Δυναμικός Προγραμματισμός

Στην ενότητα αυτή θα μελετηθούν τα εξής θέματα:

Σχεδιασμός αλγορίθμων με Δυναμικό Προγραμματισμό

Το πρόβλημα του πολλαπλασιασμού πινάκων

Δυναμικός Προγραμματισμός

- Αν η λύση ενός προβλήματος μπορεί να εκφραστεί μαθηματικά με αναδρομικό τρόπο, τότε το πρόβλημα μπορεί να λυθεί από ένα αναδρομικό αλγόριθμο.

- Συχνά οι μεταγλωττιστές γλωσσών προγραμματισμού συντείνουν ώστε η εκτέλεση πολλών αναδρομικών προγραμμάτων να μην είναι αποδοτική.

- Σε τέτοιες περιπτώσεις μπορούμε να 'βοηθήσουμε' το μεταγλωττιστή μετατρέποντας τον αλγόριθμο σε μη-αναδρομικό αλγόριθμο ο οποίος συστηματικά φυλάει απαντήσεις υποπροβλημάτων σε ένα πίνακα.

- Μια τεχνική η οποία χρησιμοποιεί αυτή τη μέθοδο είναι γνωστή ως δυναμικός προγραμματισμός (dynamic programming).

Δυναμικός Προγραμματισμός

- Όπως και η μέθοδος "διαίρει και βασίλευε" ο δυναμικός προγραμματισμός *επιλύει προβλήματα συνδυάζοντας λύσεις υποπροβλήματων*.

- Σημαντική διαφορά των δύο μεθόδων: Η "διαίρει και βασίλευε" χωρίζει ένα πρόβλημα σε ανεξάρτητα υποπροβλήματα. Ο δυναμικός προγραμματισμός είναι εφαρμόσιμος και εκεί όπου τα υποπροβλήματα "επικαλύπτονται", και έχουν κοινά υποπροβλήματα.

- Ενώ ένας "διαίρει και βασίλευε" αλγόριθμος θα έλυνε κοινά υποπροβλήματα ξανά και ξανά, ένας αλγόριθμος δυναμικού προγραμματισμού λύνει κάθε υποπρόβλημα ακριβώς μια φορά και καταχωρεί τη λύση σε ένα πίνακα από όπου μπορεί να την διαβάσει κάθε φορά που το πρόβλημα ξανασυναντιέται.

Φάσεις στη σχεδίαση ενός αλγόριθμου ΔΠ

1. Χαρακτήρισε τη δομή μιας βέλτιστης λύσης.
2. Όρισε αναδρομικά την τιμή μιας βέλτιστης λύσης.
3. Υπολόγισε τη βέλτιστη λύση και την τιμή της "από κάτω προς τα πάνω".

Πολλαπλασιασμός πινάκων

- Δοθέντων πινάκων Α, Β, Γ, Δ, ποιος είναι ο πιο αποδοτικός τρόπος για να υπολογίσουμε το **γινόμενο Α×Β×Γ×Δ**;
- Υπάρχουν διάφορες επιλογές που αντιστοιχούν στους διαφορετικούς τρόπους τοποθέτησης παρενθέσεων:

  ((ΑΒ)Γ)Δ, (ΑΒ)(ΓΔ), (Α(ΒΓ))Δ, Α((ΒΓ)Δ), …

  Όλοι οι τρόποι οδηγούν στο ίδιο αποτέλεσμα λόγω της επιμεριστικότητας της πράξης του πολλαπλασιασμού πινάκων

- Όμως το κόστος (ο συνολικός αριθμός βημάτων) κάθε παρενθεσιοποίησης είναι διαφορετικός και εξαρτάται από τις διαστάσεις των πινάκων.

Πολλαπλασιασμός πινάκων

- **Κόστος πολλαπλασιασμού**: Αν ο πίνακας A έχει a σειρές και b στήλες και ο πίνακας B έχει b σειρές και c στήλες, τότε ο πίνακας AB έχει a σειρές και c στήλες και υπολογισμός του απαιτεί $a \cdot b \cdot c$ πολλαπλασιασμούς.

$$AB[i][j] = \sum_{k=1}^{b} A[i][k] \cdot B[k][j]$$

- Παράδειγμα: Έστω πίνακες A, B, Γ, Δ, διαστάσεων 10×20, 20×5, 5×40, 40×10. Τότε
  - Κόστος του γινομένου $((A\ B)\ \Gamma)\ \Delta$ είναι 7000
  - Κόστος του γινομένου $(A\ B)(\Gamma\ \Delta)$ είναι 3500

Το πρόβλημα

- Έστω πίνακες

  $A_1, A_2, .., A_n$

  όπου ο i-στός πίνακας έχει

  c_{i-1} σειρές και c_i στήλες

- Στόχος: εύρεση σειράς πολλαπλασιασμών για υπολογισμό του γινομένου

$$\Gamma = \prod_{i=1}^{n} A_i$$

η οποία να ελαχιστοποιεί τον συνολικό αριθμό πολλαπλασιασμών.

Λύση 1

- Εξαντλητική ανάλυση όλων των δυνατών παρενθεσιοποιήσεων.
- Δεν οδηγεί σε αποτελεσματικό αλγόριθμο:

  Έστω P(n) ο αριθμός των παρενθεσιοποιήσεων μιας αλυσίδας μήκους n. Αφού η αλυσίδα μπορεί να "σπάσει" μεταξύ της k και της k+1 θέσης για οποιοδήποτε k, $1 \le k \le n-1$, και οι προκύπτουσες μικρότερες αλυσίδες να παρενθεσιοποιηθούν αναδρομικά και ανεξάρτητα, έχουμε

$$P(n) = \begin{cases} \sum_{k=1}^{n-1} P(k)P(n-k), & \text{if } n \ge 2 \\ 1, & \text{if } n = 1 \end{cases}$$

- Μπορεί να αποδειχθεί ότι η λύση της πιο πάνω αναδρομική εξίσωσης είναι

$$P(n) = \frac{1}{n}\binom{2(n-1)}{n-1} \in \Omega\left(\frac{4^n}{n^{3/2}}\right)$$

Λύση 2

Φάση 1: Χαρακτηρισμός δομής βέλτιστης λύσης.

ΙΔΙΟΤΗΤΑ ΒΕΛΤΙΣΤΗΣ ΠΑΡΕΝΘΕΣΙΟΠΟΙΗΣΗΣ

Έστω ότι η βέλτιστη παρενθεσιοποίηση 'σπάζει' την αλυσίδα μεταξύ των θέσεων k και k+1.

Τότε οι παρενθεσιοποιήσεις των υποαλυσιδών

$A_1, \ldots, A_k$ και $A_{k+1}, \ldots, A_n$

είναι και αυτές βέλτιστες.

Διαφορετικά, μια καλύτερη παρενθεσιοποίηση της υποαλυσίδας θα οδηγούσε σε παρενθεσιοποίηση ολόκληρης της αλυσίδας καλύτερης από τη βέλτιστη.

Φάση 2: Αναδρομικός ορισμός βέλτιστης λύσης

- Γράφουμε

  A[i,j] για το γινόμενο $A_i \times \ldots \times A_j$

  και

  m[i,j] για τον ελάχιστο αριθμό πολλαπλασιασμών που απαιτούνται για τον υπολογισμό του πίνακα A[i,j].

- Αν l≠r, έχουμε

$$m[l,r] = \min_{l \le i \le r} (m[l,i] + m[i+1,r] + c_{l-1} c_i c_r)$$

- Θέτουμε m[i,i]=0, και έτσι παίρνουμε ένα σύνολο αναδρομικών εξισώσεων.

- Το ελάχιστο κόστος υπολογισμού του γινομένου Γ δίνεται από το m[1,n].

Αλγόριθμος 1

- Σε αυτό το σημείο φαίνεται προφανής ο πιο κάτω αναδρομικός αλγόριθμος ο οποίος υπολογίζει το c[n,m].

```
multiply(int l, int r){
if (l==r) return 0;
else
   m = ∞;
   for (i=1; i<n; i++){
   k = multiply(l,i)
        + multiply(i+1,r)
        + c[l-1]·c[i]·c[r];
   m = min(m,k);
   }
}
```

- Ποιος ο χρόνος εκτέλεσης της πιο πάνω διαδικασίας;

Αλγόριθμος 1 – Χρόνος Εκτέλεσης

- Έστω T(n) ο χρόνος εκτέλεσης της διαδικασίας multiply(l,r), με r-l +1= n. Έχουμε την πιο κάτω αναδρομική σχέση:

$$T(1) \geq 1$$

$$T(n) \geq 1 + \sum_{k=1}^{n-1} (T(k) + T(n-k) + 1), \quad \text{if } n > 1$$

- Αφού κάθε όρος T(ι) εμφανίζεται δύο φορές μέσα στο άθροισμα έχουμε ότι

$$T(n) \geq 1 + 2 \cdot \sum_{k=1}^{n-1} T(k) \quad + n - 1$$

$$= n + 2 \cdot \sum_{k=1}^{n-1} T(k)$$

- Επιλύουμε την αναδρομική ανισότητα με τη μέθοδο της επαγωγής.
  1. Προβλέπουμε ότι $T(n) \in \Omega(2^n)$.
  2. Θα δείξουμε με τη μέθοδο της επαγωγής ότι για κάθε $n \geq 1$ $T(n) \geq 2^{n-1}$.

Αλγόριθμος 1 – Χρόνος Εκτέλεσης

- Προφανώς η πρόταση ισχύει για n=1.
- Υποθέτουμε ότι ισχύει για κάθε k<n. Τότε

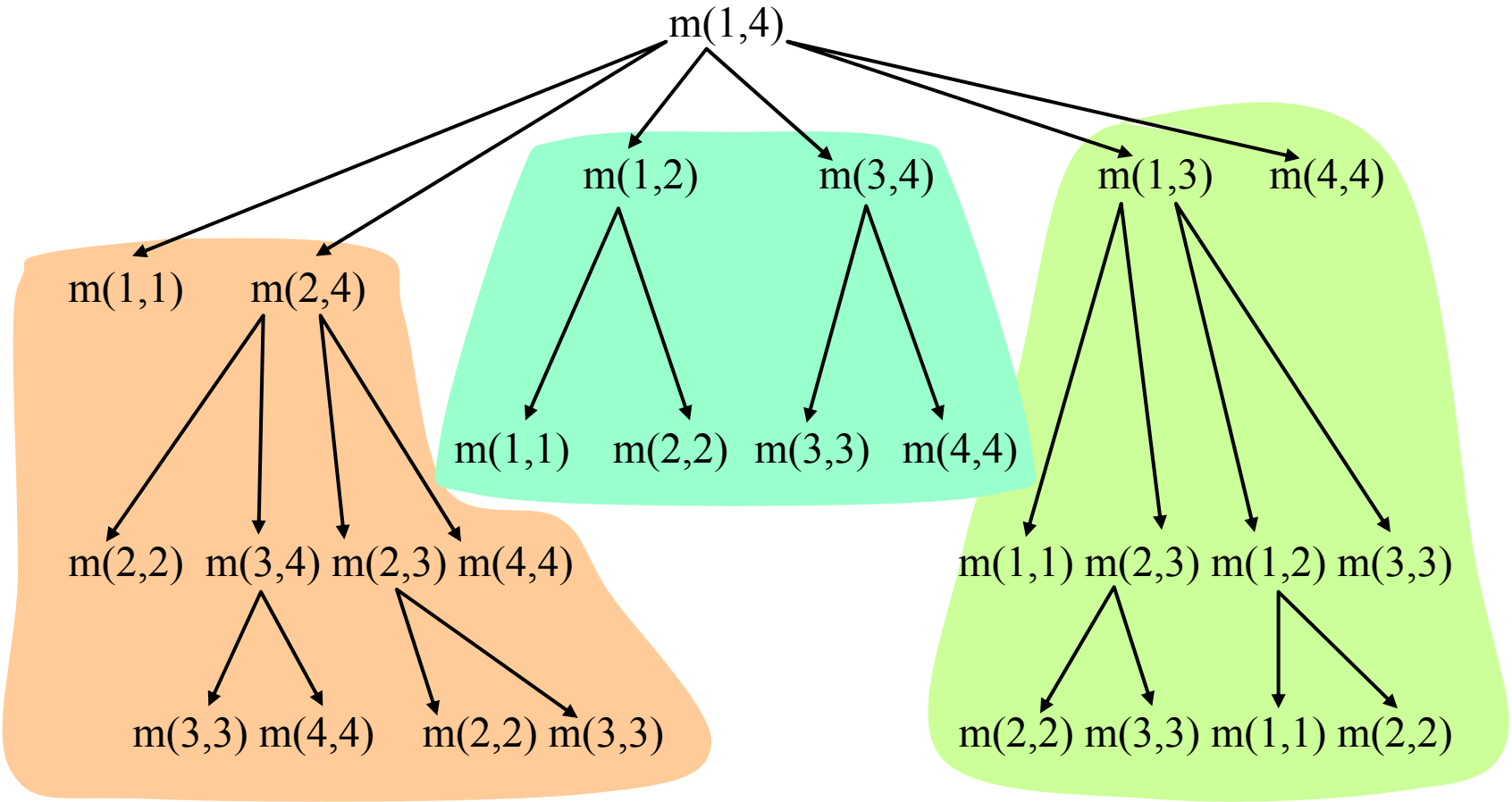
$$
\begin{aligned}
T(n) &\geq n + 2 \cdot \sum_{k=1}^{n-1} T(k) \\
&\geq n + 2 \cdot \sum_{k=1}^{n-1} 2^{k-1} \\
&\geq n + 2 \cdot \sum_{k=1}^{n-2} 2^{k} \\
&\geq n + 2 \cdot \frac{2^{n-1} - 1}{2 - 1} \\
&\geq n + 2^{n} - 2 \geq 2^{n-1}
\end{aligned}
$$

  όπως χρειάζεται.

- Συνεπώς ο αλγόριθμος είναι εκθετικός, και δεν αποτελεί βελτίωση από την εξαντλητική ανάλυση της λύσης 1.

Αλγόριθμος 1 – Χρόνος Εκτέλεσης

- Γιατί απέτυχε ο προφανής αλγόριθμος;



Φάση 3: Υπολογισμός βέλτιστης λύσης από ‘κάτω προς τα πάνω’

- Πόσα υποπροβλήματα υπάρχουν;

  Όσα και τα ζεύγη i, j, $1 \le i \le j \le n$, δηλαδή

$$\binom{n}{2} + n \in \Theta(n^2)$$

- **Θα υπολογίσουμε τις τιμές m[l,r], από ‘κάτω’ και πηγαίνοντας προς τα ‘πάνω’: αρχικά το κόστος ακολουθιών μήκους 0, στη συνέχεια ακολουθιών μήκους 1, 2, …**

- Διατηρούμε τις τιμές στον πίνακα C, του οποίου αρχικά όλες οι θέσεις έχουν τη τιμή 0.

- Ο πίνακας D περιέχει τις διαστάσεις των πινάκων, δηλαδή:

$$D = [c_0, c_1, \ldots, c_n]$$

Αλγόριθμος 2

```
for (l = 1; l <= n; l++)
   C[l,l] = 0;

for (k=1; k<=n; k++)
    for (l=1; l<= n-k; l++)
        r=l+k;
        C[l,r]= ∞;
        for (i=l; i<r; i++)
              temp = C[l,i]+C[i+1,r]
                     + D[l-1]·D[i]·D[r];
              if (C[l,r] > temp)
                    C[l,r] = temp;
```

Ανάλυση Αλγόριθμου 2

- Με τον πιο πάνω αλγόριθμο, το κάθε υποπρόβλημα λύνεται μόνο μια φορά. Η λύση του καταχωρείται στον πίνακα C από όπου μπορεί να χρησιμοποιηθεί κάθε φορά που το υποπρόβλημα καλείται από ένα μεγαλύτερο (υπο)πρόβλημα.

- *Ανάλυση χρόνου εκτέλεσης:*

  Υπάρχουν $\Theta(n^2)$ υποπροβλήματα. Το κάθε ένα από αυτά απαιτεί χρόνο $O(n)$ (για εύρεση του ελάχιστου από τα $O(n)$ αθροίσματα που αντιστοιχούν σε κάθε πιθανή διάσπαση της αλυσίδας).

- Άρα συνολικά απαιτείται χρόνος $O(n^3)$

Παράδειγμα Εκτέλεσης

- Έστω πίνακες

A1	διαστάσεων	6×5
A2	διαστάσεων	5×8
A3	διαστάσεων	8×4
A4	διαστάσεων	4×10
A5	διαστάσεων	10×3
A6	διαστάσεων	3×7

π.χ. η θέση m[2,5] υπολογίζεται ως εξής

$$m[2,5] = \begin{cases} m[2,2] + m[3,5] + c_1 c_2 c_5 = 0 + 216 + 5 \cdot 8 \cdot 3 = 336, \\ m[2,3] + m[4,5] + c_1 c_3 c_5 = 160 + 120 + 5 \cdot 4 \cdot 3 = 340, \\ m[2,4] + m[5,5] + c_1 c_4 c_5 = 360 + 0 + 5 \cdot 10 \cdot 3 = 510 \end{cases}$$
$$= 336$$

Παράδειγμα Εκτέλεσης

- Έστω A1[6,5], A2[5,8], A3[8,4], A4[4,10], A5[10,3], A6[3,7],

i/j	1	2	3	4	5	6
1	0	240	280	520	426	552
2		0	160	360	336	441
3			0	320	216	384
4				0	120	204
5					0	210
6						0