ΕΘΝΙΚΟ ΚΑΠΟΔΙΣΤΡΙΑΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ
ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΤΗΛΕΠΙΚΟΙΝΩΝΙΩΝ

ΤΕΧΝΗΤΗ ΝΟΗΜΟΣΥΝΗ

Καθηγητής : ***Κουμπαράκης Μανόλης*** **Ημ/νία παράδοσης:** ***09/12/2011***

Ονομ/μο φοιτητή : ***Μπεγέτης Νικόλαος***
Α.Μ.: ***1115200700281***

Σχόλια και παρουσίαση αποτελεσμάτων για Πρόβλημα 5:

Όπως ζητείται στο αρχείο *Problem1_5.py* έχουμε λύσει το πρόβλημα των πύργων του Hanoi χρησιμοποιώντας τον αλγόριθμο *A** επαναορίζοντας σημεία του κώδικα του *AIMA* και υλοποιώντας την πρώτη από τις τέσσερις προτάσεις του Προβλήματος 5 "Ο αριθμός των δίσκων σε ένα στύλο διαφορετικό από τον δεξιότερο" ως ευρετική συνάρτηση.

Κατά την εκκίνηση του κυρίως προγράμματος ορίζουμε ένα αντικείμενο *hanoi* το οποίο κατά την αρχικοποίηση του λαμβάνει για είσοδο μία λίστα όπου κάθε στοιχείο της λίστας αντιστοιχεί σε ένα δίσκο ο οποίος μπορεί να παίρνει τιμές από 1 εως 3 ανάλογα σε ποιό στύλο βρίσκεται. Ο αριθμός των στύλων(peg-πάσσαλος) είναι 3 και είναι σταθερός. Ο αριθμός των δίσκων από την άλλη αφήνεται να επιλεγεί από τον χρήστη ώστε να δημιουργηθεί και η ανάλογη λίστα που περιγράψαμε παραπάνω. Κάνουμε δηλαδή ακριβώς ότι εξηγήσαμε και στο θεωρητικό κομμάτι της άσκησης. Οπότε για παράδειγμα αν ο χρήστης επιλέξει ότι θέλει 5 δίσκους κατά την αρχικοποίηση του προβλήματος θα δημιουργηθούν 2 *tuples* ένα για την αρχική θέση *(1,1,1,1,1)* και ένα για την τελική θέση-θέση στόχου *(3,3,3,3,3)* τα οποία θα έχουν μήκος όσο το μήκος της λίστας που δημιουργήθηκε.

Έπειτα αφού γίνει η αρχικοποίηση του βασικού αντικειμένου του προβλήματος ξεκινάει και χρονομετρείται η εύρεση βέλτιστης λύσης σύμφωνα με την παραδεκτή ευρεστική συνάρτηση που αναφέραμε παραπάνω χρησιμοποιώντας τον *A** αλγόριθμο. Έπειτα εκτυπώνονται τα αποτελέσματα:

- *συνολικοί κόμβοι που εξερευνήθηκαν,*
- *μονοπάτι καταστάσεων βέλτιστης λύσης και αλλαγές που πρέπει να γίνουν από την εκάστοτε τρέχουσα κατάσταση στην επόμενη κατάσταση*
- *κόστος λύσης*
- *συνολικός χρόνος εκτέλεσης αλγορίθμου.*

Στην κλάση *Hanoi* του αρχείου *problem1_5.py* που δέχεται ως όρισμα το πρόβλημα το οποίο χρησιμοποιεί ο αλγόριθμος A* επαναορίζονται οι συναρτήσεις *actions*, *result* και *path_cost* του αρχείου *search.py* .

Η συνάρτηση *actions* επιστρέφει τις έγκυρες-επιτρεπτές ενέργειες-κινήσεις που μπορούν να επιτευχθούν από κάθε θέση κόμβου. Αυτές οι ενέργειες είναι οι *(1,2), (1,3), (2,1), (2,3), (3,1)* και *(3,2)* και όπως εξηγήθηκε και στο θεωρητικό κομμάτι στο

πρώτο στοιχείο των παραπάνω δυάδων τοποθετούμε το στοιχείο που πρέπει να αλλάξει και στο δεύτερο το στοιχείο το οποίο θα αντικαταστήσει το πρώτο. Συγκεκριμένα για τους πύργους του Hanoi το πρώτο στοιχείο της 2άδας αφορά τον στύλο(*peg*) από τον οποίο θα μετακινήσουμε τον ελαφρύτερο δίσκο και το δεύτερο στοιχείο της δυάδας αφορά τον στύλο στον οποίο θα τοποθετήσουμε τον δίσκο αυτό. Οι απαραίτητες ενέργειες για να επιτευχθεί αυτό γίνονται στην επόμενη συνάρτηση *result*. Ο σκοπός της συνάρτησης *actions* που περιγράφουμε τώρα είναι απλά να επιστρέψει ένα σύνολο με τις επιτρεπτές ενέργειες από τις οποίες θα αποφασίσει ο αλγόριθμος ποια θα ακολουθήσει.

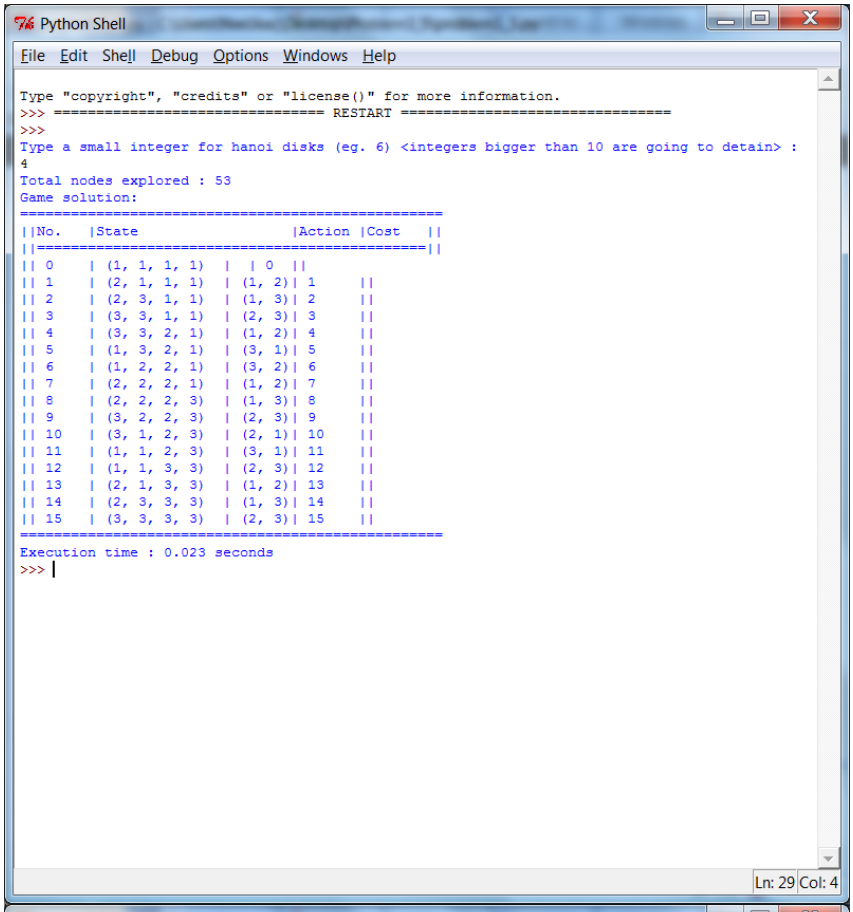
Η συνάρτηση *result* παίρνει την κατάσταση που επέλεξε ο αλγόριθμος από το σύνολο καταστάσεων που επέτρεψε η συνάρτηση *actions* και επιστρέφει κάνοντας τις απαραίτητες ενέργειες την επόμενη κατάσταση του προβλήματος των πύργων *Hanoi.*

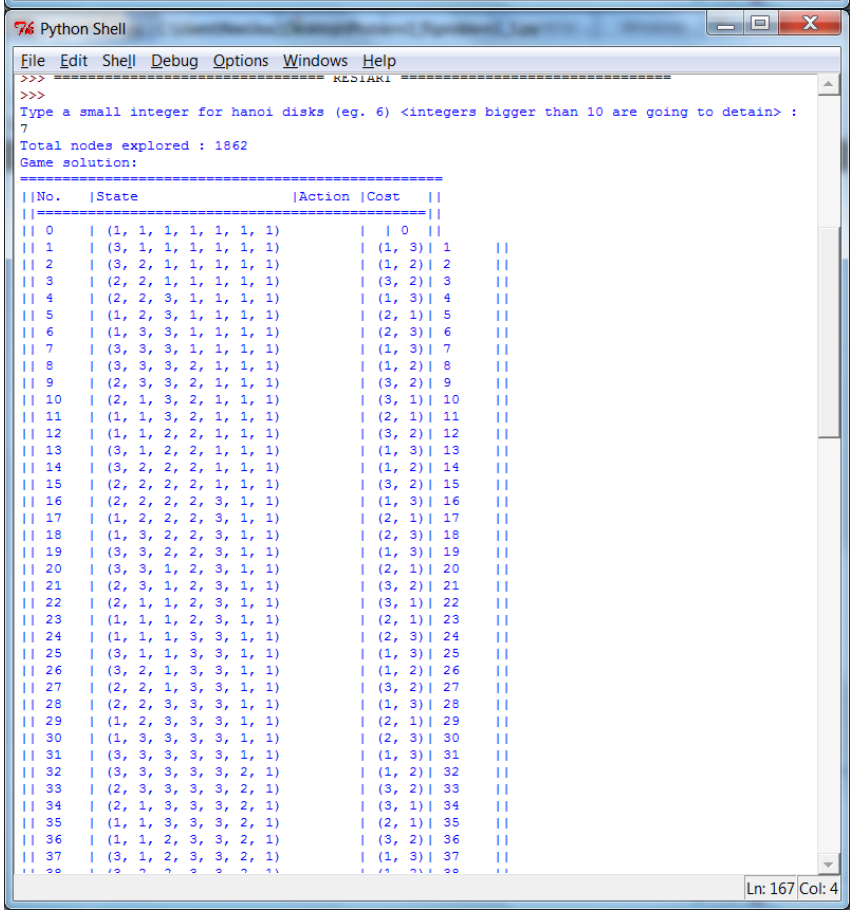
Η συνάρτηση *path_cost* επιστρέφει το κόστος ανάμεσα στην τρέχουσα και στην επόμενη θέση κόμβου που όπως έχουμε ορίσει αυξάνεται συνέχεια κατά ένα.

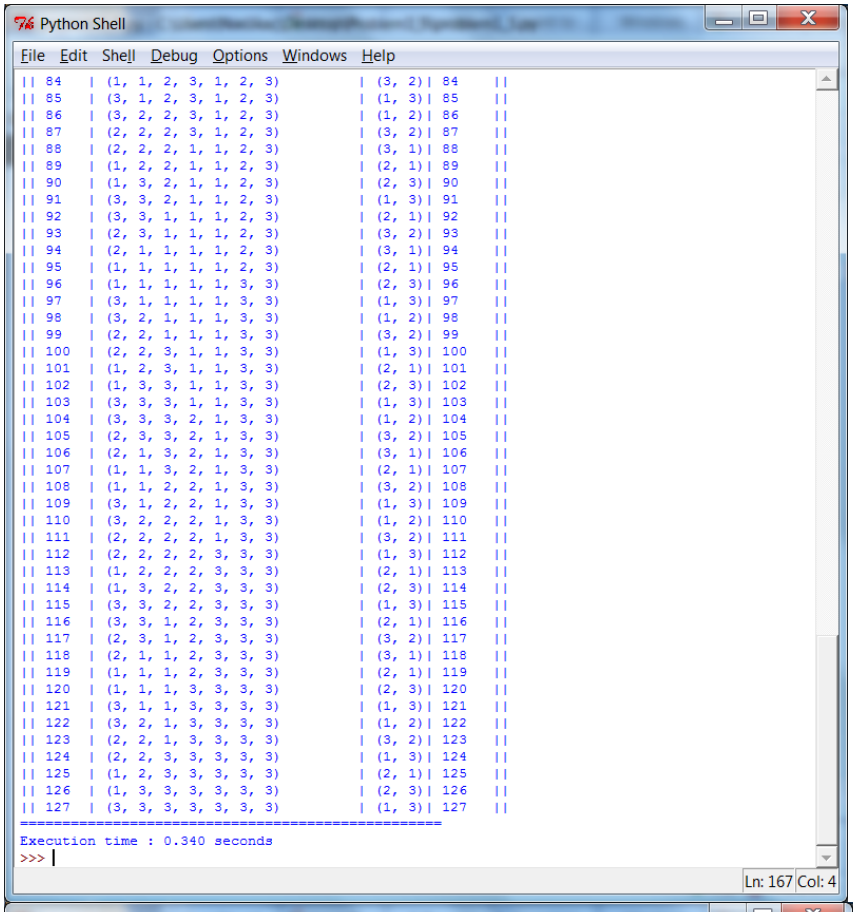
Στο θεωρητικό κομμάτι του προβλήματος 5 του πρώτου πακέτου ασκήσεων μας ζητήθηκε να ελέγξουμε ποιες ευρετικές συναρτήσεις είναι παραδεκτές και να χρησιμοποιήσουμε τη μία από αυτές. Ως παραδεκτές ευρετικές συναρτήσεις βρήκαμε την πρώτη και την τέταρτη και εδώ επιλέξαμε να υλοποιήσουμε την πρώτη όπου λέει "Ο αριθμός των δίσκων σε ένα στύλο διαφορετικό από τον δεξιότερο". Για την εξήγηση της ευρετικής συνάρτησης παρακαλούμε ανατρέξτε στη θεωρητική απάντηση που δώσαμε στο πρόβλημα ή και στα σχόλια που γράψαμε εντός του αρχείου κώδικα.

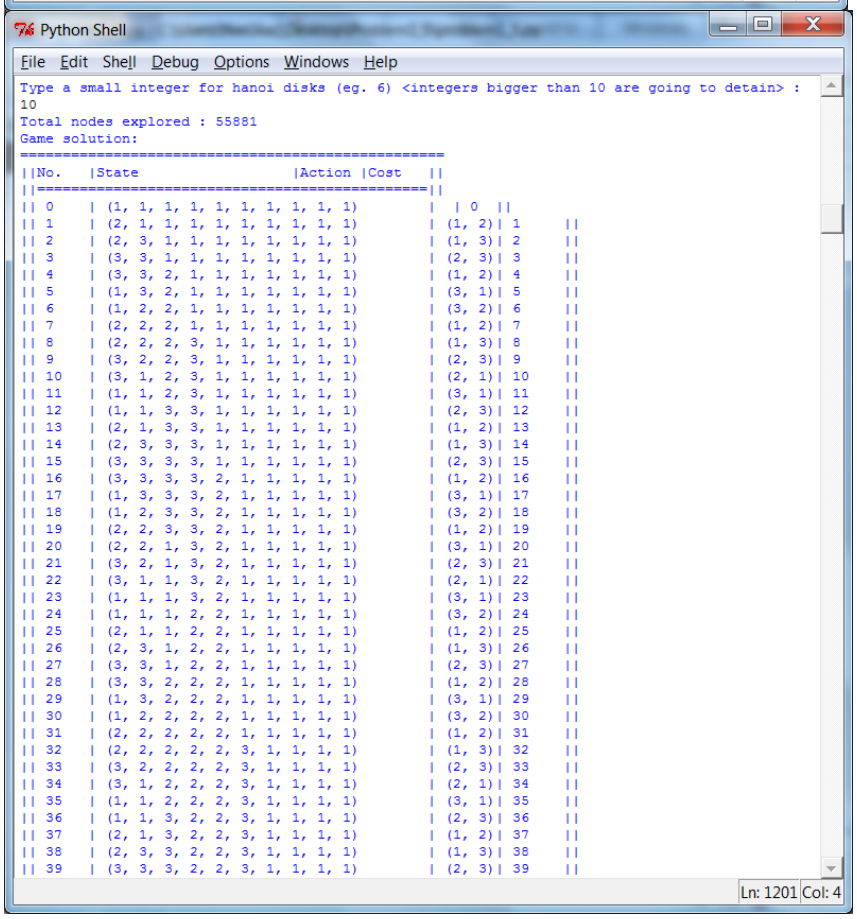
Η ευρεστική συνάρτηση με σύντομα λόγια ελέγχει το πλήθος των δίσκων που δεν βρίσκονται στον στύλο-τερματισμού(*goal peg*), δηλαδή συγκεκριμένα για αυτό το πρόβλημα ελέγχει το πλήθος των δίσκων που δεν είναι τοποθετημένοι στον τελευταίο στύλο.

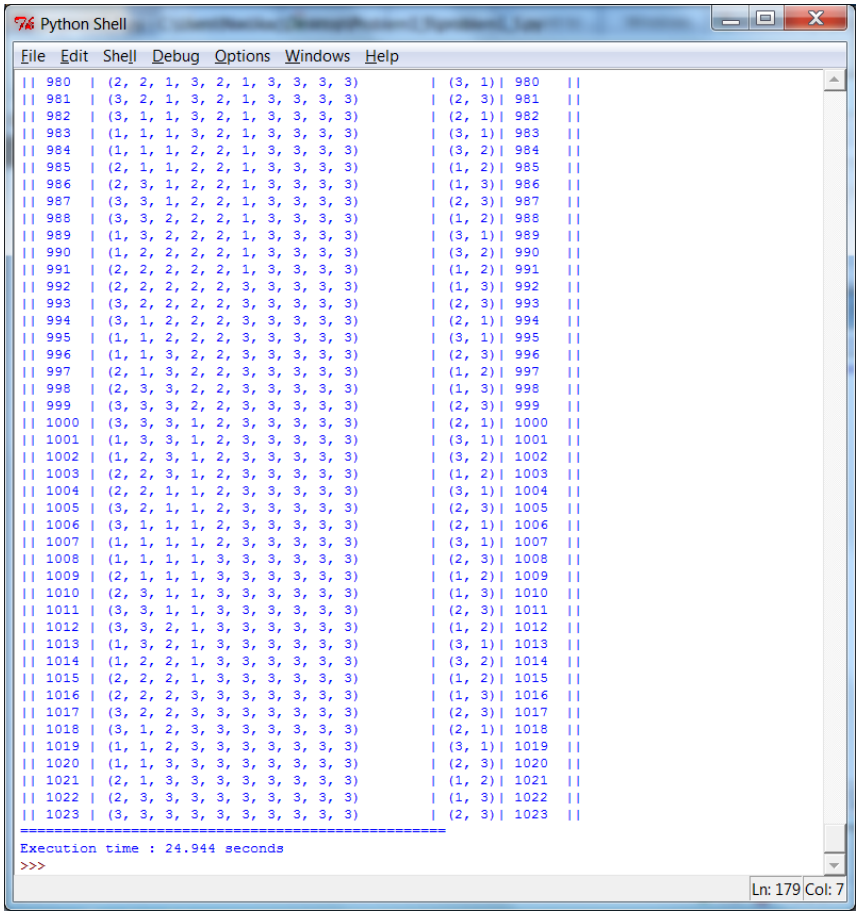
Ακολουθούν 3 ενδεικτικές εκτυπώσεις για *4, 7* και *10* στύλους:


Python Shell
File Edit Shell Debug Options Windows Help
Type "copyright", "credits" or "license()" for more information.
>>> ================================ RESTART ================================
>>>
Type a small integer for hanoi disks (eg. 6) <integers bigger than 10 are going to detain> :
4
Total nodes explored : 53
Game solution:
==
	No.	State	Action	Cost	
	==	
	0	(1, 1, 1, 1)		0	
	1	(2, 1, 1, 1)	(1, 2)	1	
	2	(2, 3, 1, 1)	(1, 3)	2	
	3	(3, 3, 1, 1)	(2, 3)	3	
	4	(3, 3, 2, 1)	(1, 2)	4	
	5	(1, 3, 2, 1)	(3, 1)	5	
	6	(1, 2, 2, 1)	(3, 2)	6	
	7	(2, 2, 2, 1)	(1, 2)	7	
	8	(2, 2, 2, 3)	(1, 3)	8	
	9	(3, 2, 2, 3)	(2, 3)	9	
	10	(3, 1, 2, 3)	(2, 1)	10	
	11	(1, 1, 2, 3)	(3, 1)	11	
	12	(1, 1, 3, 3)	(2, 3)	12	
	13	(2, 1, 3, 3)	(1, 2)	13	
	14	(2, 3, 3, 3)	(1, 3)	14	
	15	(3, 3, 3, 3)	(2, 3)	15	
==
Execution time : 0.023 seconds
>>>
Ln: 29 Col: 4


Python Shell
File Edit Shell Debug Options Windows Help
>>> ================================ RESTART ================================
>>>
Type a small integer for hanoi disks (eg. 6) <integers bigger than 10 are going to detain> :
7
Total nodes explored : 1862
Game solution:
==
	No.	State	Action	Cost	
	==	
	0	(1, 1, 1, 1, 1, 1, 1)		0	
	1	(3, 1, 1, 1, 1, 1, 1)	(1, 3)	1	
	2	(3, 2, 1, 1, 1, 1, 1)	(1, 2)	2	
	3	(2, 2, 1, 1, 1, 1, 1)	(3, 2)	3	
	4	(2, 2, 3, 1, 1, 1, 1)	(1, 3)	4	
	5	(1, 2, 3, 1, 1, 1, 1)	(2, 1)	5	
	6	(1, 3, 3, 1, 1, 1, 1)	(2, 3)	6	
	7	(3, 3, 3, 1, 1, 1, 1)	(1, 3)	7	
	8	(3, 3, 3, 2, 1, 1, 1)	(1, 2)	8	
	9	(2, 3, 3, 2, 1, 1, 1)	(3, 2)	9	
	10	(2, 1, 3, 2, 1, 1, 1)	(3, 1)	10	
	11	(1, 1, 3, 2, 1, 1, 1)	(2, 1)	11	
	12	(1, 1, 2, 2, 1, 1, 1)	(3, 2)	12	
	13	(3, 1, 2, 2, 1, 1, 1)	(1, 3)	13	
	14	(3, 2, 2, 2, 1, 1, 1)	(1, 2)	14	
	15	(2, 2, 2, 2, 1, 1, 1)	(3, 2)	15	
	16	(2, 2, 2, 2, 3, 1, 1)	(1, 3)	16	
	17	(1, 2, 2, 2, 3, 1, 1)	(2, 1)	17	
	18	(1, 3, 2, 2, 3, 1, 1)	(2, 3)	18	
	19	(3, 3, 2, 2, 3, 1, 1)	(1, 3)	19	
	20	(3, 3, 1, 2, 3, 1, 1)	(2, 1)	20	
	21	(2, 3, 1, 2, 3, 1, 1)	(3, 2)	21	
	22	(2, 1, 1, 2, 3, 1, 1)	(3, 1)	22	
	23	(1, 1, 1, 2, 3, 1, 1)	(2, 1)	23	
	24	(1, 1, 1, 3, 3, 1, 1)	(2, 3)	24	
	25	(3, 1, 1, 3, 3, 1, 1)	(1, 3)	25	
	26	(3, 2, 1, 3, 3, 1, 1)	(1, 2)	26	
	27	(2, 2, 1, 3, 3, 1, 1)	(3, 2)	27	
	28	(2, 2, 3, 3, 3, 1, 1)	(1, 3)	28	
	29	(1, 2, 3, 3, 3, 1, 1)	(2, 1)	29	
	30	(1, 3, 3, 3, 3, 1, 1)	(2, 3)	30	
	31	(3, 3, 3, 3, 3, 1, 1)	(1, 3)	31	
	32	(3, 3, 3, 3, 3, 2, 1)	(1, 2)	32	
	33	(2, 3, 3, 3, 3, 2, 1)	(3, 2)	33	
	34	(2, 1, 3, 3, 3, 2, 1)	(3, 1)	34	
	35	(1, 1, 3, 3, 3, 2, 1)	(2, 1)	35	
	36	(1, 1, 2, 3, 3, 2, 1)	(3, 2)	36	
	37	(3, 1, 2, 3, 3, 2, 1)	(1, 3)	37	
Ln: 167 Col: 4


Python Shell
File Edit Shell Debug Options Windows Help
	84	(1, 1, 2, 3, 1, 2, 3)	(3, 2)	84	
	85	(3, 1, 2, 3, 1, 2, 3)	(1, 3)	85	
	86	(3, 2, 2, 3, 1, 2, 3)	(1, 2)	86	
	87	(2, 2, 2, 3, 1, 2, 3)	(3, 2)	87	
	88	(2, 2, 2, 1, 1, 2, 3)	(3, 1)	88	
	89	(1, 2, 2, 1, 1, 2, 3)	(2, 1)	89	
	90	(1, 3, 2, 1, 1, 2, 3)	(2, 3)	90	
	91	(3, 3, 2, 1, 1, 2, 3)	(1, 3)	91	
	92	(3, 3, 1, 1, 1, 2, 3)	(2, 1)	92	
	93	(2, 3, 1, 1, 1, 2, 3)	(3, 2)	93	
	94	(2, 1, 1, 1, 1, 2, 3)	(3, 1)	94	
	95	(1, 1, 1, 1, 1, 2, 3)	(2, 1)	95	
	96	(1, 1, 1, 1, 1, 3, 3)	(2, 3)	96	
	97	(3, 1, 1, 1, 1, 3, 3)	(1, 3)	97	
	98	(3, 2, 1, 1, 1, 3, 3)	(1, 2)	98	
	99	(2, 2, 1, 1, 1, 3, 3)	(3, 2)	99	
	100	(2, 2, 3, 1, 1, 3, 3)	(1, 3)	100	
	101	(1, 2, 3, 1, 1, 3, 3)	(2, 1)	101	
	102	(1, 3, 3, 1, 1, 3, 3)	(2, 3)	102	
	103	(3, 3, 3, 1, 1, 3, 3)	(1, 3)	103	
	104	(3, 3, 3, 2, 1, 3, 3)	(1, 2)	104	
	105	(2, 3, 3, 2, 1, 3, 3)	(3, 2)	105	
	106	(2, 1, 3, 2, 1, 3, 3)	(3, 1)	106	
	107	(1, 1, 3, 2, 1, 3, 3)	(2, 1)	107	
	108	(1, 1, 2, 2, 1, 3, 3)	(3, 2)	108	
	109	(3, 1, 2, 2, 1, 3, 3)	(1, 3)	109	
	110	(3, 2, 2, 2, 1, 3, 3)	(1, 2)	110	
	111	(2, 2, 2, 2, 1, 3, 3)	(3, 2)	111	
	112	(2, 2, 2, 2, 3, 3, 3)	(1, 3)	112	
	113	(1, 2, 2, 2, 3, 3, 3)	(2, 1)	113	
	114	(1, 3, 2, 2, 3, 3, 3)	(2, 3)	114	
	115	(3, 3, 2, 2, 3, 3, 3)	(1, 3)	115	
	116	(3, 3, 1, 2, 3, 3, 3)	(2, 1)	116	
	117	(2, 3, 1, 2, 3, 3, 3)	(3, 2)	117	
	118	(2, 1, 1, 2, 3, 3, 3)	(3, 1)	118	
	119	(1, 1, 1, 2, 3, 3, 3)	(2, 1)	119	
	120	(1, 1, 1, 3, 3, 3, 3)	(2, 3)	120	
	121	(3, 1, 1, 3, 3, 3, 3)	(1, 3)	121	
	122	(3, 2, 1, 3, 3, 3, 3)	(1, 2)	122	
	123	(2, 2, 1, 3, 3, 3, 3)	(3, 2)	123	
	124	(2, 2, 3, 3, 3, 3, 3)	(1, 3)	124	
	125	(1, 2, 3, 3, 3, 3, 3)	(2, 1)	125	
	126	(1, 3, 3, 3, 3, 3, 3)	(2, 3)	126	
	127	(3, 3, 3, 3, 3, 3, 3)	(1, 3)	127	
Execution time : 0.340 seconds
>>>
Ln: 167 Col: 4


Python Shell
File Edit Shell Debug Options Windows Help
Type a small integer for hanoi disks (eg. 6) <integers bigger than 10 are going to detain> :
10
Total nodes explored : 55881
Game solution:
	No.	State	Action	Cost	
	0	(1, 1, 1, 1, 1, 1, 1, 1, 1, 1)		0	
	1	(2, 1, 1, 1, 1, 1, 1, 1, 1, 1)	(1, 2)	1	
	2	(2, 3, 1, 1, 1, 1, 1, 1, 1, 1)	(1, 3)	2	
	3	(3, 3, 1, 1, 1, 1, 1, 1, 1, 1)	(2, 3)	3	
	4	(3, 3, 2, 1, 1, 1, 1, 1, 1, 1)	(1, 2)	4	
	5	(1, 3, 2, 1, 1, 1, 1, 1, 1, 1)	(3, 1)	5	
	6	(1, 2, 2, 1, 1, 1, 1, 1, 1, 1)	(3, 2)	6	
	7	(2, 2, 2, 1, 1, 1, 1, 1, 1, 1)	(1, 2)	7	
	8	(2, 2, 2, 3, 1, 1, 1, 1, 1, 1)	(1, 3)	8	
	9	(3, 2, 2, 3, 1, 1, 1, 1, 1, 1)	(2, 3)	9	
	10	(3, 1, 2, 3, 1, 1, 1, 1, 1, 1)	(2, 1)	10	
	11	(1, 1, 2, 3, 1, 1, 1, 1, 1, 1)	(3, 1)	11	
	12	(1, 1, 3, 3, 1, 1, 1, 1, 1, 1)	(2, 3)	12	
	13	(2, 1, 3, 3, 1, 1, 1, 1, 1, 1)	(1, 2)	13	
	14	(2, 3, 3, 3, 1, 1, 1, 1, 1, 1)	(1, 3)	14	
	15	(3, 3, 3, 3, 1, 1, 1, 1, 1, 1)	(2, 3)	15	
	16	(3, 3, 3, 3, 2, 1, 1, 1, 1, 1)	(1, 2)	16	
	17	(1, 3, 3, 3, 2, 1, 1, 1, 1, 1)	(3, 1)	17	
	18	(1, 2, 3, 3, 2, 1, 1, 1, 1, 1)	(3, 2)	18	
	19	(2, 2, 3, 3, 2, 1, 1, 1, 1, 1)	(1, 2)	19	
	20	(2, 2, 1, 3, 2, 1, 1, 1, 1, 1)	(3, 1)	20	
	21	(3, 2, 1, 3, 2, 1, 1, 1, 1, 1)	(2, 3)	21	
	22	(3, 1, 1, 3, 2, 1, 1, 1, 1, 1)	(2, 1)	22	
	23	(1, 1, 1, 3, 2, 1, 1, 1, 1, 1)	(3, 1)	23	
	24	(1, 1, 1, 2, 2, 1, 1, 1, 1, 1)	(3, 2)	24	
	25	(2, 1, 1, 2, 2, 1, 1, 1, 1, 1)	(1, 2)	25	
	26	(2, 3, 1, 2, 2, 1, 1, 1, 1, 1)	(1, 3)	26	
	27	(3, 3, 1, 2, 2, 1, 1, 1, 1, 1)	(2, 3)	27	
	28	(3, 3, 2, 2, 2, 1, 1, 1, 1, 1)	(1, 2)	28	
	29	(1, 3, 2, 2, 2, 1, 1, 1, 1, 1)	(3, 1)	29	
	30	(1, 2, 2, 2, 2, 1, 1, 1, 1, 1)	(3, 2)	30	
	31	(2, 2, 2, 2, 2, 1, 1, 1, 1, 1)	(1, 2)	31	
	32	(2, 2, 2, 2, 2, 3, 1, 1, 1, 1)	(1, 3)	32	
	33	(3, 2, 2, 2, 2, 3, 1, 1, 1, 1)	(2, 3)	33	
	34	(3, 1, 2, 2, 2, 3, 1, 1, 1, 1)	(2, 1)	34	
	35	(1, 1, 2, 2, 2, 3, 1, 1, 1, 1)	(3, 1)	35	
	36	(1, 1, 3, 2, 2, 3, 1, 1, 1, 1)	(2, 3)	36	
	37	(2, 1, 3, 2, 2, 3, 1, 1, 1, 1)	(1, 2)	37	
	38	(2, 3, 3, 2, 2, 3, 1, 1, 1, 1)	(1, 3)	38	
	39	(3, 3, 3, 2, 2, 3, 1, 1, 1, 1)	(2, 3)	39	
Ln: 1201 Col: 4



Όπως φαίνεται από τις παραπάνω 5 εκτυπώσεις για *δίσκους = 4, 7, 10* η ευρετική συνάρτηση είναι βέλτιστη, γιατί όπως αποδείξαμε στο θεωρητικό κομμάτι το βέλτιστο κόστος λύσης για το πρόβλημα των πύργων του *Hanoi* είναι 2^k*-1*, όπου *k* είναι ο αριθμός των δίσκων. Πράγματι παρατηρούμε ότι για *4* δίσκους το κόστος είναι *15*, δηλαδή 2^4*-1*, για *7* δίσκους είναι *127*, δηλαδή 2^7*-1* και τέλος για *10* δίσκους είναι *1023*, δηλαδή 2^{10}*-1*.

Επίσης αξίζει να τονιστεί η τεράστια κλιμάκωση τους κόμβους-καταστάσεις που επισκέφτηκε ο αλγόριθμος(53, 1862 και 55881 αντίστοιχα) για να βρει την βέλτιστη λύση και η συνέπειά αυτού στον τελικό χρόνο(0.023, 0.34 και 24.944 sec αντίστοιχα)