SAE S1.02 – Comparaison d’approches algorithmiques

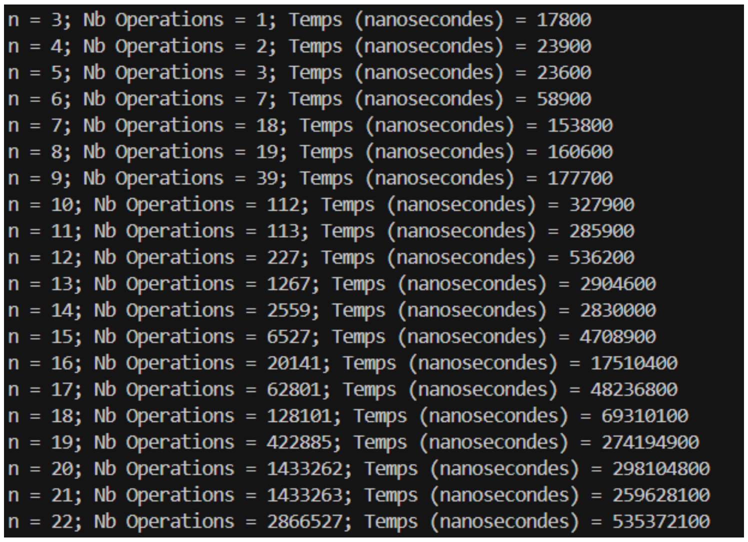
Elwan YVIN - Emile THEVENIN

I. Introduction : Objectifs du travail

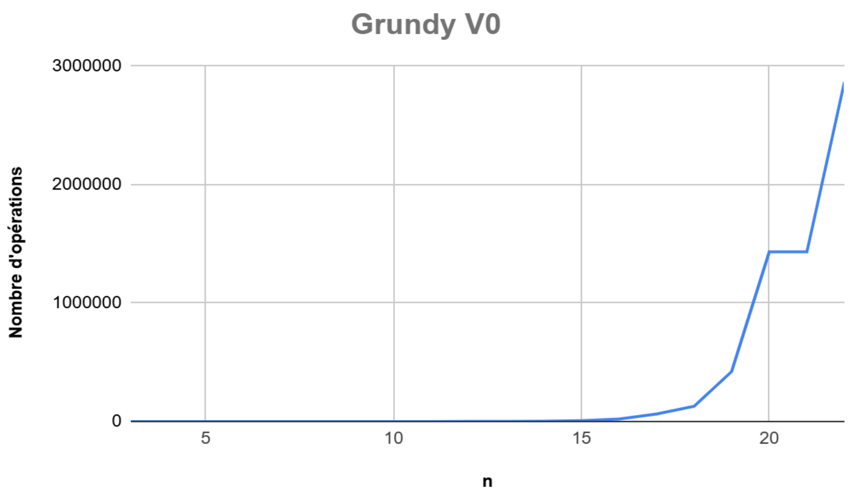
Au cours de cette SAE, nous avons cherché à comprendre, faire marcher puis améliorer l’efficacité d’une intelligence artificielle (fonctionnant de manière récursive) fournie dès le départ, pour le jeu de Grundy. Nous avons tout d’abord codé une boucle de jeu et un test d’efficacité, implémenté en version 0, puis nous avons ajouté des améliorations des améliorations succinctes pour chaque version, augmentant l’efficacité de l’ordinateur. Nous avons ensuite mesuré et consigné dans des tableaux le temps de réflexion de l’ordinateur, pour chaque version et pour des tailles de jeu variant de petites à très grandes. Nous avons finalement utilisé ses données pour tracer des graphes comparant l’efficacité des différentes versions du programme et interprété les données pour en tirer des conclusions sur l’efficacité de l’intelligence artificielle récursive du jeu de Grundy avec ses améliorations.

II. Graphiques et Tableaux de Complexité

Pour l'ensemble des tests d'efficacité, n a été incrémenté jusqu'à ce que le temps d'exécution de la méthode soit supérieur à 500 000 000 (sauf pour la dernière version).

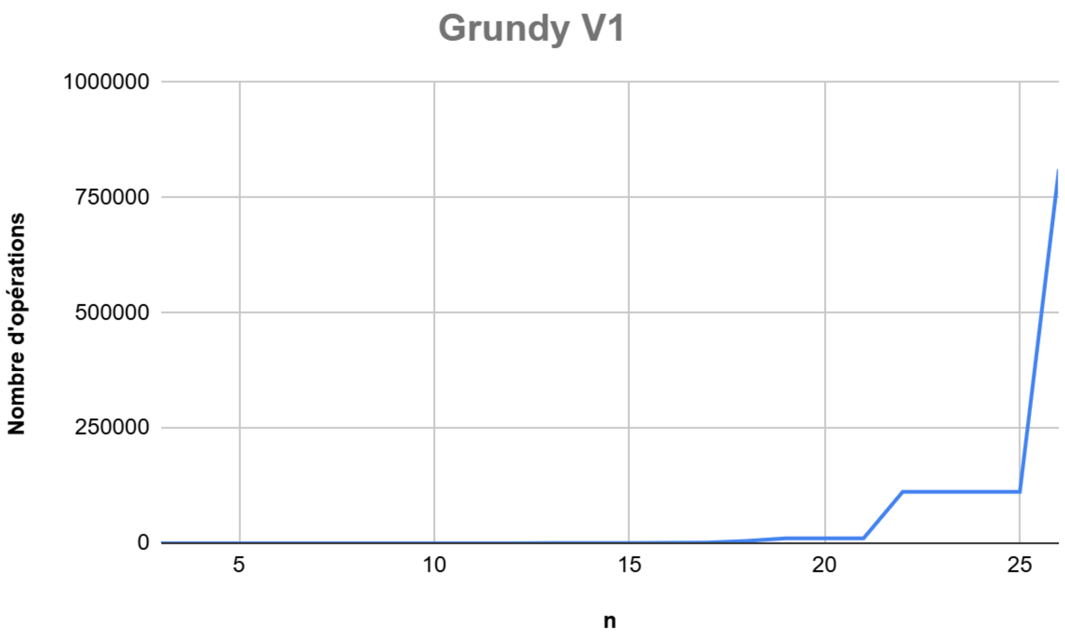


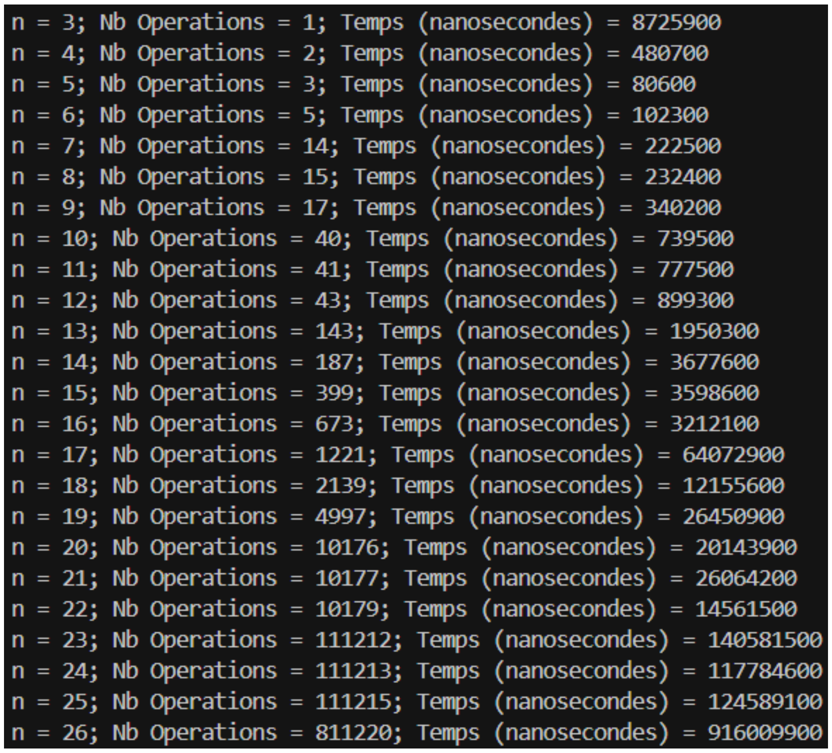
```
n = 3; Nb Operations = 1; Temps (nanosecondes) = 17800
n = 4; Nb Operations = 2; Temps (nanosecondes) = 23900
n = 5; Nb Operations = 3; Temps (nanosecondes) = 23600
n = 6; Nb Operations = 7; Temps (nanosecondes) = 58900
n = 7; Nb Operations = 18; Temps (nanosecondes) = 153800
n = 8; Nb Operations = 19; Temps (nanosecondes) = 160600
n = 9; Nb Operations = 39; Temps (nanosecondes) = 177700
n = 10; Nb Operations = 112; Temps (nanosecondes) = 327900
n = 11; Nb Operations = 113; Temps (nanosecondes) = 285900
n = 12; Nb Operations = 227; Temps (nanosecondes) = 536200
n = 13; Nb Operations = 1267; Temps (nanosecondes) = 2904600
n = 14; Nb Operations = 2559; Temps (nanosecondes) = 2830000
n = 15; Nb Operations = 6527; Temps (nanosecondes) = 4708900
n = 16; Nb Operations = 20141; Temps (nanosecondes) = 17510400
n = 17; Nb Operations = 62801; Temps (nanosecondes) = 48236800
n = 18; Nb Operations = 128101; Temps (nanosecondes) = 69310100
n = 19; Nb Operations = 422885; Temps (nanosecondes) = 274194900
n = 20; Nb Operations = 1433262; Temps (nanosecondes) = 298104800
n = 21; Nb Operations = 1433263; Temps (nanosecondes) = 259628100
n = 22; Nb Operations = 2866527; Temps (nanosecondes) = 535372100
```



Ce premier graphique et ce tableau mettent bien en évidence la faible efficacité de cette première version, qui perd en efficacité de manière exponentielle lorsque n grandit.

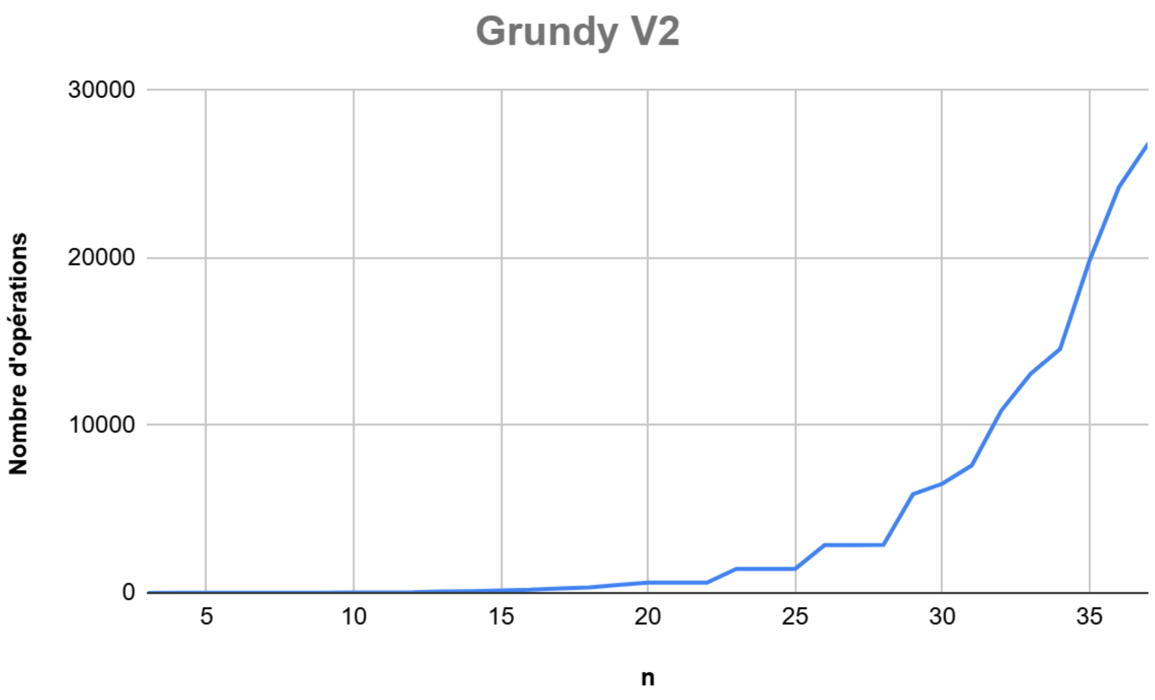
Le temps d'exécution devient vite très long.

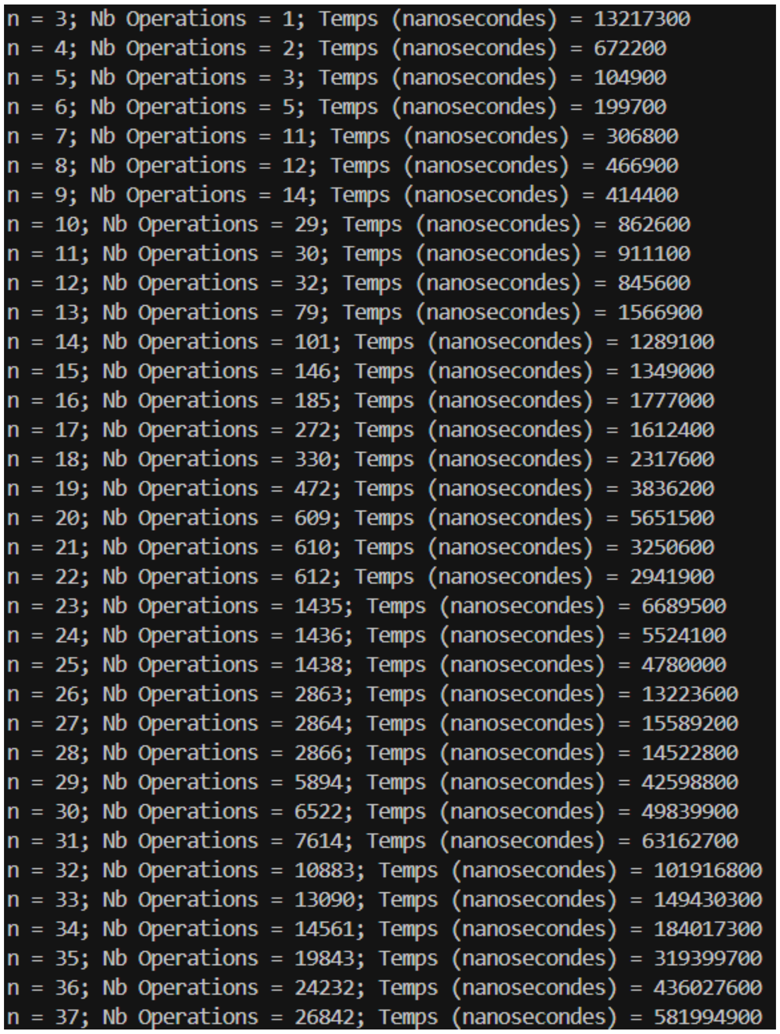




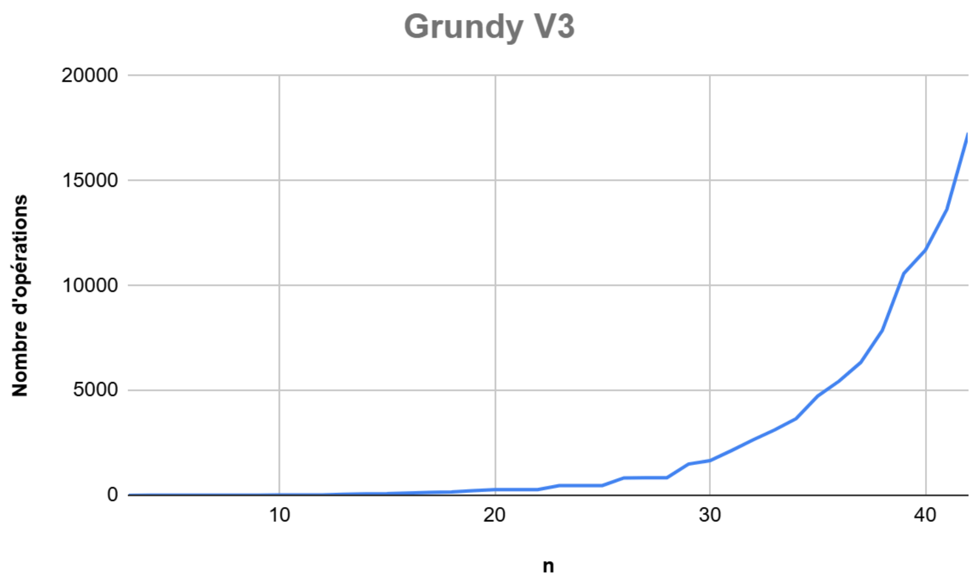
```
n = 3; Nb Operations = 1; Temps (nanosecondes) = 8725900
n = 4; Nb Operations = 2; Temps (nanosecondes) = 480700
n = 5; Nb Operations = 3; Temps (nanosecondes) = 80600
n = 6; Nb Operations = 5; Temps (nanosecondes) = 102300
n = 7; Nb Operations = 14; Temps (nanosecondes) = 222500
n = 8; Nb Operations = 15; Temps (nanosecondes) = 232400
n = 9; Nb Operations = 17; Temps (nanosecondes) = 340200
n = 10; Nb Operations = 40; Temps (nanosecondes) = 739500
n = 11; Nb Operations = 41; Temps (nanosecondes) = 777500
n = 12; Nb Operations = 43; Temps (nanosecondes) = 899300
n = 13; Nb Operations = 143; Temps (nanosecondes) = 1950300
n = 14; Nb Operations = 187; Temps (nanosecondes) = 3677600
n = 15; Nb Operations = 399; Temps (nanosecondes) = 3598600
n = 16; Nb Operations = 673; Temps (nanosecondes) = 3212100
n = 17; Nb Operations = 1221; Temps (nanosecondes) = 64072900
n = 18; Nb Operations = 2139; Temps (nanosecondes) = 12155600
n = 19; Nb Operations = 4997; Temps (nanosecondes) = 26450900
n = 20; Nb Operations = 10176; Temps (nanosecondes) = 20143900
n = 21; Nb Operations = 10177; Temps (nanosecondes) = 26064200
n = 22; Nb Operations = 10179; Temps (nanosecondes) = 14561500
n = 23; Nb Operations = 111212; Temps (nanosecondes) = 140581500
n = 24; Nb Operations = 111213; Temps (nanosecondes) = 117784600
n = 25; Nb Operations = 111215; Temps (nanosecondes) = 124589100
n = 26; Nb Operations = 811220; Temps (nanosecondes) = 916009900
```

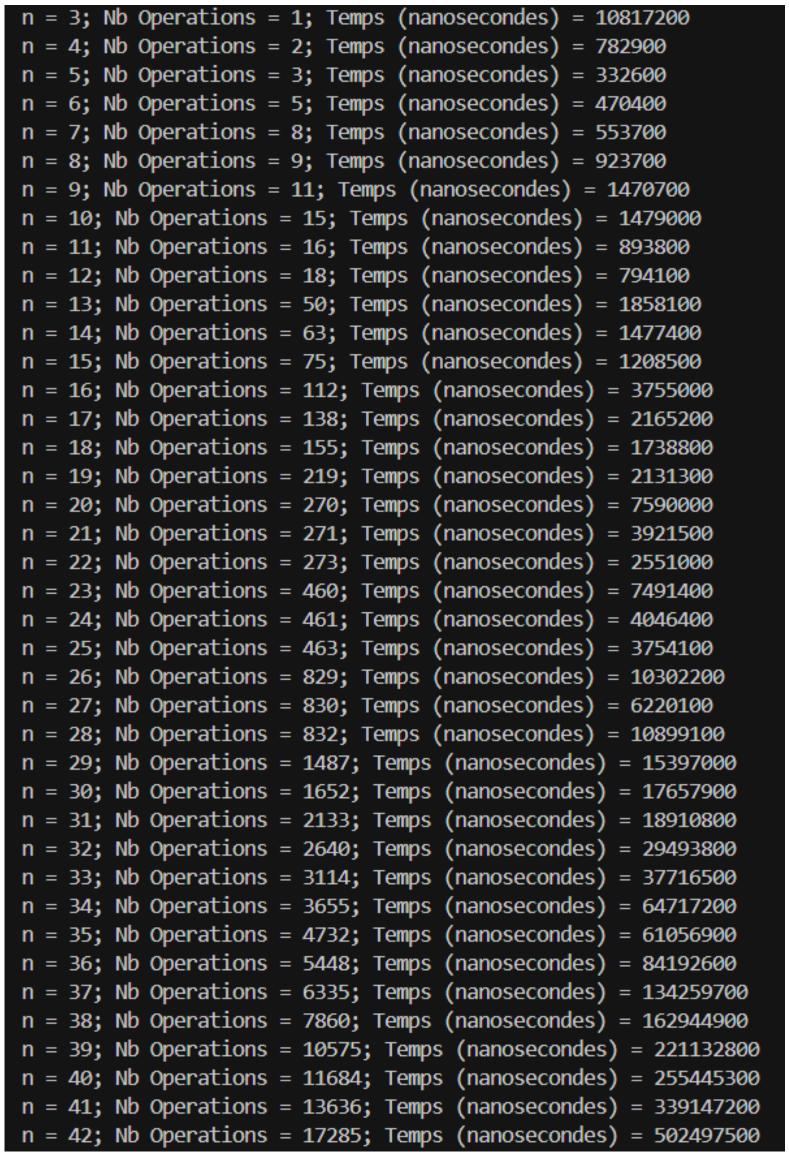
On voit grâce à ce tableau et ce 2e graphique, que la version 1 est nettement plus efficace (+de 280 fois moins d'opérations pour n=22) et plus rapide (+ de 36 fois moins long pour n=22) que la version 0.




```
n = 3; Nb Operations = 1; Temps (nanosecondes) = 13217300
n = 4; Nb Operations = 2; Temps (nanosecondes) = 672200
n = 5; Nb Operations = 3; Temps (nanosecondes) = 104900
n = 6; Nb Operations = 5; Temps (nanosecondes) = 199700
n = 7; Nb Operations = 11; Temps (nanosecondes) = 306800
n = 8; Nb Operations = 12; Temps (nanosecondes) = 466900
n = 9; Nb Operations = 14; Temps (nanosecondes) = 414400
n = 10; Nb Operations = 29; Temps (nanosecondes) = 862600
n = 11; Nb Operations = 30; Temps (nanosecondes) = 911100
n = 12; Nb Operations = 32; Temps (nanosecondes) = 845600
n = 13; Nb Operations = 79; Temps (nanosecondes) = 1566900
n = 14; Nb Operations = 101; Temps (nanosecondes) = 1289100
n = 15; Nb Operations = 146; Temps (nanosecondes) = 1349000
n = 16; Nb Operations = 185; Temps (nanosecondes) = 1777000
n = 17; Nb Operations = 272; Temps (nanosecondes) = 1612400
n = 18; Nb Operations = 330; Temps (nanosecondes) = 2317600
n = 19; Nb Operations = 472; Temps (nanosecondes) = 3836200
n = 20; Nb Operations = 609; Temps (nanosecondes) = 5651500
n = 21; Nb Operations = 610; Temps (nanosecondes) = 3250600
n = 22; Nb Operations = 612; Temps (nanosecondes) = 2941900
n = 23; Nb Operations = 1435; Temps (nanosecondes) = 6689500
n = 24; Nb Operations = 1436; Temps (nanosecondes) = 5524100
n = 25; Nb Operations = 1438; Temps (nanosecondes) = 4780000
n = 26; Nb Operations = 2863; Temps (nanosecondes) = 13223600
n = 27; Nb Operations = 2864; Temps (nanosecondes) = 15589200
n = 28; Nb Operations = 2866; Temps (nanosecondes) = 14522800
n = 29; Nb Operations = 5894; Temps (nanosecondes) = 42598800
n = 30; Nb Operations = 6522; Temps (nanosecondes) = 49839900
n = 31; Nb Operations = 7614; Temps (nanosecondes) = 63162700
n = 32; Nb Operations = 10883; Temps (nanosecondes) = 101916800
n = 33; Nb Operations = 13090; Temps (nanosecondes) = 149430300
n = 34; Nb Operations = 14561; Temps (nanosecondes) = 184017300
n = 35; Nb Operations = 19843; Temps (nanosecondes) = 319399700
n = 36; Nb Operations = 24232; Temps (nanosecondes) = 436027600
n = 37; Nb Operations = 26842; Temps (nanosecondes) = 581994900
```

Ce 3e tableau ainsi que ce 3e graphique nous permettent de voir que la version 2 est également bien plus efficace (là aussi + de 280 fois moins d'opérations pour n=26) et plus rapide (quasiment 70 fois moins long pour n=26) que la version 1 (et donc que la version 0 aussi).



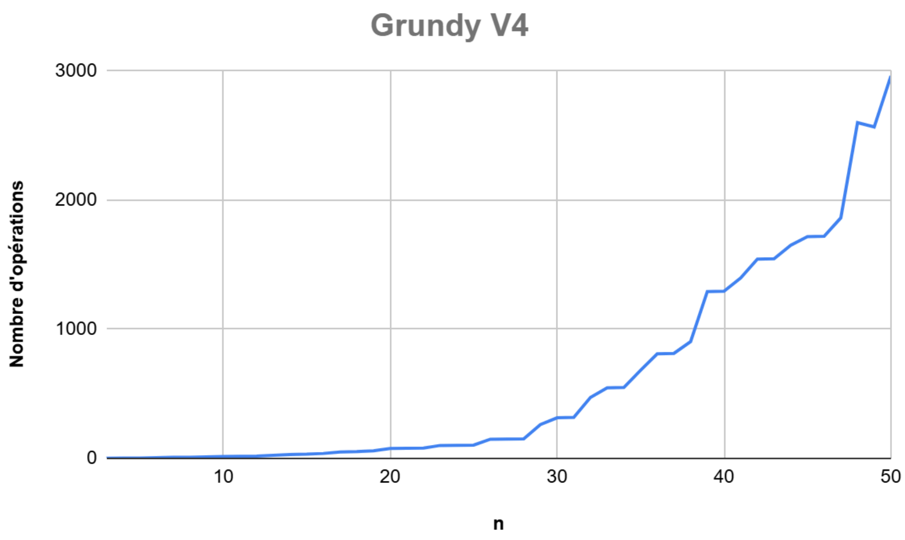


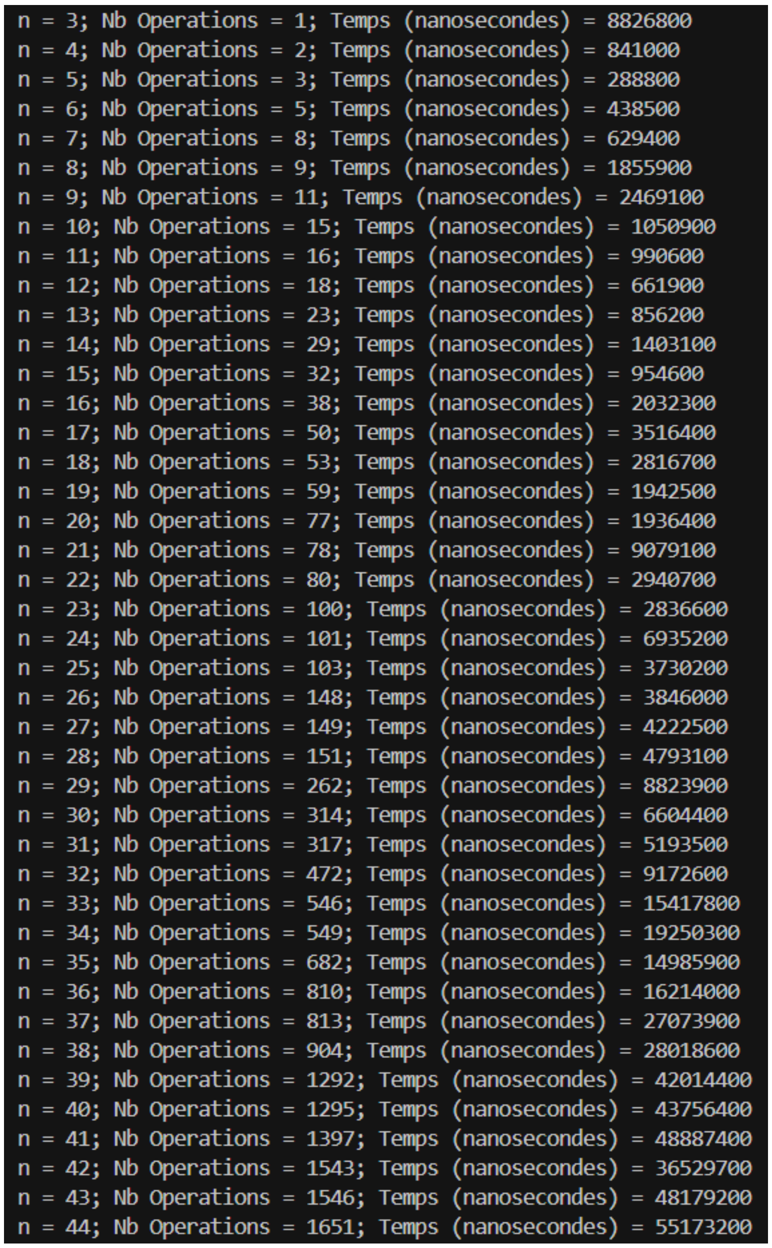
```
n = 3; Nb Operations = 1; Temps (nanosecondes) = 10817200
n = 4; Nb Operations = 2; Temps (nanosecondes) = 782900
n = 5; Nb Operations = 3; Temps (nanosecondes) = 332600
n = 6; Nb Operations = 5; Temps (nanosecondes) = 470400
n = 7; Nb Operations = 8; Temps (nanosecondes) = 553700
n = 8; Nb Operations = 9; Temps (nanosecondes) = 923700
n = 9; Nb Operations = 11; Temps (nanosecondes) = 1470700
n = 10; Nb Operations = 15; Temps (nanosecondes) = 1479000
n = 11; Nb Operations = 16; Temps (nanosecondes) = 893800
n = 12; Nb Operations = 18; Temps (nanosecondes) = 794100
n = 13; Nb Operations = 50; Temps (nanosecondes) = 1858100
n = 14; Nb Operations = 63; Temps (nanosecondes) = 1477400
n = 15; Nb Operations = 75; Temps (nanosecondes) = 1208500
n = 16; Nb Operations = 112; Temps (nanosecondes) = 3755000
n = 17; Nb Operations = 138; Temps (nanosecondes) = 2165200
n = 18; Nb Operations = 155; Temps (nanosecondes) = 1738800
n = 19; Nb Operations = 219; Temps (nanosecondes) = 2131300
n = 20; Nb Operations = 270; Temps (nanosecondes) = 7590000
n = 21; Nb Operations = 271; Temps (nanosecondes) = 3921500
n = 22; Nb Operations = 273; Temps (nanosecondes) = 2551000
n = 23; Nb Operations = 460; Temps (nanosecondes) = 7491400
n = 24; Nb Operations = 461; Temps (nanosecondes) = 4046400
n = 25; Nb Operations = 463; Temps (nanosecondes) = 3754100
n = 26; Nb Operations = 829; Temps (nanosecondes) = 10302200
n = 27; Nb Operations = 830; Temps (nanosecondes) = 6220100
n = 28; Nb Operations = 832; Temps (nanosecondes) = 10899100
n = 29; Nb Operations = 1487; Temps (nanosecondes) = 15397000
n = 30; Nb Operations = 1652; Temps (nanosecondes) = 17657900
n = 31; Nb Operations = 2133; Temps (nanosecondes) = 18910800
n = 32; Nb Operations = 2640; Temps (nanosecondes) = 29493800
n = 33; Nb Operations = 3114; Temps (nanosecondes) = 37716500
n = 34; Nb Operations = 3655; Temps (nanosecondes) = 64717200
n = 35; Nb Operations = 4732; Temps (nanosecondes) = 61056900
n = 36; Nb Operations = 5448; Temps (nanosecondes) = 84192600
n = 37; Nb Operations = 6335; Temps (nanosecondes) = 134259700
n = 38; Nb Operations = 7860; Temps (nanosecondes) = 162944900
n = 39; Nb Operations = 10575; Temps (nanosecondes) = 221132800
n = 40; Nb Operations = 11684; Temps (nanosecondes) = 255445300
n = 41; Nb Operations = 13636; Temps (nanosecondes) = 339147200
n = 42; Nb Operations = 17285; Temps (nanosecondes) = 502497500
```

D'après ce tableau et ce graphique, cette version 3 est encore plus optimisée :

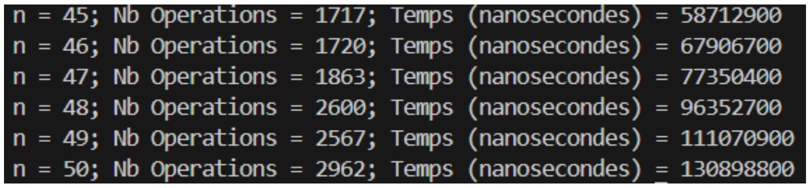
- +de 4 fois moins d'opérations
- +de 4 fois plus rapide

(pour n=37)





```
n = 3; Nb Operations = 1; Temps (nanosecondes) = 8826800
n = 4; Nb Operations = 2; Temps (nanosecondes) = 841000
n = 5; Nb Operations = 3; Temps (nanosecondes) = 288800
n = 6; Nb Operations = 5; Temps (nanosecondes) = 438500
n = 7; Nb Operations = 8; Temps (nanosecondes) = 629400
n = 8; Nb Operations = 9; Temps (nanosecondes) = 1855900
n = 9; Nb Operations = 11; Temps (nanosecondes) = 2469100
n = 10; Nb Operations = 15; Temps (nanosecondes) = 1050900
n = 11; Nb Operations = 16; Temps (nanosecondes) = 990600
n = 12; Nb Operations = 18; Temps (nanosecondes) = 661900
n = 13; Nb Operations = 23; Temps (nanosecondes) = 856200
n = 14; Nb Operations = 29; Temps (nanosecondes) = 1403100
n = 15; Nb Operations = 32; Temps (nanosecondes) = 954600
n = 16; Nb Operations = 38; Temps (nanosecondes) = 2032300
n = 17; Nb Operations = 50; Temps (nanosecondes) = 3516400
n = 18; Nb Operations = 53; Temps (nanosecondes) = 2816700
n = 19; Nb Operations = 59; Temps (nanosecondes) = 1942500
n = 20; Nb Operations = 77; Temps (nanosecondes) = 1936400
n = 21; Nb Operations = 78; Temps (nanosecondes) = 9079100
n = 22; Nb Operations = 80; Temps (nanosecondes) = 2940700
n = 23; Nb Operations = 100; Temps (nanosecondes) = 2836600
n = 24; Nb Operations = 101; Temps (nanosecondes) = 6935200
n = 25; Nb Operations = 103; Temps (nanosecondes) = 3730200
n = 26; Nb Operations = 148; Temps (nanosecondes) = 3846000
n = 27; Nb Operations = 149; Temps (nanosecondes) = 4222500
n = 28; Nb Operations = 151; Temps (nanosecondes) = 4793100
n = 29; Nb Operations = 262; Temps (nanosecondes) = 8823900
n = 30; Nb Operations = 314; Temps (nanosecondes) = 6604400
n = 31; Nb Operations = 317; Temps (nanosecondes) = 5193500
n = 32; Nb Operations = 472; Temps (nanosecondes) = 9172600
n = 33; Nb Operations = 546; Temps (nanosecondes) = 15417800
n = 34; Nb Operations = 549; Temps (nanosecondes) = 19250300
n = 35; Nb Operations = 682; Temps (nanosecondes) = 14985900
n = 36; Nb Operations = 810; Temps (nanosecondes) = 16214000
n = 37; Nb Operations = 813; Temps (nanosecondes) = 27073900
n = 38; Nb Operations = 904; Temps (nanosecondes) = 28018600
n = 39; Nb Operations = 1292; Temps (nanosecondes) = 42014400
n = 40; Nb Operations = 1295; Temps (nanosecondes) = 43756400
n = 41; Nb Operations = 1397; Temps (nanosecondes) = 48887400
n = 42; Nb Operations = 1543; Temps (nanosecondes) = 36529700
n = 43; Nb Operations = 1546; Temps (nanosecondes) = 48179200
n = 44; Nb Operations = 1651; Temps (nanosecondes) = 55173200
```



```
n = 45; Nb Operations = 1717; Temps (nanosecondes) = 58712900
n = 46; Nb Operations = 1720; Temps (nanosecondes) = 67906700
n = 47; Nb Operations = 1863; Temps (nanosecondes) = 77350400
n = 48; Nb Operations = 2600; Temps (nanosecondes) = 96352700
n = 49; Nb Operations = 2567; Temps (nanosecondes) = 111070900
n = 50; Nb Operations = 2962; Temps (nanosecondes) = 130898800
```

Cette version finale est la version la plus efficace est rapide :

- + de 11 fois moins d'opérations que la version 3 pour n = 42
- + de 13 fois plus rapide que la version 3 pour n = 42

Les tests auraient pu continuer plus loin sans problème pour cette méthode, mais pour cela il nous aurait fallu connaître les types des tas > à 50.

III. Conclusion

Dans ce projet, nous avons étudié plusieurs versions d’un algorithme pour résoudre le jeu de Grundy et améliorer ses performances :

Version 0 : La version de base était simple mais lente, car elle recalculait plusieurs fois les mêmes choses.
Version 1 : En mémorisant les positions perdantes, nous avons réduit les calculs inutiles, ce qui a rendu l'algorithme plus rapide.
Version 2 : En ajoutant aussi les positions gagnantes dans la mémorisation, l’algorithme est devenu encore plus efficace.
Version 3 : Avec le théorème des tas perdants, nous avons simplifié le raisonnement et encore amélioré les performances.
Les résultats montrent clairement que chaque optimisation a permis de gagner en efficacité et en rapidité, surtout pour les grandes valeurs.