A logo of a book and tree

Description automatically generated

Τεχνητή Νοημοσύνη

2η άσκηση

*Δεμερτζόγλου Ευστράτιος | ΤΗ20580*

Table of Contents

[Περιγραφή του Προβλήματος 3](#_Toc194845276)

[Προσέγγιση Επίλυσης 3](#_Toc194845277)

[Επεξήση Κώδικα 4](#_Toc194845278)

[Παράδειγμα Εκτέλεσης 4](#_Toc194845279)

[BFS run 4](#_Toc194845280)

[DFS run 4](#_Toc194845281)

[Κώδικας σε Python που εκτελέστηκε 4](#_Toc194845282)

[Συμπέρασμα 7](#_Toc194845283)

# Περιγραφή του Προβλήματος

Το πρόβλημα αφορά τη στοίβαξη τριών κουτιών a, b, c πάνω σε ένα τραπέζι t. Δίνεται μία **αρχική κατάσταση** και επιθυμούμε να φτάσουμε σε μία **τελική κατάσταση** χρησιμοποιώντας έναν και μόνο τελεστή:  
move(X,Y)  
που σημαίνει "μετακίνησε το αντικείμενο X πάνω στο Y".

Ο τελεστής έχει τις εξής **προϋποθέσεις**:

* Το X να μην έχει άλλο αντικείμενο από πάνω του.
* Το Y, αν δεν είναι το τραπέζι t, να μην έχει επίσης άλλο αντικείμενο από πάνω του.
* Η νέα κατάσταση να μην έχει ήδη επισκεφθεί ξανά στο παρελθόν (για αποφυγή βρόχων).

# Προσέγγιση Επίλυσης

Το πρόβλημα επιλύεται με αναζήτηση στο **χώρο καταστάσεων**, χρησιμοποιώντας δύο κλασικούς αλγορίθμους τεχνητής νοημοσύνης:

* **Αναζήτηση Πρώτα κατά Πλάτος (BFS)**: Εξετάζει όλες τις πιθανές κινήσεις επιπέδου πριν προχωρήσει βαθύτερα.
* **Αναζήτηση Πρώτα κατά Βάθος (DFS)**: Εξερευνά πρώτα τις πιο πρόσφατες διαδρομές και επιστρέφει πίσω όταν χρειάζεται.

# Επεξήση Κώδικα

Κάθε κατάσταση αναπαρίσταται ως ένα λεξικό που δείχνει που βρίσκεται κάθε block

Π.χ.:

initial\_state = {'a': 'c', 'b': 't', 'c': 't'}

Το a βρίσκεται πάνω στο c, το b βρίσκεται πάνω στο t και το c βρίσκεται πάνω στο t

**Βασικές Συναρτήσεις:**

* has\_on\_top(state, x): Ελέγχει αν υπάρχει άλλο μπλοκ πάνω στο x.
* successors(state): Επιστρέφει όλες τις έγκυρες μετακινήσεις από την παρούσα κατάσταση.
* is\_goal(state): Ελέγχει αν η κατάσταση είναι η τελική.
* bfs(start): Υλοποίηση του BFS για εύρεση της συντομότερης λύσης.
* dfs(start): Υλοποίηση DFS για εύρεση κάποιας λύσης.

# Παράδειγμα Εκτέλεσης

Αυτό το παράδειγμα είναι drop down menu γιατί έχει πολύ μεγάλη έκταση

## BFS run

BFS Solution:

Εξετάζουμε νέα κατάσταση:

Κατάσταση:

a πάνω σε c

b πάνω σε t

c πάνω σε t

Εκτελώ: move(a,b)

Κατάσταση:

a πάνω σε b

b πάνω σε t

c πάνω σε t

Εκτελώ: move(a,t)

Κατάσταση:

a πάνω σε t

b πάνω σε t

c πάνω σε t

Εκτελώ: move(b,a)

Κατάσταση:

a πάνω σε c

b πάνω σε a

c πάνω σε t

Εξετάζουμε νέα κατάσταση:

Κατάσταση:

a πάνω σε b

b πάνω σε t

c πάνω σε t

Εκτελώ: move(a,c)

Κατάσταση:

a πάνω σε c

b πάνω σε t

c πάνω σε t

Εκτελώ: move(a,t)

Κατάσταση:

a πάνω σε t

b πάνω σε t

c πάνω σε t

Εκτελώ: move(c,a)

Κατάσταση:

a πάνω σε b

b πάνω σε t

c πάνω σε a

Εξετάζουμε νέα κατάσταση:

Κατάσταση:

a πάνω σε t

b πάνω σε t

c πάνω σε t

Εκτελώ: move(a,b)

Κατάσταση:

a πάνω σε b

b πάνω σε t

c πάνω σε t

Εκτελώ: move(a,c)

Κατάσταση:

a πάνω σε c

b πάνω σε t

c πάνω σε t

Εκτελώ: move(b,a)

Κατάσταση:

a πάνω σε t

b πάνω σε a

c πάνω σε t

Εκτελώ: move(b,c)

Κατάσταση:

a πάνω σε t

b πάνω σε c

c πάνω σε t

Εκτελώ: move(c,a)

Κατάσταση:

a πάνω σε t

b πάνω σε t

c πάνω σε a

Εκτελώ: move(c,b)

Κατάσταση:

a πάνω σε t

b πάνω σε t

c πάνω σε b

Εξετάζουμε νέα κατάσταση:

Κατάσταση:

a πάνω σε c

b πάνω σε a

c πάνω σε t

Εκτελώ: move(b,t)

Κατάσταση:

a πάνω σε c

b πάνω σε t

c πάνω σε t

Εξετάζουμε νέα κατάσταση:

Κατάσταση:

a πάνω σε b

b πάνω σε t

c πάνω σε a

Εκτελώ: move(c,t)

Κατάσταση:

a πάνω σε b

b πάνω σε t

c πάνω σε t

Εξετάζουμε νέα κατάσταση:

Κατάσταση:

a πάνω σε t

b πάνω σε a

c πάνω σε t

Εκτελώ: move(b,c)

Κατάσταση:

a πάνω σε t

b πάνω σε c

c πάνω σε t

Εκτελώ: move(b,t)

Κατάσταση:

a πάνω σε t

b πάνω σε t

c πάνω σε t

Εκτελώ: move(c,b)

Κατάσταση:

a πάνω σε t

b πάνω σε a

c πάνω σε b

Εξετάζουμε νέα κατάσταση:

Κατάσταση:

a πάνω σε t

b πάνω σε c

c πάνω σε t

Εκτελώ: move(a,b)

Κατάσταση:

a πάνω σε b

b πάνω σε c

c πάνω σε t

Εκτελώ: move(b,a)

Κατάσταση:

a πάνω σε t

b πάνω σε a

c πάνω σε t

Εκτελώ: move(b,t)

Κατάσταση:

a πάνω σε t

b πάνω σε t

c πάνω σε t

Εξετάζουμε νέα κατάσταση:

Κατάσταση:

a πάνω σε t

b πάνω σε t

c πάνω σε a

Εκτελώ: move(b,c)

Κατάσταση:

a πάνω σε t

b πάνω σε c

c πάνω σε a

Εκτελώ: move(c,b)

Κατάσταση:

a πάνω σε t

b πάνω σε t

c πάνω σε b

Εκτελώ: move(c,t)

Κατάσταση:

a πάνω σε t

b πάνω σε t

c πάνω σε t

Εξετάζουμε νέα κατάσταση:

Κατάσταση:

a πάνω σε t

b πάνω σε t

c πάνω σε b

Εκτελώ: move(a,c)

Κατάσταση:

a πάνω σε c

b πάνω σε t

c πάνω σε b

Εκτελώ: move(c,a)

Κατάσταση:

a πάνω σε t

b πάνω σε t

c πάνω σε a

Εκτελώ: move(c,t)

Κατάσταση:

a πάνω σε t

b πάνω σε t

c πάνω σε t

Εξετάζουμε νέα κατάσταση:

Κατάσταση:

a πάνω σε t

b πάνω σε a

c πάνω σε b

Εκτελώ: move(c,t)

Κατάσταση:

a πάνω σε t

b πάνω σε a

c πάνω σε t

--- Τελική κατάσταση επιτεύχθηκε (BFS) ---

Κατάσταση:

a πάνω σε b

b πάνω σε c

c πάνω σε t

['move(a,t)', 'move(b,c)', 'move(a,b)']

## DFS run

DFS Solution:

Εξετάζουμε νέα κατάσταση:

Κατάσταση:

a πάνω σε c

b πάνω σε t

c πάνω σε t

Εκτελώ: move(a,b)

Κατάσταση:

a πάνω σε b

b πάνω σε t

c πάνω σε t

Εκτελώ: move(a,t)

Κατάσταση:

a πάνω σε t

b πάνω σε t

c πάνω σε t

Εκτελώ: move(b,a)

Κατάσταση:

a πάνω σε c

b πάνω σε a

c πάνω σε t

Εξετάζουμε νέα κατάσταση:

Κατάσταση:

a πάνω σε c

b πάνω σε a

c πάνω σε t

Εκτελώ: move(b,t)

Κατάσταση:

a πάνω σε c

b πάνω σε t

c πάνω σε t

Εξετάζουμε νέα κατάσταση:

Κατάσταση:

a πάνω σε t

b πάνω σε t

c πάνω σε t

Εκτελώ: move(a,b)

Κατάσταση:

a πάνω σε b

b πάνω σε t

c πάνω σε t

Εκτελώ: move(a,c)

Κατάσταση:

a πάνω σε c

b πάνω σε t

c πάνω σε t

Εκτελώ: move(b,a)

Κατάσταση:

a πάνω σε t

b πάνω σε a

c πάνω σε t

Εκτελώ: move(b,c)

Κατάσταση:

a πάνω σε t

b πάνω σε c

c πάνω σε t

Εκτελώ: move(c,a)

Κατάσταση:

a πάνω σε t

b πάνω σε t

c πάνω σε a

Εκτελώ: move(c,b)

Κατάσταση:

a πάνω σε t

b πάνω σε t

c πάνω σε b

Εξετάζουμε νέα κατάσταση:

Κατάσταση:

a πάνω σε t

b πάνω σε t

c πάνω σε b

Εκτελώ: move(a,c)

Κατάσταση:

a πάνω σε c

b πάνω σε t

c πάνω σε b

Εκτελώ: move(c,a)

Κατάσταση:

a πάνω σε t

b πάνω σε t

c πάνω σε a

Εκτελώ: move(c,t)

Κατάσταση:

a πάνω σε t

b πάνω σε t

c πάνω σε t

Εξετάζουμε νέα κατάσταση:

Κατάσταση:

a πάνω σε t

b πάνω σε t

c πάνω σε a

Εκτελώ: move(b,c)

Κατάσταση:

a πάνω σε t

b πάνω σε c

c πάνω σε a

Εκτελώ: move(c,b)

Κατάσταση:

a πάνω σε t

b πάνω σε t

c πάνω σε b

Εκτελώ: move(c,t)

Κατάσταση:

a πάνω σε t

b πάνω σε t

c πάνω σε t

Εξετάζουμε νέα κατάσταση:

Κατάσταση:

a πάνω σε t

b πάνω σε c

c πάνω σε a

Εκτελώ: move(b,t)

Κατάσταση:

a πάνω σε t

b πάνω σε t

c πάνω σε a

Εξετάζουμε νέα κατάσταση:

Κατάσταση:

a πάνω σε c

b πάνω σε t

c πάνω σε b

Εκτελώ: move(a,t)

Κατάσταση:

a πάνω σε t

b πάνω σε t

c πάνω σε b

Εξετάζουμε νέα κατάσταση:

Κατάσταση:

a πάνω σε t

b πάνω σε c

c πάνω σε t

Εκτελώ: move(a,b)

Κατάσταση:

a πάνω σε b

b πάνω σε c

c πάνω σε t

Εκτελώ: move(b,a)

Κατάσταση:

a πάνω σε t

b πάνω σε a

c πάνω σε t

Εκτελώ: move(b,t)

Κατάσταση:

a πάνω σε t

b πάνω σε t

c πάνω σε t

Εξετάζουμε νέα κατάσταση:

Κατάσταση:

a πάνω σε t

b πάνω σε a

c πάνω σε t

Εκτελώ: move(b,c)

Κατάσταση:

a πάνω σε t

b πάνω σε c

c πάνω σε t

Εκτελώ: move(b,t)

Κατάσταση:

a πάνω σε t

b πάνω σε t

c πάνω σε t

Εκτελώ: move(c,b)

Κατάσταση:

a πάνω σε t

b πάνω σε a

c πάνω σε b

Εξετάζουμε νέα κατάσταση:

Κατάσταση:

a πάνω σε t

b πάνω σε a

c πάνω σε b

Εκτελώ: move(c,t)

Κατάσταση:

a πάνω σε t

b πάνω σε a

c πάνω σε t

--- Τελική κατάσταση επιτεύχθηκε (DFS) ---

Κατάσταση:

a πάνω σε b

b πάνω σε c

c πάνω σε t

['move(a,t)', 'move(b,c)', 'move(a,b)']

# Κώδικας σε Python που εκτελέστηκε

from collections import deque

import copy

# Ελέγχει αν ένα αντικείμενο έχει κάτι πάνω του

def has\_on\_top(state, x):

    return any(v == x for v in state.values())

# Παράγει όλες τις επόμενες έγκυρες καταστάσεις από την τρέχουσα

def successors(state):

    result = []

    blocks = ['a', 'b', 'c']

    support = ['a', 'b', 'c', 't']  # Πάνω σε ποιο αντικείμενο μπορεί να μπει ένα μπλοκ

    for x in blocks:

        if has\_on\_top(state, x):

            continue  # Αν το X έχει κάτι πάνω του, δεν μπορούμε να το μετακινήσουμε

        for y in support:

            if x == y:

                continue  # Δεν επιτρέπεται να βάλουμε ένα μπλοκ πάνω στον εαυτό του

            if y != 't' and has\_on\_top(state, y):

                continue  # Αν το Y είναι άλλο μπλοκ και έχει κάτι πάνω του, δεν επιτρέπεται

            new\_state = copy.deepcopy(state)

            new\_state[x] = y  # Μετακινούμε το X πάνω στο Y

            if new\_state != state:

                result.append((f"move({x},{y})", new\_state))  # Προσθέτουμε νέο βήμα και κατάσταση

    return result

# Ελέγχει αν η κατάσταση είναι η τελική

def is\_goal(state):

    return state == {'a': 'b', 'b': 'c', 'c': 't'}

# Εκτυπώνει την κατάσταση

def print\_state(state):

    print("Κατάσταση:")

    for block in sorted(state.keys()):

        print(f"  {block} πάνω σε {state[block]}")

    print()

# BFS αναζήτηση

def bfs(start):

    visited = set()

    queue = deque([(start, [])])  # (τρέχουσα κατάσταση, λίστα κινήσεων)

    while queue:

        state, path = queue.popleft()

        state\_tuple = tuple(sorted(state.items()))

        if state\_tuple in visited:

            continue

        visited.add(state\_tuple)

        if is\_goal(state):

            print("\n--- Τελική κατάσταση επιτεύχθηκε (BFS) ---\n")

            print\_state(state)

            return path # Επιστροφή των βημάτων που οδηγήσαν στην τελική κατάσταση

        print("Εξετάζουμε νέα κατάσταση:")

        print\_state(state)

        for action, new\_state in successors(state):

            print(f"Εκτελώ: {action}")

            print\_state(new\_state)

            queue.append((new\_state, path + [action]))

    return None # Δεν βρέθηκε λύση

# DFS αναζήτηση

def dfs(start):

    visited = set()

    stack = [(start, [])]  # (τρέχουσα κατάσταση, λίστα κινήσεων)

    while stack:

        state, path = stack.pop()

        state\_tuple = tuple(sorted(state.items()))

        if state\_tuple in visited:

            continue

        visited.add(state\_tuple)

        if is\_goal(state):

            print("\n--- Τελική κατάσταση επιτεύχθηκε (DFS) ---\n")

            print\_state(state)

            return path # Επιστροφή των βημάτων που οδηγήσαν στην τελική κατάσταση

        print("Εξετάζουμε νέα κατάσταση:")

        print\_state(state)

        for action, new\_state in successors(state):

            print(f"Εκτελώ: {action}")

            print\_state(new\_state)

            stack.append((new\_state, path + [action]))

    return None # Δεν βρέθηκε λύση

# Αρχική κατάσταση

initial\_state = {'a': 'c', 'b': 't', 'c': 't'}

# Εκτέλεση BFS και DFS

print("BFS Solution:")

print(bfs(initial\_state))

print("\nDFS Solution:")

print(dfs(initial\_state))

# Συμπέρασμα

Και ο BFS και ο DFS αλγόριθμος κατέληξαν στην καλύτερη λύση η οποία είναι

['move(a,t)', 'move(b,c)', 'move(a,b)']