



ΕΛΛΗΝΙΚΗ ΔΗΜΟΚΡΑΤΙΑ
ΠΑΝΕΠΙΣΤΗΜΙΟ ΚΡΗΤΗΣ

Δομές Δεδομένων

Ιωάννης Γ. Τόλλης
Τμήμα Επιστήμης Υπολογιστών
Πανεπιστήμιο Κρήτης

Χρηματοδότηση

- Το παρόν εκπαιδευτικό υλικό έχει αναπτυχθεί στα πλαίσια του εκπαιδευτικού έργου του διδάσκοντα.
- Το έργο «**Ανοικτά Ακαδημαϊκά Μαθήματα στο Πανεπιστήμιο Κρήτης**» έχει χρηματοδοτήσει μόνο τη αναδιαμόρφωση του εκπαιδευτικού υλικού.
- Το έργο υλοποιείται στο πλαίσιο του Επιχειρησιακού Προγράμματος «Εκπαίδευση και Δια Βίου Μάθηση» και συγχρηματοδοτείται από την Ευρωπαϊκή Ένωση (Ευρωπαϊκό Κοινωνικό Ταμείο) και από εθνικούς πόρους.



Σημείωμα αδειοδότησης

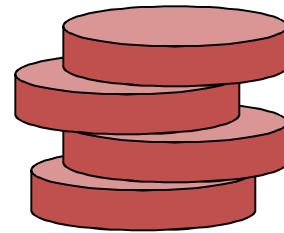
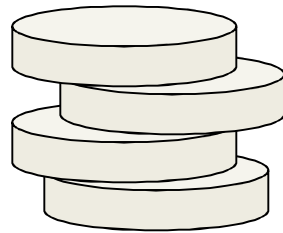
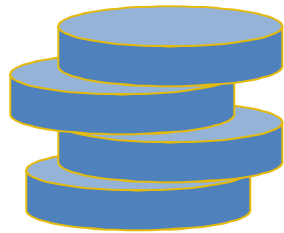
- Το παρόν υλικό διατίθεται με τους όρους της άδειας χρήσης Creative Commons Αναφορά, Μη Εμπορική Χρήση, Όχι Παράγωγο Έργο 4.0 [1] ή μεταγενέστερη, Διεθνής Έκδοση. Εξαιρούνται τα αυτοτελή έργα τρίτων π.χ. φωτογραφίες, διαγράμματα κ.λ.π., τα οποία εμπεριέχονται σε αυτό και τα οποία αναφέρονται μαζί με τους όρους χρήσης τους στο «Σημείωμα Χρήσης Έργων Τρίτων».

[1] <http://creativecommons.org/licenses/by-nc-nd/4.0/>



- Ως **Μη Εμπορική** ορίζεται η χρήση:
 - που δεν περιλαμβάνει άμεσο ή έμμεσο οικονομικό όφελος από την χρήση του έργου, για το διανομέα του έργου και αδειοδόχο
 - που δεν περιλαμβάνει οικονομική συναλλαγή ως προϋπόθεση για τη χρήση ή πρόσβαση στο έργο
 - που δεν προσπορίζει στο διανομέα του έργου και αδειοδόχο έμμεσο οικονομικό όφελος (π.χ. διαφημίσεις) από την προβολή του έργου σε διαδικτυακό τόπο
- Ο δικαιούχος μπορεί να παρέχει στον αδειοδόχο ξεχωριστή άδεια να χρησιμοποιεί το έργο για εμπορική χρήση, εφόσον αυτό του ζητηθεί.

Στοιίβες



Περιγραφή και Υλικό Ανάγνωσης

- Ο Αφηρημένος Τύπος Δεδομένων της Στοίβας (Stack Abstract Data Type (ADT)) (§2.1.1)
- Εφαρμογές για Στοίβες (§2.1.1)
- Υλοποίηση με βάση πίνακες (§2.1.1)
- Στοίβα βασισμένη σε πίνακα η οποία μπορεί να μεγαλώσει (§1.5)

Αφηρημένοι Τύποι Δεδομένων (ΑΤΔ)

- Ένας αφηρημένος τύπος δεδομένων είναι η αφηρημένη έννοια για μια δομής δεδομένων
- Ένας ΑΤΔ καθορίζει:
 - Τα δεδομένα που αποθηκεύονται
 - Τις πράξεις επάνω στα δεδομένα
 - Τις συνθήκες σφάλματος που σχετίζονται με πράξεις
- Παράδειγμα: Ένας ΑΤΔ που μοντελοποιεί ένα απλό σύστημα συναλλαγών μετοχών
 - Τα δεδομένα που αποθηκεύονται είναι εντολές αγοράς/πώλησης
 - Οι πράξεις που υποστηρίζονται είναι
 - order **buy**(stock, shares, price)
 - order **sell**(stock, shares, price)
 - void **cancel**(order)
 - Συνθήκες σφάλματος:
 - Αγορά/πώληση μιας ανύπαρκτης μετοχής
 - Ακύρωση μιας ανύπαρκτης εντολής

Ο ΑΤΔ της Στοίβας

- Ο ΑΤΔ της Στοίβας αποθηκεύει αυθαίρετα αντικείμενα
- Εισαγωγές και διαγραφές γίνονται με LIFO (last-in first-out)
- Σκεφτείτε ένα δοχείο με πιάτα με ελατήριο στον πάτο
- Κύριες πράξεις στοίβας:
 - `push(object)`: εισάγει ένα αντικείμενο
 - `object pop()`: διαγράφει και επιστρέφει το αντικείμενο που εισήχθη τελευταίο
- Βοηθητικές πράξεις στοίβας:
 - `object top()`: επιστρέφει το πιο πρόσφατα εισαχθέν αντικείμενο χωρίς να το διαγράψει
 - `integer size()`: επιστρέφει τον αριθμό των αντικειμένων που βρίσκονται στη στοίβα
 - `boolean isEmpty()`: δηλώνει εάν υπάρχουν αντικείμενα στη στοίβα

Exceptions

- Κατά την προσπάθεια εκτέλεσης μιας πράξης ενός ΑΤΔ μπορεί κάποιες φορές να προκληθεί μια κατάσταση σφάλματος που ονομάζεται exception
- Τα exceptions λέγεται ότι “πετάγονται” από μια πράξη που δεν μπορεί να εκτελεστεί
- Στον ΑΤΔ της Στοίβας οι πράξεις pop και top δεν μπορούν να εκτελεστούν αν η στοίβα είναι άδεια
- Η εκτέλεση της πράξης pop ή top σε μια άδεια στοίβα πετάει ένα [EmptyStackException](#)

Εφαρμογές για Στοίβες

- Άμεσες εφαρμογές
 - Το ιστορικό των σελίδων που επισκεφθήκαμε σε έναν Web browser
 - Η σειρά των πράξεων Undo σε έναν επεξεργαστή κειμένου
 - Η αλυσίδα των κλήσεων μεθόδων στην Java Virtual Machine
- Έμμεσες εφαρμογές
 - Βοηθητικές δομές δεδομένων σε αλγόριθμους
 - Συστατικό σε άλλες δομές δεδομένων

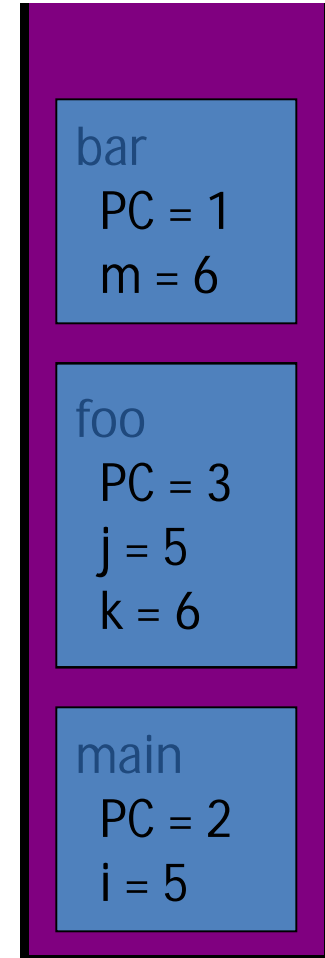
Η Στοιίβα Μεθόδων στη JVM

- Η Java Virtual Machine (JVM) παρακολουθεί την αλυσίδα των ενεργών μεθόδων με μια στοιίβα
- Όταν κληθεί μια μέθοδος, η JVM κάνει push ένα πλαίσιο το οποίο περιέχει
 - Τοπικές μεταβλητές και τιμή επιστροφής
 - Τον program counter, ο οποίος παρακολουθεί την εντολή η οποία εκτελείται
- Όταν τελειώνει μια μέθοδος, το πλαίσιο εξάγεται από την στοιίβα (γίνεται pop) και ο έλεγχος περνά στην μέθοδο που βρίσκεται πλέον στην κορυφή της στοιίβας

```
main() {  
    int i = 5;  
    foo(i);  
}
```

```
foo(int j) {  
    int k;  
    k = j+1;  
    bar(k);  
}
```

```
bar(int m) {  
    ...  
}
```



Στοιίβες

Στοίβα βασισμένη σε πίνακα

- Ένας απλός τρόπος υλοποίησης του ΑΤΔ της Στοίβας γίνεται με πίνακα
- Προσθέτουμε στοιχεία από αριστερά προς τα δεξιά
- Μια μεταβλητή παρακολουθεί τον δείκτη του πάνω στοιχείου της στοίβας

Algorithm *size()*

return $t + 1$

Algorithm *pop()*

if *isEmpty()* then

throw *EmptyStackException*

else

$t \leftarrow t - 1$

return $S[t + 1]$



Στοίβες

Στοίβα βασισμένη σε πίνακα (συνέχεια)

- Ο πίνακας που περιέχει τα στοιχεία της στοίβας μπορεί να γεμίσει
- Μια πράξη push θα πετάξει τότε ένα **FullStackException**
 - Περιορισμός της υλοποίησης με βάση πίνακες
 - Δεν είναι εγγενής στον ΑΤΔ της Στοίβας

```
Algorithm push(o)  
  if  $t = S.length - 1$  then  
    throw FullStackException  
  else  
     $t \leftarrow t + 1$   
     $S[t] \leftarrow o$ 
```



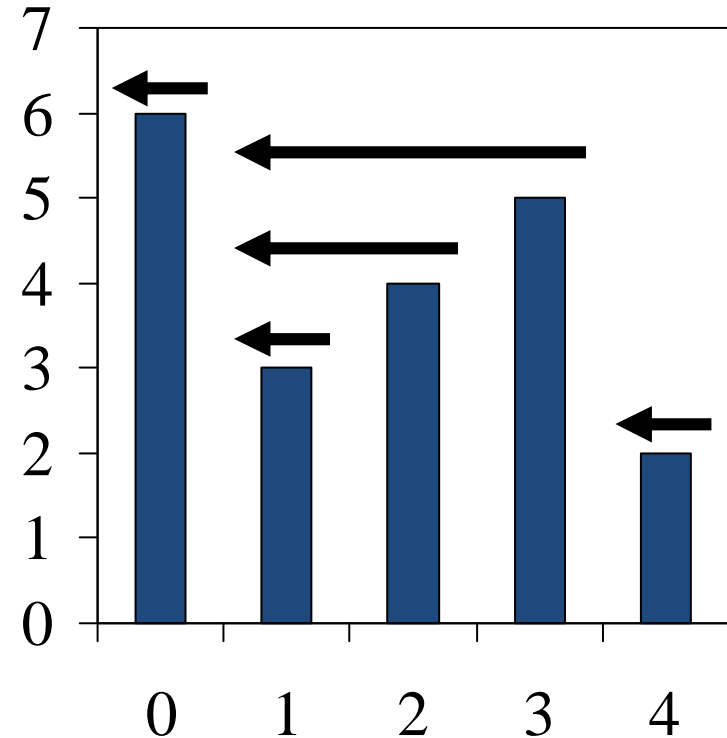
Στοίβες

Απόδοση και Περιορισμοί

- Απόδοση
 - Έστω ότι η στοίβα περιέχει n στοιχεία
 - Ο χώρος που καταλαμβάνεται είναι $O(n)$
 - Κάθε πράξη τρέχει σε χρόνο $O(1)$
- Περιορισμοί
 - Το μέγεθος της στοίβας πρέπει να ορισθεί εξ' αρχής και δεν μπορεί να αλλάξει
 - Το να γίνει push ένα νέο στοιχείο σε μια γεμάτη στοίβα προκαλεί μια exception που εξαρτάται από την υλοποίηση

Υπολογίζοντας Spans

- Θα δείξουμε πως μπορεί να χρησιμοποιηθεί η στοίβα σαν μια βοηθητική δομή δεδομένων σε έναν αλγόριθμο
- Δεδομένου ενός πίνακα X , το span $S[i]$ του $X[i]$ είναι ο μέγιστος αριθμός συνεχών στοιχείων $X[j]$ που βρίσκονται αμέσως πριν το $X[i]$ έτσι ώστε $X[j] \leq X[i]$
- Τα spans βρίσκουν εφαρμογή στην οικονομική ανάλυση
 - Π.χ., μετοχές σε μέγιστα 52 εβδομάδων



X	6	3	4	5	2
S	1	1	2	3	1

Στοίβες

Ο Αλγόριθμος σε Τετραγωνικό Χρόνο

Algorithm *spans1*(X, n)

Input array X of n integers

Output array S of spans of X

$S \leftarrow$ new array of n integers

for $i \leftarrow 0$ to $n - 1$ do

$s \leftarrow 1$

 while $s \leq i \wedge X[i - s] \leq X[i]$

$s \leftarrow s + 1$

$S[i] \leftarrow s$

return S

#

n

n

n

$1 + 2 + \dots + (n - 1)$

$1 + 2 + \dots + (n - 1)$

n

1

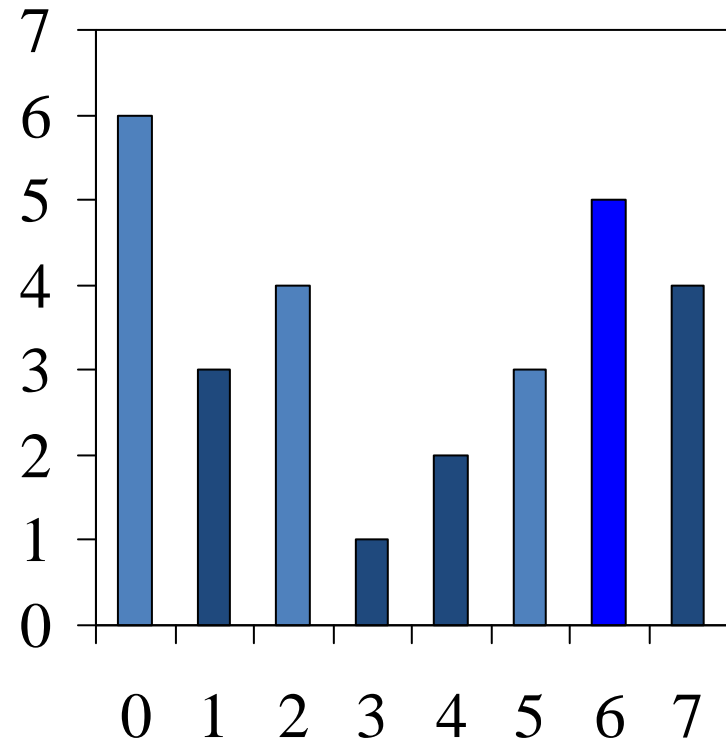
◆ Ο Αλγόριθμος *spans1* τρέχει σε χρόνο $O(n^2)$

Στοιβες

Υπολογίζοντας Spans χρησιμοποιώντας μια Στοίβα

- Κρατάμε σε μια στοίβα τους δείκτες των στοιχείων που είναι ορατά αν "κοιτάξουμε πίσω"
- Σαρώνουμε τον πίνακα από τα αριστερά προς τα δεξιά
 - Έστω ότι i είναι ο τρέχον δείκτης
 - Κάνουμε pop τους δείκτες από την στοίβα μέχρι να βρούμε ένα δείκτη j τέτοιο ώστε $X[i] < X[j]$
 - Θέτουμε $S[i] \leftarrow i - j$
 - Κάνουμε push το x στη στοίβα

Στοιβες



Ο Αλγόριθμος σε Γραμμικό Χρόνο

- Κάθε δείκτης του πίνακα
 - Εισάγεται (με push) στη στοίβα ακριβώς μια φορά
 - Εξάγεται (με pop) από την στοίβα το πολύ μια φορά
- Οι εντολές στον while βρόχο εκτελούνται το πολύ n φορές
- Ο Αλγόριθμος *spans2* τρέχει σε χρόνο $O(n)$

Algorithm <i>spans2</i> (X, n)	#
$S \leftarrow$ new array of n integers	n
$A \leftarrow$ new empty stack	1
for $i \leftarrow 0$ to $n - 1$ do	n
while ($\neg A.isEmpty()$ \wedge $X[top()] \leq X[i]$) do	n
$j \leftarrow A.pop()$	n
if $A.isEmpty()$ then	n
$S[i] \leftarrow i + 1$	n
else	
$S[i] \leftarrow i - j$	n
$A.push(i)$	n
return S	1

Στοίβα βασισμένη σε πίνακα η οποία μπορεί να μεγαλώσει

- Σε μια πράξη *push*, όταν ο πίνακας είναι γεμάτος, αντί να πετάξουμε ένα *exception*, μπορούμε να αντικαταστήσουμε τον πίνακα με έναν μεγαλύτερο
- Πόσο μεγάλος πρέπει να είναι ο νέος πίνακας;
 - αυξητική στρατηγική: αύξησε το μέγεθος κατά c (σταθερά)
 - στρατηγική διπλασιασμού: διπλασίασε το μέγεθος

```
Algorithm push(o)  
  if  $t = S.length - 1$  then  
     $A \leftarrow$  new array of  
      size ...  
    for  $i \leftarrow 0$  to  $t$  do  
       $A[i] \leftarrow S[i]$   
     $S \leftarrow A$   
   $t \leftarrow t + 1$   
   $S[t] \leftarrow o$ 
```

Σύγκριση των Στρατηγικών

- Θα συγκρίνουμε την αυξητική στρατηγική με την στρατηγική διπλασιασμού με το να αναλύσουμε τον συνολικό χρόνο που απαιτείται για να εκτελεστεί μια σειρά από n πράξεις push
- Υποθέτουμε ότι ξεκινάμε με μια άδεια στοίβα η οποία αντιπροσωπεύεται από έναν πίνακα μεγέθους 1
- Θα ονομάσουμε αποσβησμένο χρόνο μιας πράξης push τον μέσο χρόνο που χρειάζεται η push για όλη τη σειρά των πράξεων, δηλαδή, $T(n)/n$

Ανάλυση Αυξητικής Στρατηγικής

- Αντικαθιστούμε τον πίνακα $k = n/c$ φορές
- Ο συνολικός χρόνος $T(n)$ μιας σειράς από n πράξεις push είναι ανάλογος με

$$\begin{aligned}n + c + 2c + 3c + 4c + \dots + kc &= \\n + c(1 + 2 + 3 + \dots + k) &= \\n + ck(k + 1)/2 &\end{aligned}$$

- Αφού ο c είναι μια σταθερά, ο χρόνος $T(n)$ είναι $O(n + k^2)$, δηλαδή, $O(n^2)$
- Ο αποσβησμένος χρόνος μιας πράξης push είναι $O(n)$

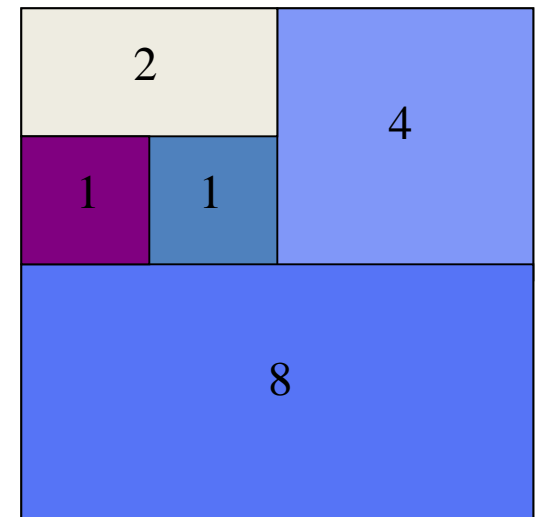
Ανάλυση Στρατηγικής Διπλασιασμού

- Αντικαθιστούμε τον πίνακα $k = \log_2 n$ φορές
- Ο συνολικός χρόνος $T(n)$ μιας σειράς από n πράξεις push είναι ανάλογος με

$$n + 1 + 2 + 4 + 8 + \dots + 2^k =$$
$$n + 2^{k+1} - 1 = 2n - 1$$

- Ο χρόνος $T(n)$ είναι $O(n)$
- Ο αποσβησμένος χρόνος μιας πράξης push είναι $O(1)$

geometric series



Το Interface της Στοίβας στη Java

- Το Java interface που αντιστοιχεί στον δικό μας ΑΤΔ της Στοίβας
- Απαιτεί τον ορισμό της κλάσης `EmptyStackException`
- Διαφέρει από την ενσωματωμένη στην Java κλάση `java.util.Stack`

```
public interface Stack {  
    public int size();  
    public boolean isEmpty();  
    public Object top()  
        throws EmptyStackException;  
    public void push(Object o);  
    public Object pop()  
        throws EmptyStackException;  
}
```

Στοίβα βασισμένη σε πίνακα στη Java

```
public class ArrayStack
    implements Stack {

    // holds the stack elements
    private Object S[ ];

    // index to top element
    private int top = -1;

    // constructor
    public ArrayStack(int capacity) {
        S = new Object[capacity];
    }
}
```

```
public Object pop()
    throws EmptyStackException {
    if isEmpty()
        throw new EmptyStackException
            ("Empty stack: cannot pop");
    Object temp = S[top];
    // facilitates garbage collection
    S[top] = null;
    top = top - 1;
    return temp;
}
```

Τέλος Ενότητας



Ευρωπαϊκή Ένωση
Ευρωπαϊκό Κοινωνικό Ταμείο



Με τη συγχρηματοδότηση της Ελλάδας και της Ευρωπαϊκής Ένωσης

