repliement de protéines, prix Nobel de chimie 2024) et à MuZero (jeux dont on ne connaît même pas les règles). Les **mathématiques** de l'apprentissage par renforcement viennent aussi des recherches sur les jeux. C'est un domaine de recherche où les échecs sont le banc d'essai historique. »*

Q22. Quels enjeux de sécurité informatique soulève l'IA dans les échecs en ligne ?

> *« La principale **question** est la détection de triche. Les moteurs étant plus forts que tout humain, leur usage en compétition en ligne ou en présentiel est interdit. Cela soulève des **systèmes** de détection : analyse statistique de la précision (combien de fois le joueur a joué le coup que l'engin recommande), test sur des positions atypiques, **fonctionnement** des serveurs comme Chess.com qui ont des équipes anti-triche. C'est un cas concret de **sécurité informatique** appliquée. »*

Q23. Quel rôle jouent les bases de données dans les programmes d'échecs ?

> *« Plusieurs **systèmes** de bases sont utilisés : (1) Les bibliothèques d'ouvertures (Encyclopaedia of Chess Openings, ECO) qui stockent des millions de parties humaines pour les 20 premiers coups. (2) Les bases de finales (Syzygy tablebase) qui résolvent toutes les positions à 7 pièces ou moins (1,5 To de **données**). (3) Les bases de parties (Lichess publie sa base ouverte de 4 milliards de parties au format PGN). C'est un **exemple** de big data appliqué à un domaine bien défini. »*

Q24. Comment les échecs illustrent-ils le débat sur les limites de l'IA ?

> *« Les échecs ont nourri trois grandes étapes du débat. (1) Deep Blue 1997 : « les machines peuvent battre les humains aux tâches calculatoires ». (2) AlphaZero 2017 : « les machines peuvent apprendre sans connaissance humaine ». (3) GPT-4 (2024) jouant aux échecs mal mais avec un raisonnement linguistique : « la généralité n'implique pas l'expertise ». Les échecs sont un fil rouge pour mesurer ce que l'IA peut et ne peut pas faire. »*

Q25. Si tu devais reprendre ton sujet dans dix ans, qu'y changerais-tu ?

> *« En dix ans, l'IA aura probablement intégré encore plus le quotidien. Je m'attendrais à ce que les programmes d'échecs soient utilisés pour comprendre les **réseaux** de neurones eux-mêmes : pourquoi un coup est-il jugé meilleur qu'un autre ? Les chercheurs commencent à « interpréter » AlphaZero, à extraire des principes stratégiques que le réseau a découverts. Dans dix ans, ce sera peut-être la principale leçon : utiliser les IA pour comprendre les IA. »*

Anti-sèche imprimable : à plier dans ta poche

Voici une fiche mémo concentrée que tu peux imprimer ou recopier sur une feuille A5. Quand tu doutes, tu sors cette fiche dans ta tête.

Les trois chiffres à connaître par cœur

- **35** : facteur de branchement moyen aux échecs (b)
- **10^120** : nombre de Shannon (parties possibles)
- **28-0-72** : score d'AlphaZero vs Stockfish en 2017 (100 parties)

Les trois dates pivots

- **1950** : Turing et Shannon, fondations algorithmiques
- **1997** : Deep Blue bat Kasparov (symbolique, force brute)
- **2017** : AlphaZero bat Stockfish en 4 h (connexionniste, apprentissage)

Les trois notations de coût à manier

- **O(b^d)** : minimax pur (exponentielle)
- **O(b^(d/2))** : alpha-bêta meilleur cas (racine)
- **O(1)** : accès table de transposition (constante)

Les trois mots-clés à placer

- **Récursivité** (cœur de minimax)
- **Mémoïsation** (table de transposition)
- **Renforcement** (apprentissage AlphaZero)

Fiche mémo NSI complète

```
╔═══════════════════════════════════════════════════════════╗
║          COÛTS ALGORITHMIQUES, GRAND ORAL NSI             ║
╠═══════════════════════════════════════════════════════════╣
║ MINIMAX (sans optimisation)                               ║
║   Coût temporel : O(b^d)                                  ║
║   b = facteur de branchement (≈35 aux échecs)             ║
║   d = profondeur de recherche                             ║
║   Exemple : b=35, d=4 → 35^4 ≈ 1 500 000 positions        ║
╠═══════════════════════════════════════════════════════════╣
║ ALPHA-BÊTA (meilleur cas : coups bien triés)              ║
║   Coût temporel : O(b^(d/2))                              ║
║   Exemple : b=35, d=4 → 35^2 = 1 225 positions            ║
║   Gain facteur b^(d/2) : environ 1000× sur d=6            ║
╠═══════════════════════════════════════════════════════════╣
║ PARAMÈTRES ALPHA-BÊTA                                     ║
║   alpha : meilleur score garanti pour le maximiseur       ║
║   beta  : meilleur score garanti pour le minimiseur       ║
║   Coupure : si beta ≤ alpha → branche ignorée             ║
╠═══════════════════════════════════════════════════════════╣
║ VOCABULAIRE CLÉ                                           ║
║   Récursivité   : fonction qui s'appelle elle-même        ║
║   Élagage       : suppression de branches inutiles        ║
║   Heuristique   : règle approchée pour guider la recherche ║
║   Mémoïsation   : stocker pour éviter recalcul            ║
║   Transposition : même position via chemins différents    ║
╠═══════════════════════════════════════════════════════════╣
║ COMPARAISON DES APPROCHES                                 ║
║   Minimax / Alpha-bêta : règles explicites, explicable    ║
║   AlphaZero / MCTS+NN  : apprentissage, boîte noire       ║
║   Stockfish NNUE       : hybride (alpha-bêta + réseau)    ║
╚═══════════════════════════════════════════════════════════╝
```

Conseils pour réussir le jour J

Posture et présence

Tu es debout pendant les **cinq minutes** d'exposé. Les détails comptent :

- **Pieds ancrés** à largeur d'épaules, **jamais** croisés
- **Mains visibles** : sur la table, en mouvement, jamais dans les poches
- **Regard distribué** entre les deux membres du **jury** : 60% pour celui de ta spécialité, 40% pour l'autre
- **Voix** : projetée, posée, **pas** monotone. Marque les pauses après les chiffres clés
- **Sourire** au moins une fois pendant l'introduction et une fois en conclusion

Gestion du stress avant l'épreuve

Le **stress** d'avant **oral** est normal et même utile (il booste la concentration). Trois techniques rapides :

1. **Respiration carrée** (4-4-4-4) : inspire 4 s, retiens 4 s, expire 4 s, pause 4 s. Trois cycles juste avant d'entrer dans la salle.
2. **Ancrage corporel** : pieds bien à plat dans le sol, poings serrés 5 secondes puis relâche. Reconnecte au corps.
3. **Premier mot automatique** : ta première phrase doit être prête par cœur. Si tu démarres sans hésiter, le reste suit.

Gestion du temps pendant l'exposé

- **Chronométrage discret** : pose ta montre face à toi, ou utilise un chrono visible. Si tu n'as ni l'un ni l'autre, marque mentalement à 5 minutes (mi-exposé).
- **Plan en repère** : à chaque transition, dis « première partie », « deuxième partie » : cela aide le **jury** à suivre **et** te recale dans ta progression.
- **Si tu accélères** par stress : repère un fait technique (un chiffre, un nom) et ralentis dessus. Une seconde de pause vaut mieux qu'un débit qui s'emballe.

Pendant l'échange : 10 minutes de questions

Le **jury** te questionne. Quelques règles :

- **Reformule la question** dans tes mots avant de répondre. Cela te donne du temps de réflexion.
- **Si tu ne sais pas**, dis « je ne suis pas certain mais je dirais que… » plutôt que « je ne sais pas » sec.
- **Distingue les niveaux** : si on te demande « définissez la récursivité », commence par une phrase simple puis nuance.
- **Quand tu ne comprends pas**, demande poliment « pourriez-vous reformuler ? » plutôt que de répondre à côté.

Le projet d'orientation : les 5 dernières minutes

C'est la partie où beaucoup décrochent par fatigue. Tiens bon. Prépare une réponse courte et cohérente :

- Si tu vises une **CPGE** : « Ce **sujet** m'a fait découvrir le raisonnement formel, c'est ce que je veux approfondir en maths-info ou MPI. »
- Si tu vises une **école d'ingénieur** post-bac : « L'algorithmique appliquée à un cas concret m'a passionné, je veux continuer dans un cursus pratique comme l'INSA ou l'UTC. »
- Si tu vises un **BUT informatique** : « J'ai préféré le code à la théorie pure, je veux un cursus qui privilégie la pratique. »
- Si tu vises une **licence informatique** ou **mathématiques** : « J'ai aimé manipuler les **mathématiques** discrètes et les structures de **données**, je veux les creuser en université. »

Important : ne mens pas sur ton projet. Le **jury** sent l'incohérence. Un projet « en construction » est mieux accepté qu'un projet inventé.

Check-list de préparation

J-30 (un mois avant)

- [ ] Choisir la problématique parmi les trois proposées
- [ ] Implémenter minimax simple en Python (même pour morpion, c'est suffisant)
- [ ] Tracer l'arbre minimax à la main pour une position à 2 niveaux (exercice essentiel)
- [ ] Mémoriser les trois dates : 1950, 1997, 2017
- [ ] Lire au moins deux **questions** du **jury** par jour à voix haute

J-15 (deux semaines avant)

- [ ] Premier entraînement chrono : 10 minutes exposé complet, seul, devant miroir
- [ ] Lister les **mathématiques** spécifiques à maîtriser (b^d, racine carrée, logarithme)
- [ ] Préparer les feuilles de code imprimées (à pouvoir poser sur la table le jour J)
- [ ] Identifier deux ou trois **exemples** numériques à connaître par cœur (35^4, 30^3, etc.)

J-7 (une semaine avant)

- [ ] Deuxième entraînement chrono avec un proche jouant le **jury**
- [ ] Répondre à au moins 10 **questions** du **jury** à voix haute
- [ ] Être capable d'expliquer chaque ligne du code sans le lire
- [ ] Revoir la fiche mémo complète

J-2 (l'avant-veille)

- [ ] Dernier entraînement complet, chrono pris, exposé filmé pour révision
- [ ] Vérifier la tenue (pantalon ou jupe propre, chemise/chemisier, chaussures fermées)
- [ ] Préparer le sac : pièce d'identité, convocation, stylos, eau

J-1 (la veille)

- [ ] Relire l'anti-sèche imprimable (cinq minutes max, **pas** plus)
- [ ] Vérifier l'heure et l'adresse de la convocation
- [ ] Se coucher tôt : le **stress** dérègle déjà le sommeil, anticipe

Jour J

- [ ] Manger normalement le matin (**pas** plus, **pas** moins ; ventre vide accentue le stress)
- [ ] Arriver 30 minutes en avance
- [ ] Eau dans la salle d'attente, **pas** de café à jeun
- [ ] Phase de préparation : 20 minutes pour choisir et organiser. Ne perds **pas** plus de 2 minutes à choisir.

Sources et références

- **Bulletin officiel, Note de service 2020-014.** Modalités du Grand Oral au baccalauréat général. (Cadre réglementaire de l'**épreuve**.)
- **Shannon, C. E. (1950).** *Programming a Computer for Playing Chess.* *Philosophical Magazine*. (Fondation de l'algorithme minimax.)
- **Silver, D., et al. (DeepMind, 2018).** *A general reinforcement learning algorithm that masters chess, shogi, and Go.* *Science*, 362(6419). (AlphaZero : apprentissage par renforcement et MCTS.)
- **Knuth, D. & Moore, R. (1975).** *An Analysis of Alpha-Beta Pruning.* *Artificial Intelligence*. (Analyse formelle de la coupure alpha-bêta.)
- **Documentation Stockfish.** *Stockfish Chess Engine, Source code.* GitHub. (Référence pour l'implémentation réelle.)
- **Lichess Open Database.** *lichess.org/database*. (Base de **données** publique de parties d'échecs au format PGN.)
- **Sadler, M. & Regan, N. (2019).** *Game Changer : AlphaZero's Groundbreaking Chess Strategies*. New In Chess. (Analyse du style d'AlphaZero, accessible aux lycéens.)
- **Russell, S. & Norvig, P.** *Artificial Intelligence : A Modern Approach* (4e éd., 2020). Pearson. (Manuel de référence IA pour CPGE, chapitres 5-6 sur minimax.)

*Ce guide est librement utilisable et imprimable. Si d'autres lycéens préparent le même **sujet**, partage-leur le lien. Et bonne chance : tu as fait le travail.*

Questions fréquentes

Quelle problématique choisir pour un Grand Oral NSI avec les échecs ?

Trois angles solides : (1) 'Comment un programme informatique peut-il jouer aux échecs ?' : idéal pour explorer l'arbre minimax, la récursivité et la fonction d'évaluation. (2) 'En quoi l'élagage alpha-bêta illustre-t-il l'optimisation d'une recherche arborescente ?' : plus technique, montre une réduction de O(b^d) à O(b^(d/2)). (3) 'Pourquoi AlphaZero représente-t-il une rupture dans l'histoire de l'intelligence artificielle appliquée aux jeux ?' : angle épistémologique, idéal avec une spécialité complémentaire Maths ou Philosophie.

Peut-on montrer du code Python au jury du Grand Oral ?

Oui : c'est même recommandé en NSI. Tu peux avoir des feuilles imprimées avec ton code. Le jury peut te demander d'expliquer une ligne précise, de tracer l'exécution, ou d'identifier un bug hypothétique. Le code doit être commenté (une ligne de commentaire pour chaque bloc logique) et tu dois être capable d'en parler sans le lire mot à mot.

Comment expliquer le coût d'un algorithme au jury sans perdre le fil ?

Utilise toujours un exemple numérique avant la notation O. 'Le minimax explore 35 coups possibles à chaque niveau. Sur 4 niveaux de profondeur, ça fait 35^4 = 1,5 million de positions. C'est une croissance exponentielle, O(b^d) avec b=35 et d=4.' Ensuite seulement tu montres comment alpha-bêta réduit ça. Le jury apprécie le raisonnement, pas la récitation.

AlphaZero est-il au programme de Terminale NSI ?

Pas directement, mais l'apprentissage par renforcement et les réseaux de neurones sont des thèmes cohérents avec le programme. AlphaZero est un exemple spectaculaire d'apprentissage : il illustre concrètement la différence entre une IA par règles explicites (minimax) et une IA par apprentissage (deep reinforcement learning). Le jury de Grand Oral valorise les exemples concrets et bien maîtrisés, même s'ils dépassent légèrement le programme.

Quelle est la durée idéale de chaque partie de l'exposé NSI ?

Pour 10 minutes : introduction 1 minute (problématique + plan), partie 1 de 3 minutes (algo minimax + code), partie 2 de 3 minutes (alpha-bêta + optimisation), partie 3 de 2 minutes (AlphaZero + limites), conclusion 1 minute. Le jury interroge ensuite pendant 10 minutes. Chronomètre-toi impérativement : le jury coupe à 10 minutes.