

Problem Solving - Στρατηγική

Εργαλεία και ιδέες

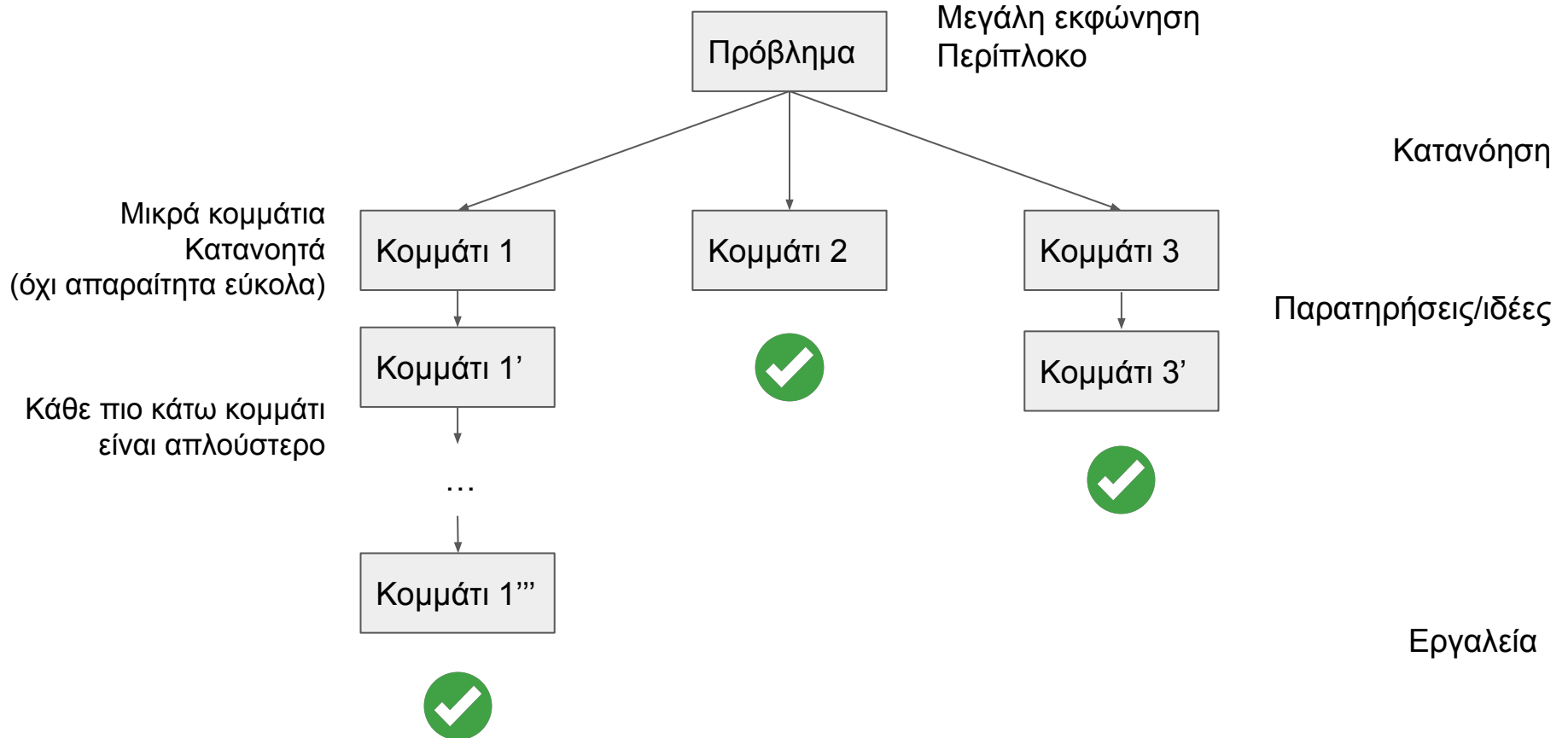
A fool with a tool is still a fool

- Grady Booch

Χρησιμοποιούμε ιδέες για να σπάσουμε ένα πρόβλημα σε πιο μικρά και εύκολα προβλήματα

Χρησιμοποιούμε εργαλεία όταν απλοποιήσουμε το πρόβλημα αρκετά έτσι ώστε να ταιριάζει ακριβώς στο καλούπι

Problem solving



Subtasks: ο σωσίας μας

- Παρόμοια χαρακτηριστικά με το full πρόβλημα
- ΑΛΛΆ πιο ειδικό => πιο εύκολο
- Συχνά είδη subtasks:
 - Χρονικά - π.χ. $N \leq 4000$ (Τι χρονική πολυπλοκότητα μπορείτε να χρησιμοποιήσετε;)
 - Ειδικές περιπτώσεις: π.χ. $K=1$
 - By hand: π.χ. $N=3$
- Progression:
 - Τα subtasks πολλές φορές βοηθούν στο να βρούμε την λύση του επόμενου subtask

The numbers don't lie

EJOI 2022: Αν κάποιος έπιανε τα πρώτα 1-2 subtasks από κάθε άσκηση, θα έπιανε χάλκινο μετάλλιο

EJOI 2021: Κάτι παρόμοιο

Παραδείγματα subtasks:

1. (4 βαθμοί): $l_i = r_i$ ($1 \leq i \leq m$)
2. (6 βαθμοί): Το άθροισμα των m για όλες τις περιπτώσεις ελέγχου δεν υπερβαίνει το 10
3. (10 βαθμοί): Το άθροισμα των m για όλες τις περιπτώσεις ελέγχου δεν υπερβαίνει το 20

1. (3 βαθμοί): $m = 1$
2. (6 βαθμοί): $b_{i+1} = b_i$ ($1 \leq i < m$), δηλαδή όλα τα στοιχεία του b είναι τα ίδια
3. (15 βαθμοί): $b_{i+1} \bmod b_i = 0$ ($1 \leq i < m$)
4. (9 βαθμοί): $1 \leq m \leq 7$
5. (11 βαθμοί): $1 \leq m \leq 20$

Πως πιάνουμε τα πρώτα subtasks όμως;

Ποια είναι η σωστή στρατηγική;

Πότε πρέπει να αλλάξω πρόβλημα;

Πως αντέχω για να μην κουράζομαι στην μέση του διαγωνισμού;

Πως δεν αποθαρρύνομαι όταν έχω 0 πόντους;

Step 1: Διαβάζω το πρόβλημα

Step 2: **Καταλαβαίνω** το πρόβλημα

Time Limit Exceeded: Electric Boogaloo
aka **Brute Force & Binary Search**

Όταν σας είπαν ότι δεν θα ξαναπιάσετε TLE



Ah shit, here we go again.

Unexpected TLE vs Expected TLE

Υπολογισμός time complexities \Rightarrow Αποφυγή time limit exceeded

Κάποιες φορές όμως το TLE δεν είναι τόσο κακό

Κάποιοι πόντοι $>$ Μηδέν πόντοι

Brute Force / Complete Search

Δοκιμάζουμε ΌΛΕΣ τις πιθανές περιπτώσεις, και επιλέγουμε αυτή που ταιριάζει

Έλεγχος όλων των περιπτώσεων \Rightarrow Εγγύηση σωστής απάντησης (αν ο κώδικας μας είναι σωστός)

Για παράδειγμα: Θέλω να βρω τον μικρότερο αριθμό στον πιο κάτω πίνακα

6	3	5	1	7	2	4	8
---	---	---	---	---	---	---	---

Πιθανές λύσεις;

Δομή λύσης με brute force

Πολλές φορές η λύση μας στο brute force έχει την εξής μορφή:

```
for(int i=0;i<(αριθμός περιπτώσεων);i++){  
    int periptosi = periptoseis[i];  
    // Ελέγχουμε αν η περίπτωση τηρεί τις προϋποθέσεις,  
    // σύμφωνα με το πρόβλημα  
    if (check(periptosi)) { cout << periptosi << endl;}  
}
```

Πως κινούμαστε προς την λύση

Σύμφωνα με την δομή που είδαμε πριν, χρειαζόμαστε 2 δεδομένα, τα οποία πρέπει να πιάσουμε από την εκφώνηση και να μετατρέψουμε σε κώδικα:

- Ποιες είναι οι περιπτώσεις που έχουμε;
- Πότε είναι “σωστή” μια περίπτωση;

Τι είναι οι απαντήσεις για το προηγούμενο παράδειγμα;

Πως κινούμαστε προς την λύση

Σύμφωνα με την δομή που είδαμε πριν, χρειαζόμαστε 2 δεδομένα, τα οποία πρέπει να πιάσουμε από την εκφώνηση και να μετατρέψουμε σε κώδικα:

- Ποιες είναι οι περιπτώσεις που έχουμε;
- Πότε είναι “σωστή” μια περίπτωση;

Τι είναι οι απαντήσεις για το προηγούμενο παράδειγμα;

- Ο κάθε αριθμός είναι μια “περίπτωση”
- Για να είναι σωστή μια περίπτωση πρέπει να είναι μικρότερη από όλες τις άλλες περιπτώσεις

Χρονική πολυπλοκότητα;

Για την δομή που δείξαμε, η χρονική πολυπλοκότητα είναι:

$$O(\text{αριθμός περιπτώσεων}) \times (\text{χρόνος για έλεγχο κάθε περίπτωσης})$$

Στο παράδειγμα που δώσαμε είναι $O(N^2)$

Ο “πιο απλός” τρόπος να το λύσουμε είναι πιο γρήγορος!

Συχνά μοτίβα

Μπορούμε να χωρίσουμε τα πιο συχνά προβλήματα που θα βρούμε που λύνονται με brute force στις πιο κάτω κατηγορίες:

- Επιλογή - Πρέπει να επιλέξουμε κάποια από τα στοιχεία ενός πίνακα
- Μεταθέσεις - Πρέπει να βρούμε την σειρά που πρέπει να βάλουμε κάποια στοιχεία
- Recursion

Σε ποια κατηγορία ανήκει το παράδειγμα που είδαμε;

Επιλογή

Παράδειγμα:

Σας δίνονται N αριθμοί. Να βρείτε αν υπάρχει τρόπος να επιλέξουμε κάποια από αυτά, τα οποία να έχουν άθροισμα K .

π.χ.

3	5	1	7
---	---	---	---

$K=9$

Απάντηση: Υπάρχει τρόπος (δεν μας ενδιαφέρει προς το παρόν το ποιος είναι αυτός ο τρόπος)

Επιλογή

Θυμάστε όσα είδατε τις προηγούμενες μέρες;



Επιλογή

3	5	1	7
---	---	---	---

Οι περιπτώσεις μας είναι:

$\{\}$	$\{3\}$
$\{7\}$	$\{3, 7\}$
$\{1\}$	$\{3, 1\}$
$\{1,7\}$	$\{3, 1,7\}$
$\{5\}$	$\{3, 5\}$
$\{5,7\}$	$\{3, 5,7\}$
$\{5,1\}$	$\{3, 5,1\}$
$\{5,1,7\}$	$\{3, 5,1,7\}$

Πως μπορούμε να τις δοκιμάσουμε
όλες όπως στην δομή μας;

Επιλογή

3	5	1	7
---	---	---	---

Οι περιπτώσεις μας είναι:

{}	0000	{3}	1000
{7}	0001	{3, 7}	1001
{1}	0010	{3, 1}	1010
{1,7}	0011	{3, 1,7}	1011
{5}	0100	{3, 5}	1100
{5,7}	0101	{3, 5,7}	1101
{5,1}	0110	{3, 5,1}	1110
{5,1,7}	0111	{3, 5,1,7}	1111

Επιλογή

3	5	1	7
---	---	---	---

Οι περιπτώσεις μας είναι:

$$\{\} \quad \{\} \Rightarrow 0$$

$$\{7\} \quad \{1\} \Rightarrow 1$$

$$\{1\} \quad \{1,0\} \Rightarrow 2$$

$$\{1,7\} \quad \{1,1\} \Rightarrow 3$$

$$\{5\} \quad 0100 \Rightarrow 4$$

$$\{5,7\} \quad 0101 \Rightarrow 5$$

$$\{5,1\} \quad 0110 \Rightarrow 6$$

$$\{5,1,7\} \quad 0111 \Rightarrow 7$$

$$\{3\} \quad 1000 \Rightarrow 8$$

$$\{3,7\} \quad 1001 \Rightarrow 9$$

$$\{3,1\} \quad 1010 \Rightarrow 10$$

$$\{3,1,7\} \quad 1011 \Rightarrow 11$$

$$\{3,5\} \quad 1100 \Rightarrow 12$$

$$\{3,5,7\} \quad 1101 \Rightarrow 13$$

$$\{3,5,1\} \quad 1110 \Rightarrow 14$$

$$\{3,5,1,7\} \quad 1111 \Rightarrow 15$$

Πόσες διαφορετικές επιλογές έχουμε;

Επιλογή

Μέχρι τώρα, έχουμε

```
bool iparxei = false;
for(int i=0;i<pow(2, N);i++){
    vector<bool> periptosi = binary(i);
    if (check(periptosi)) { iparxei = true; }
}
if (iparxei) cout << "YES" << endl;
else cout << "NO" << endl;
```

Επιλογή

Μέχρι τώρα, έχουμε

```
bool iparxei = false;
for(int i=0;i<pow(2, N);i++){
    vector<bool> periptosi = binary(i);
    if (check(periptosi)) { iparxei = true; }
}
if (iparxei) cout << "YES" << endl;
else cout << "NO" << endl;
```

Aside 1

Πως μπορούμε να μετατρέψουμε ένα αριθμό στην δυαδική του αναπαράσταση;

π.χ. `binary(10) ⇒ 1010`

10 ⇒ 5 ⇒ 2 ⇒ 1 ⇒ 0
0 1 0 1

Απάντηση: `1010`

13 ⇒ 6 ⇒ 3 ⇒ 1 ⇒ 0
1 0 1 1

Απάντηση: `1101`

Τι κάνω με κάθε κίνηση στα δεξιά;

Τι είναι οι αριθμοί στο κάτω μέρος;

Πως πάω από τους αριθμούς στην απάντηση;

Επιλογή

Μέχρι τώρα, έχουμε

```
bool iparxei = false;
for(int i=0;i<pow(2, N);i++){
    vector<bool> periptosi = binary(i);
    if (check(periptosi)) { iparxei = true; }
}
if (iparxei) cout << "YES" << endl;
else cout << "NO" << endl;
```


Έλεγχος

Πως ελέγχουμε αν η περίπτωση μας πληροί τις προϋποθέσεις;

<code>array</code>	3	5	1	7
<code>binary(i)</code>	1	0	1	0

Πιο βασική ερώτηση (και πιο σημαντική): ΠΟΙΕΣ είναι οι προϋποθέσεις;

Επιλογή

Ξέρουμε πως μπορούμε να υλοποιήσουμε όλα τα κομμάτια!

Τι χρονική πολυπλοκότητα έχει ο κώδικας;

Μέχρι τώρα, έχουμε

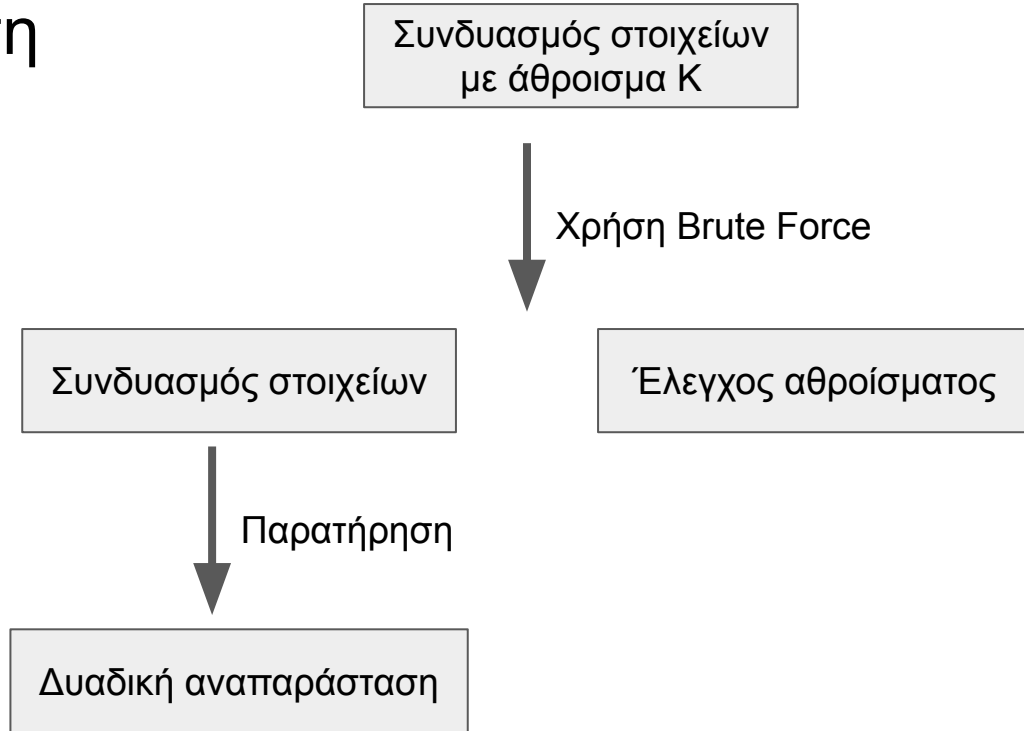
```
bool iparxei = false;
for(int i=0;i<pow(2, N);i++){
    vector<bool> periptosi = binary(i);
    if (check(periptosi)) { iparxei = true; }
}
if (iparxei) cout << "YES" << endl;
else cout << "NO" << endl;
```

Ανασκόπηση

Είχαμε ένα πρόβλημα το οποίο θέλαμε να λύσουμε. Τα βήματα που ακολουθήσαμε:

1. Αναγνωρίσαμε ότι μπορούμε να χρησιμοποιήσουμε brute force
 - Αυτό δεν θα μπορούσατε να το κάνετε κατά την διάρκεια του διαγωνισμού, επειδή τώρα μαθαίνετε την τεχνική (hopefully όταν το δείτε στο μέλλον σε κάποιο διαγωνισμό να μπορείτε να το αντιμετωπίσετε)
2. Σπάσαμε το πρόβλημα σε πιο μικρά προβλήματα (ποιές είναι οι περιπτώσεις, πως τις ελέγχουμε)
3. Λύσαμε τα πιο μικρά προβλήματα

Ανασκόπηση





Let's code

Ας το κάνουμε λίγο πιο δύσκολο

Πως μπορούμε να λύσουμε το ίδιο πρόβλημα αλλά να βρούμε τον τρόπο που να χρησιμοποιεί το ελάχιστο πλήθος αριθμών;

Μεταθέσεις

Ταξινομήστε τον πίνακα $\begin{bmatrix} 7 & 9 & 2 \end{bmatrix}$ χρησιμοποιώντας brute force

Μεταθέσεις

Ταξινομήστε τον πίνακα 7 9 2 χρησιμοποιώντας brute force

Περιπτώσεις: Όλες οι πιθανές μεταθέσεις του 7 9 2:

- 7 9 2
- 7 2 9
- 2 9 7
- 2 7 9
- 9 2 7
- 9 7 2

Μεταθέσεις

Ταξινομήστε τον πίνακα 7 9 2 χρησιμοποιώντας brute force

Περιπτώσεις: Όλες οι πιθανές μεταθέσεις του 7 9 2:

- 7 9 2
- 7 2 9
- 2 9 7
- 2 7 9
- 9 2 7
- 9 7 2

Έλεγχος: Τα στοιχεία να εμφανίζονται σε αύξουσα σειρά

Δημιουργία μεταθέσεων

Πως μπορούμε να δημιουργήσουμε όλες τις μεταθέσεις;

Δημιουργία μεταθέσεων

Πως μπορούμε να δημιουργήσουμε όλες τις μεταθέσεις;

- Μόνοι μας: Δεν είναι τόσο απλό
- Χρησιμοποιώντας την STL: Αρκετά πιο απλό
 - `next_permutation`
 - Note: Για να περάσουμε από όλα τα στοιχεία πρέπει ο πίνακας μας να είναι ταξινομημένος (άρα χάνουμε την ουσία για το συγκεκριμένο πρόβλημα, αλλά ας το αγνοήσουμε προς το παρόν)

Κώδικας

```
vector<int> nums;  
sort(nums.begin(), nums.end());  
do {  
    if(check(nums)) {  
        break;  
    }  
} while(next_permutation(nums.begin(), nums.end()));  
for(int i=0;i<n;i++) cout<<nums[i]<<" ";
```

Binary Search

Brute Force



Brute Force



Binary Search



Ιδέα; Εργαλείο; (και τα 2!)

Ιδέα: Διαίρει και βασίλευε

Από το λεξικό:

για να κυβερνάς με ασφάλεια και να διατηρείς την εξουσία σου,
φρόντισε να σπείρεις τη διχόνοια ανάμεσα στους αντιπάλους σου

Ιδέα: Διαίρει και βασίλευε

Από το λεξικό:

για να κυβερνάς με ασφάλεια και να διατηρείς την εξουσία σου, φρόντισε να σπείρεις τη διχόνοια ανάμεσα στους αντιπάλους σου

Για εμάς:

- Διαιρούμε το πρόβλημα (συνήθως στην μέση)
 - Τι ακριβώς διαιρούμε; Τις πιθανές απαντήσεις
- Λύνουμε τα 2 μισά (πως λύνουμε τα 2 μισά;)
- Ενώνουμε τις λύσεις μας

Διαίρει και βασίλευε

Από το λεξικό:

για να κυβερνάς με ασφάλεια και να διατηρείς την εξουσία σου, φρόντισε να σπείρεις τη διχόνοια ανάμεσα στους αντιπάλους σου

Για εμάς:

- Διαιρούμε το πρόβλημα (συνήθως στην μέση)
- Λύνουμε τα 2 μισά (πως λύνουμε τα 2 μισά; **τα ξανασπάζουμε στην μέση**)
- Ενώνουμε τις λύσεις μας

Binary Search

Έχουμε ένα πίνακα με αριθμούς σε αύξουσα σειρά. Θέλουμε να δούμε αν υπάρχει ο αριθμός K .

Για παράδειγμα ο πιο κάτω πίνακας με $K=17$

1	2	4	5	6	7	8	9	10	11	15	16	18	21	24	27
---	---	---	---	---	---	---	---	----	----	----	----	----	----	----	----

Πρώτο βήμα στο Δ&Β;

Binary Search

Έχουμε ένα πίνακα με αριθμούς σε αύξουσα σειρά. Θέλουμε να δούμε αν υπάρχει ο αριθμός K .

Για παράδειγμα ο πιο κάτω πίνακας με $K=19$

1	2	4	5	6	7	8	9	10	11	15	16	18	21	24	27
---	---	---	---	---	---	---	---	----	----	----	----	----	----	----	----

1	2	4	5	6	7	8	9
---	---	---	---	---	---	---	---

10	11	15	16	18	21	24	27
----	----	----	----	----	----	----	----

Binary Search

ΠΑΡΑΤΗΡΗΣΗ 1: Αφού οι αριθμοί είναι ταξινομημένοι, μπορούμε να διαπιστώσουμε σε ποιο μισό του πίνακα θα βρίσκεται το Κ αν υπάρχει

1	2	4	5	6	7	8	9	10	11	15	16	18	21	24	27
---	---	---	---	---	---	---	---	----	----	----	----	----	----	----	----

1	2	4	5	6	7	8	9
---	---	---	---	---	---	---	---

10	11	15	16	18	21	24	27
----	----	----	----	----	----	----	----

Binary Search

ΠΑΡΑΤΗΡΗΣΗ 2: Μπορούμε να δούμε τον μισό πίνακα που απέμεινε ως το ίδιο πρόβλημα, με πιο μικρή είσοδο

10	11	15	16	18	21	24	27
----	----	----	----	----	----	----	----

Binary Search

ΠΑΡΑΤΗΡΗΣΗ 2: Μπορούμε να δούμε τον μισό πίνακα που απέμεινε ως το ίδιο πρόβλημα, με πιο μικρή είσοδο

10	11	15	16	18	21	24	27
----	----	----	----	----	----	----	----

10	11	15	16
----	----	----	----

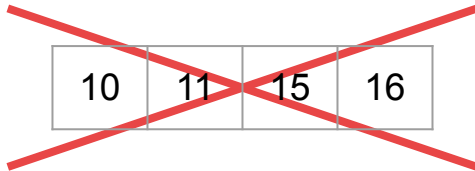
18	21	24	27
----	----	----	----

Binary Search

ΠΑΡΑΤΗΡΗΣΗ 2: Μπορούμε να δούμε τον μισό πίνακα που απέμεινε ως το ίδιο πρόβλημα, με πιο μικρή είσοδο

10	11	15	16	18	21	24	27
----	----	----	----	----	----	----	----

10	11	15	16
----	----	----	----



18	21	24	27
----	----	----	----

Binary Search

Επαναλαμβάνουμε:

10	11	15	16	18	21	24	27
----	----	----	----	----	----	----	----

10	11	15	16
---------------	---------------	---------------	---------------

18	21	24	27
----	----	----	----

18	21
----	----

24	27
---------------	---------------

Binary Search

Επαναλαμβάνουμε:

10	11	15	16	18	21	24	27
----	----	----	----	----	----	----	----

10	11	15	16
---------------	---------------	---------------	---------------

18	21	24	27
----	----	----	----

18	21
----	----

24	27
---------------	---------------

18

21

Binary Search

Όταν φτάσουμε σε κομμάτια μεγέθους 1, απλά ελέγχουμε αν είναι αυτό που ψάχνουμε

10	11	15	16	18	21	24	27
----	----	----	----	----	----	----	----

10	11	15	16
----	----	----	----

18	21	24	27
----	----	----	----

18	21
----	----

24	27
----	----

18

21

Χρονική Πολυπλοκότητα;

Με κάθε βήμα του αλγορίθμου μας, το μέγεθος του πίνακα διαιρείται με το 2

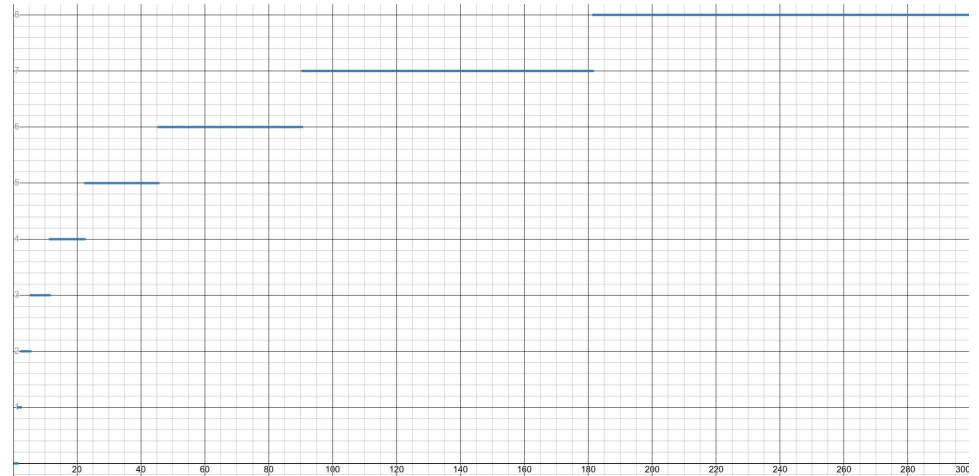
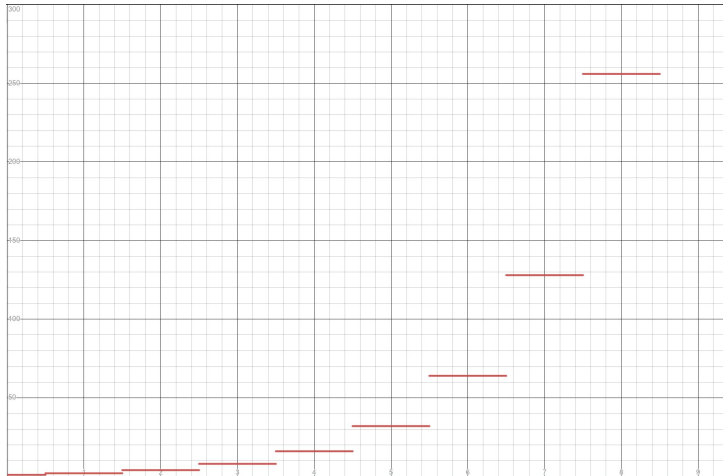
Πόσες φορές πρέπει να διαιρέσουμε το N με το 2 έτσι ώστε να φτάσει στο 1;

Χρονική Πολυπλοκότητα;

Με κάθε βήμα του αλγορίθμου μας, το μέγεθος του πίνακα διαιρείται με το 2

Πόσες φορές πρέπει να διαιρέσουμε το N με το 2 έτσι ώστε να φτάσει στο 1;

Πόσες φορές πρέπει να διπλασιάσουμε το 1 για να φτάσουμε το N ;



Χρονική Πολυπλοκότητα;

Με κάθε βήμα του αλγορίθμου μας, το μέγεθος του πίνακα διαιρείται με το 2

Πόσες φορές πρέπει να διαιρέσουμε το N με το 2 έτσι ώστε να φτάσει στο 1;

Πόσες φορές πρέπει να διπλασιάσουμε το 1 για να φτάσουμε το N ;

Η πολυπλοκότητα είναι $O(\log N)$

- Πάρα πολύ γρήγορο, μπορεί να δουλέψει για πολύ μεγάλα N

Χρήση του binary search

Το binary search πολλές φορές θα το χρησιμοποιήσουμε όχι με πίνακα, αλλά καθώς ψάχνουμε την απάντηση:

Για παράδειγμα: Να βρείτε την τετραγωνική ρίζα του N , χωρίς να χρησιμοποιήσετε την συνάρτηση `sqrt()`

Χρήση του binary search

Το binary search πολλές φορές θα το χρησιμοποιήσουμε όχι με πίνακα, αλλά καθώς ψάχνουμε την απάντηση:

Για παράδειγμα: Να βρείτε την τετραγωνική ρίζα του N (όπου N μεγαλύτερος του 1), χωρίς να χρησιμοποιήσετε την συνάρτηση `sqrt()`

Ποιές είναι οι πιθανές απαντήσεις;

Χρήση του binary search

Το binary search πολλές φορές θα το χρησιμοποιήσουμε όχι με πίνακα, αλλά καθώς ψάχνουμε την απάντηση:

Για παράδειγμα: Να βρείτε την τετραγωνική ρίζα του N (όπου N μεγαλύτερος του 1), χωρίς να χρησιμοποιήσετε την συνάρτηση `sqrt()`

Ποιές είναι οι πιθανές απαντήσεις; Όλοι οι αριθμοί από το 0 μέχρι το N

Πως επιλέγουμε αν πρέπει να επιλέξουμε το αριστερά ή το δεξί μισό;

Χρήση του binary search

Το binary search πολλές φορές θα το χρησιμοποιήσουμε όχι με πίνακα, αλλά καθώς ψάχνουμε την απάντηση:

Για παράδειγμα: Να βρείτε την τετραγωνική ρίζα του N (όπου N μεγαλύτερος του 1), χωρίς να χρησιμοποιήσετε την συνάρτηση `sqrt()`

Ποιές είναι οι πιθανές απαντήσεις; Όλοι οι αριθμοί από το 0 μέχρι το N

Πως επιλέγουμε αν πρέπει να επιλέξουμε το αριστερά ή το δεξί μισό; $X * X < N$



Let's code

Διαγωνισμοί

Σήμερα το απόγευμα (17:35 - 19:35)

Div. 4 στο Codeforces <https://codeforces.com/contests/1850>

- 5-8 προβλήματα
- 2 ώρες
- Βαθμός δυσκολίας:
 - Τα πρώτα ~3 είναι παρόμοιας δυσκολίας με τα πιο εύκολα προβλήματα από τον χθεσινό διαγωνισμό
 - Τα επόμενα γίνονται λίγο πιο δύσκολα

Αν είστε ελεύθεροι και έχετε όρεξη θα το συνιστούσα! (θα δω και εγώ τα προβλήματα, αν θέλετε μπορούμε να τα συζητήσουμε μετά τον διαγωνισμό στο Discord)

Feedback form

Τελειώσαμε με το διδακτικό μέρος του camp! Ευχαριστούμε που ήσασταν εδώ. Θα το εκτιμούσαμε αν μπορείτε να συμπληρώσετε το ερωτηματολόγιο για να μπορέσουμε να το κάνουμε καλύτερο στο μέλλον

<https://forms.gle/uZhUVL7QKDMMcUPn6>

Υλικό

Οι παρουσιάσεις από όλες τις μέρες θα ανεβούν τις επόμενες μέρες στο wiki της ολυμπιάδας, θα στείλουμε το link στο #announcements στο Discord