Binary Indexed Trees

Πρόβλημα:

Έχουμε Ν κουτιά αριθμημένα από το 1 ως το Ν. Θέλουμε να φτιάξουμε ένα πρόγραμμα που να υποστηρίζει τις πράξεις:

- ADD S X: Πρόσθεσε Χ σπίρτα στο κουτί S.
- SUM X Y: Βρες το άθροισμα των σπίρτων που βρίσκονται από το κουτί Χ έως το κουτί Υ. (X <= Y)

Binary Indexed Trees

Προφανής Λύση:

Έχουμε έναν πίνακα N ακεραίων. Στην i-οστη θέση του πίνακα είναι αποθηκευμένο το πλήθος των σπίρτων που βρίσκονται στο i-οστο κουτί.

- ADD S X: Απλά προσθέτουμε τον αριθμό X στην θέση S του πίνακα. Αυτό φυσικά απαιτεί σταθερό χρόνο.
- SUM X Y: Πρέπει να προσθέσουμε όλες τις θέσεις από την X έως και την Y. Στην χειρότερη περίπτωση θα πρέπει να εκτελέσουμε N προσθέσεις. Επομένως η πολυπλοκότητα της SUM είναι O(N).

Binary Indexed Trees

Λύση με μερικά αθροίσματα:

Έχουμε έναν πίνακα Π από Ν ακεραίους. Στην i-οστη θέση του πίνακα είναι αποθηκευμένο το άθροισμα των σπίρτων που βρίσκονται από το 1ο μέχρι και το i-οστο κουτί.

- ADD S X: Θα πρέπει να προσθέσουμε το X σε όλες τις θέσεις από την S έως και την N.

Binary Indexed Trees

Λύση με μερικά αθροίσματα:

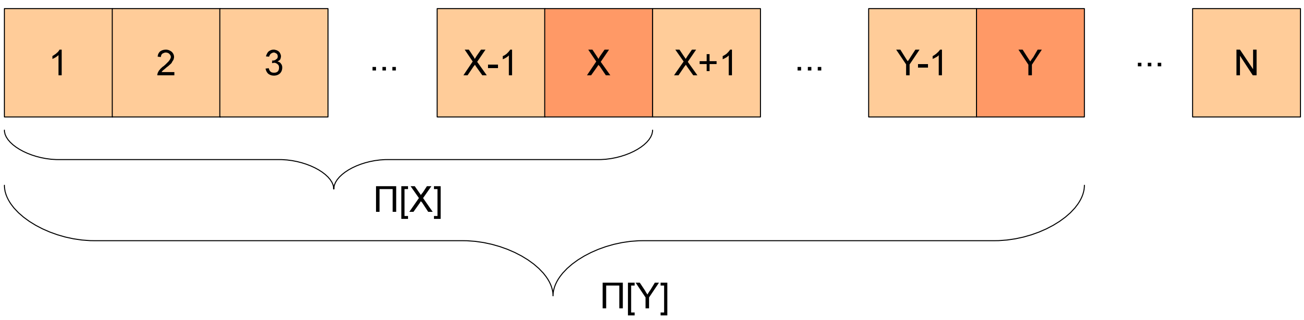
- SUM X Y: Το στοιχείο της θέσης Y (δηλαδή το Π[Y]) περιέχει το άθροισμα:

  κουτί[1] + κουτί[2] + κουτι[3] + κουτι[X-1] + κουτι[X] + … + κουτι[Y]

  Από αυτό θέλουμε να αφαιρέσουμε τα:

  κουτί[1] + κουτί[2] + κουτι[3] + κουτι[X-1]

  Που αντιστοιχούν στο στοιχείο Π[X-1]



Binary Indexed Trees

Λύση με μερικά αθροίσματα:

- Πιο προχωρημένη ιδέα, αλλά:
- ADD: O(N)
- SUM: O(1)
- Δεν κερδίσαμε τίποτα

Binary Indexed Trees

Λύση με buckets:

- Χωρίζουμε τον πίνακα σε O(N) buckets.
- ADD: O(1)
- SUM: O(sqrt(N))

Binary Indexed Trees

Λύσεις:

	ADD	**SUM**
Προφανής Λύση	O(1)	O(N)
Μερικά Αθροίσματα	O(N)	O(1)
Buckets	O(1)	O(sqrt(N))
Binary Indexed Trees	**O(logN)**	**O(logN)**

Binary Indexed Trees

- Είναι μια δομή δεδομένων που μας μας επιτρέπει να διαχειριζόμαστε μερικά αθροίσματα. Συγκεκριμένα υποστηρίζει τις πράξεις:
  - Πρόσθεση ενός αριθμού σε μια συγκεκριμένη θέση.
  - Άθροισμα των αριθμών που βρίσκονται σε ένα συγκεκριμένο διάστημα.

Binary Indexed Trees

- Ουσιαστικά είναι ένας πίνακας αριθμών. Σε κάθε θέση του πίνακα αποθηκεύεται το άθροισμα των σπίρτων που βρίσκονται στο κουτί αυτής της θέσης καθώς και σε κάποιες από τις αμέσως προηγούμενες.

  π.χ.

  - Στην 12η θέση του πίνακα αποθηκεύονται τα σπίρτα που βρίσκονται από το 9ο έως και το 12ο κουτί.
  - Αντίστοιχα στην 14η θέση του πίνακα αποθηκεύεται το πλήθος των σπίρτων που βρίσκονται από το 13ο έως και το 14ο κουτί.

Binary Indexed Trees

- Πως αποφασίζουμε τα σπίρτα πόσων κουτιων θα αποθηκεύουμε σε κάθε θέση;
- Εξαρτάται αποκλειστικά από τον αριθμό της θέσης.
- Μετατροπή στο δυαδικό σύστημα.
- Κάθε θέση αποθηκεύει τόσα κουτιά όσα δείχνει ο μικρότερος άσσος στην δυαδική αναπαράσταση.

Binary Indexed Trees

- 12_{10} = $1\mathbf{1}00_2$. Αν κρατήσουμε μόνο τα ψηφία από τον τελεύταιο άσσο και δεξιά, τότε έχουμε $\mathbf{1}00_2$ = 4_{10} . Άρα στην θέση 12 αποθηκεύουμε τα κουτιά των 4 τελευταίων θέσεων.

  Δηλαδή από το 9ο έως το 12ο κουτί.

Binary Indexed Trees

- 40_{10} = $10\mathbf{1}000_2$. Αν κρατήσουμε μόνο τα ψηφία από τον τελεύταιο άσσο και δεξιά, τότε έχουμε $\mathbf{1}000_2$ = 8_{10} . Άρα στην θέση 40 αποθηκεύουμε τα κουτιά των 8 τελευταίων θέσεων.

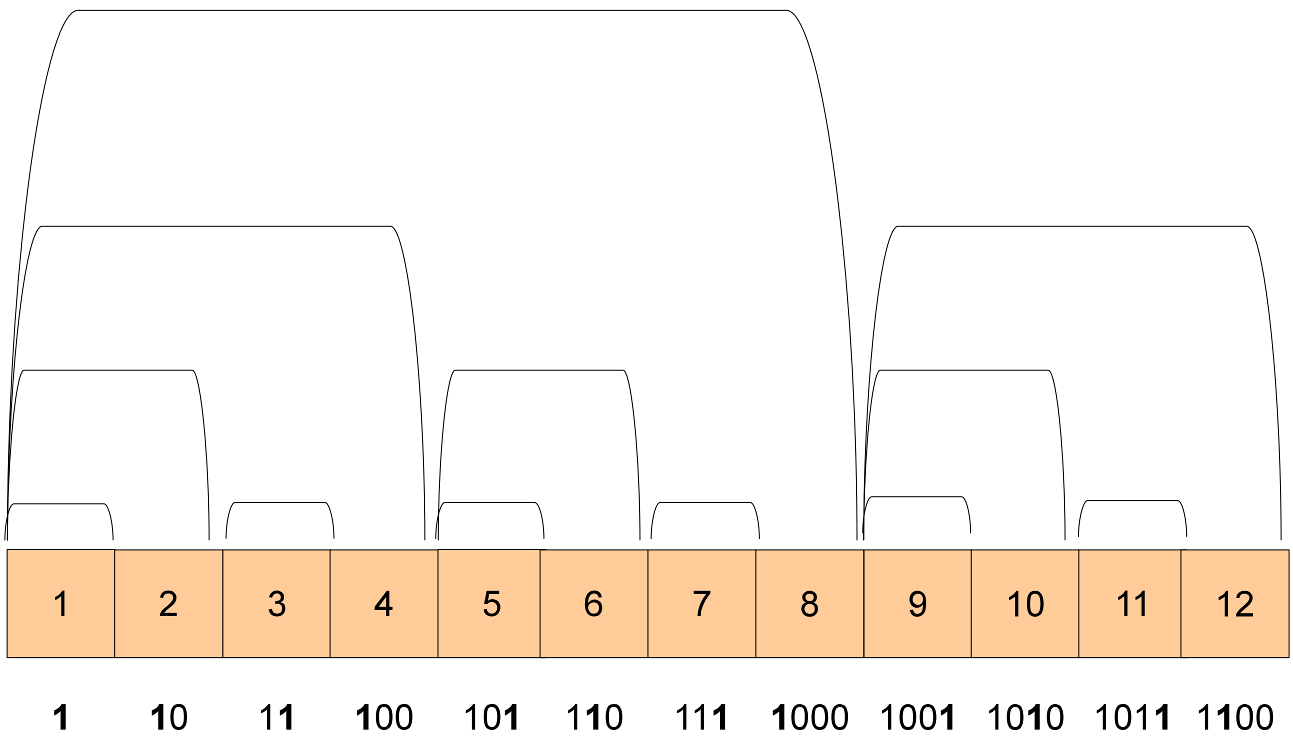
  Δηλαδή από το 33ο έως το 40ο κουτί.

Binary Indexed Trees

- $41_{10} = 10100\mathbf{1}_2$. Αν κρατήσουμε μόνο τα ψηφία από τον τελεύταιο άσσο και δεξιά, τότε έχουμε $\mathbf{1}_2 = 1_{10}$. Άρα στην θέση 40 αποθηκεύουμε το κουτί μόνο της τελευταίας θέσης.

  Δηλαδή μόνο το 41ο κουτί.

Binary Indexed Trees



Binary Indexed Trees

Θέση	Περιέχει
1	1
2	1 ... 2
3	3
4	1 ... 4
5	5
6	5 ... 6
7	7
8	1 ... 8
9	9
10	9 ... 10
11	11
12	9 ... 12
13	13
14	13 ... 14
15	15
16	1 ... 16

Binary Indexed Trees

- Πώς απαντάμε στα ερωτήματα SUM 1 Y;
  - **Βήμα 1**

    Προσθέτουμε την τιμή της θέσης Y.
  - **Βήμα 2**

    Αφαιρούμε τον μικρότερο (δεξιότερο) άσσο από την δυαδική αναπαράσταση του Y. (π.χ. 1010 → 1000)
  - **Βήμα 3**

    Επαναλαμβάνουμε μέχρι το Y να γίνει 0.

Binary Indexed Trees

Θέση:	1	2	3	4	5	6	7	8	9	10	11	12
(Σπίρτα) Σ:	7	0	3	2	0	0	0	4	6	3	2	8
BIT:	7	7	3	12	0	0	0	16	6	9	2	10

Έστω ότι θέλουμε να υπολογίσουμε το άθροισμα:
Σ[1] + Σ[2] + Σ[3] + ... + Σ[11]

Θέλουμε, δηλαδή, να απαντήσουμε στο ερώτημα:
SUM 1 11

Binary Indexed Trees

Θέση:	1	2	3	4	5	6	7	8	9	10	11	12
(Σπίρτα) Σ:	7	0	3	2	0	0	0	4	6	3	2	8
BIT:	7	7	3	12	0	0	0	16	6	9	2	10

Αρχίζουμε από την θέση 11 και προσθέτουμε το BIT[11] = 2 στο άθροισμα.

SUM = 2

Binary Indexed Trees

Θέση:	1	2	3	4	5	6	7	8	9	10	11	12
(Σπίρτα) Σ:	7	0	3	2	0	0	0	4	6	3	2	8
BIT:	7	7	3	12	0	0	0	16	6	9	2	10

Η δυαδική αναπαράσταση του 11 είναι 101**1.** Αφαιρούμε τον πρώτο άσσο, ενώ τα υπόλοιπα ψηφία παραμένουν ίδια

```
  1011
- 0001
------
  1010   = 10₁₀
```

SUM = 2

Binary Indexed Trees

Θέση:	1	2	3	4	5	6	7	8	9	10	11	12
(Σπίρτα) Σ:	7	0	3	2	0	0	0	4	6	3	2	8
BIT:	7	7	3	12	0	0	0	16	6	9	2	10

Πλέον βρισκόμαστε στην θέση 10, οπότε προσθέτουμε την τιμή του BIT[10] στο άθροισμα.

SUM = 2 **+ 9** = 11

Binary Indexed Trees

Θέση:	1	2	3	4	5	6	7	8	9	10	11	12
(Σπίρτα) Σ:	7	0	3	2	0	0	0	4	6	3	2	8
BIT:	7	7	3	12	0	0	0	16	6	9	2	10

Η δυαδική αναπαράσταση του 10 είναι 10**10.** Αφαιρούμε τον πρώτο άσσο, ενώ τα υπόλοιπα ψηφία παραμένουν ίδια

```
  1010
- 0010
------
  1000   = 8₁₀
```

SUM = 11

Binary Indexed Trees

Θέση:	1	2	3	4	5	6	7	8	9	10	11	12
(Σπίρτα) Σ:	7	0	3	2	0	0	0	4	6	3	2	8
BIT:	7	7	3	12	0	0	0	16	6	9	2	10

Τώρα είμαστε στην θέση 8, οπότε προσθέτουμε την τιμή του BIT[8] στο άθροισμα.

SUM =11 **+ 16** = 27

Binary Indexed Trees

Θέση:	1	2	3	4	5	6	7	8	9	10	11	12
(Σπίρτα) Σ:	7	0	3	2	0	0	0	4	6	3	2	8
BIT:	7	7	3	12	0	0	0	16	6	9	2	10

Η δυαδική αναπαράσταση του 8 είναι 1000. Αφαιρούμε τον πρώτο άσσο, ενώ τα υπόλοιπα ψηφία παραμένουν ίδια

```
  1000
- 1000
------
  0000
```

$= 0_{10}$

Άρα η τελική απάντηση είναι SUM = 27.

Binary Indexed Trees

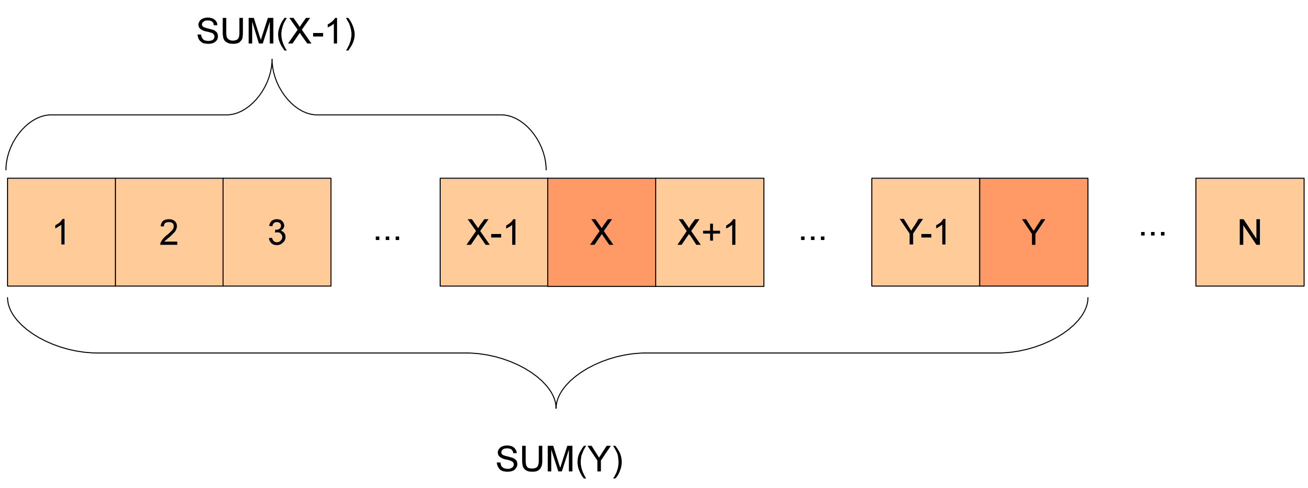
- **Ερώτηση:** Γιατί είναι O(logN);
  - Ξεκινώντας από το Y, θα κάνουμε τόσες προσθέσεις όσοι είναι οι άσσοι του Y (αφού σε κάθε βήμα θα κάνουμε μια πρόσθεση και θα σβήνουμε έναν άσσο από την δυαδική αναπαράστασή του).
  - Η δυαδική αναπαράσταση κάθε αριθμού N αποτελείται από $\log_2 N + 1$ ψηφία.
  - Θα εκτελέστουν το πολύ $\log_2 N + 1 = O(\log N)$ προσθέσεις.

Binary Indexed Trees

- Πως απαντάμε στο ερώτημα SUM X Y;

Το σπάμε σε δύο:

SUM X Y = SUM 1 Y – SUM 1 X-1

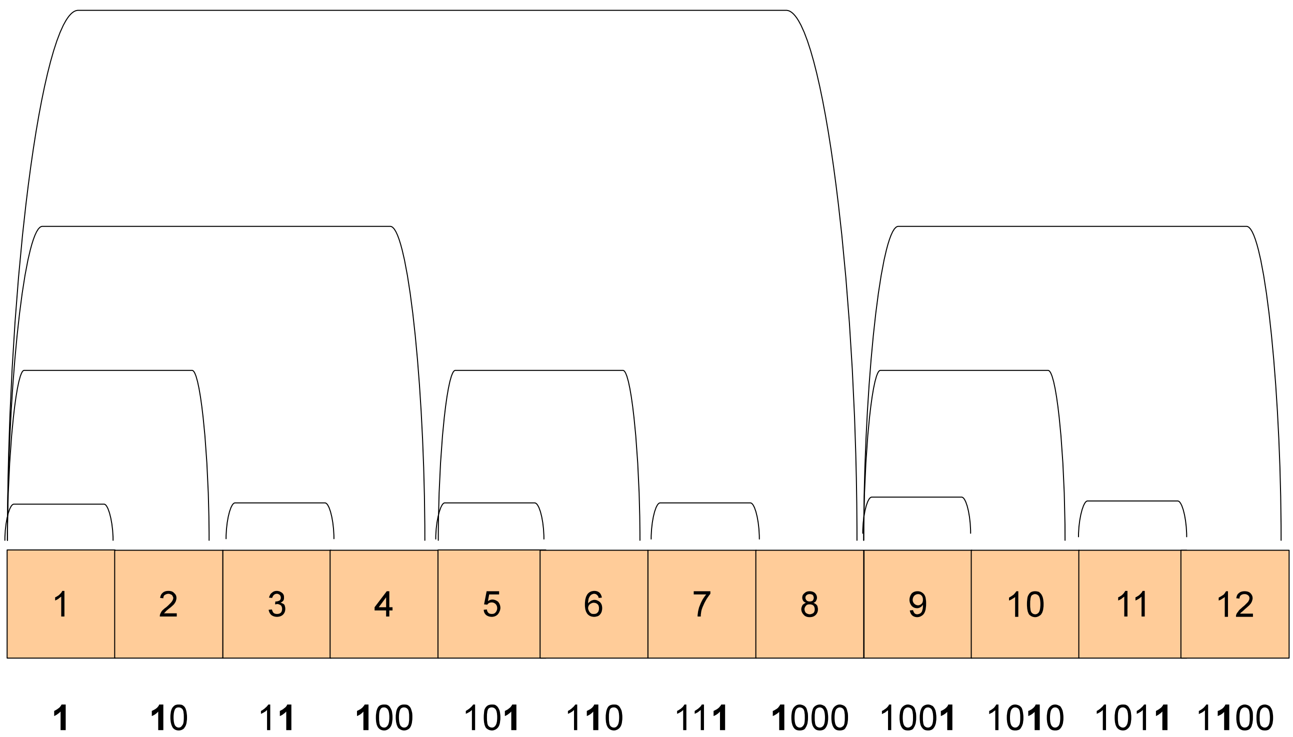


Binary Indexed Trees

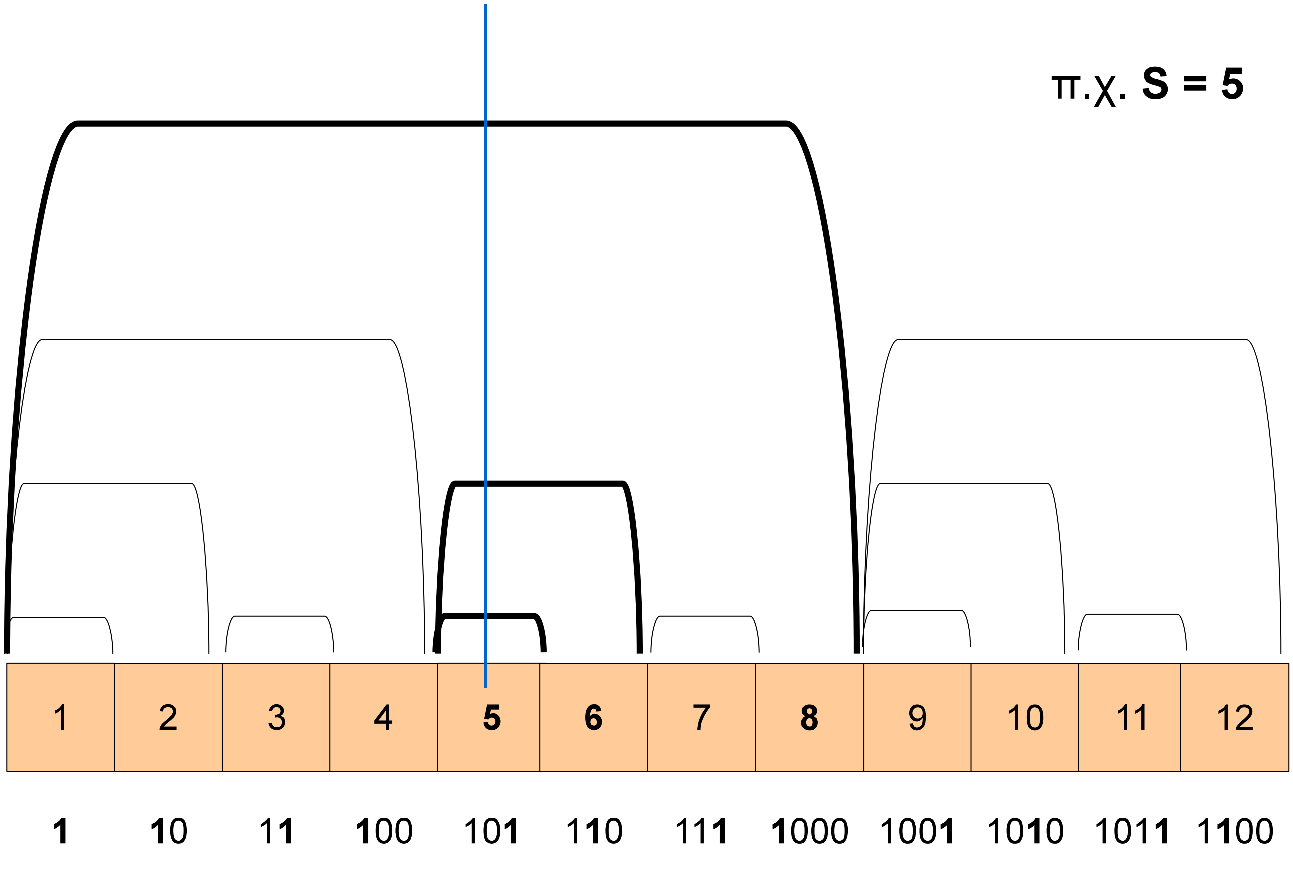
- Πώς απαντάμε στα ερωτήματα ADD S X;
  - Πρέπει να αλλάξουμε την τιμή όλων των θέσεων που περιέχουν την S.
  - Αντίστροφη Διαδικασία

Binary Indexed Trees

π.χ. **S = 5**

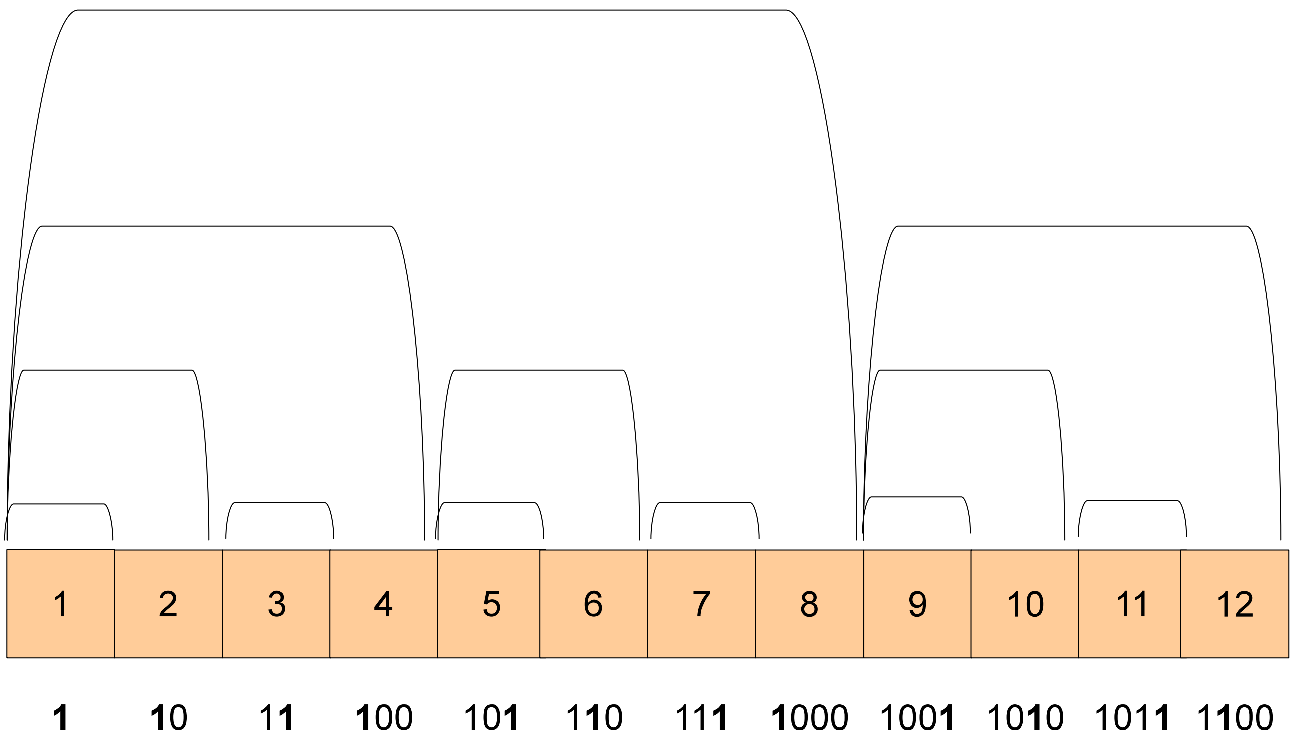


Binary Indexed Trees

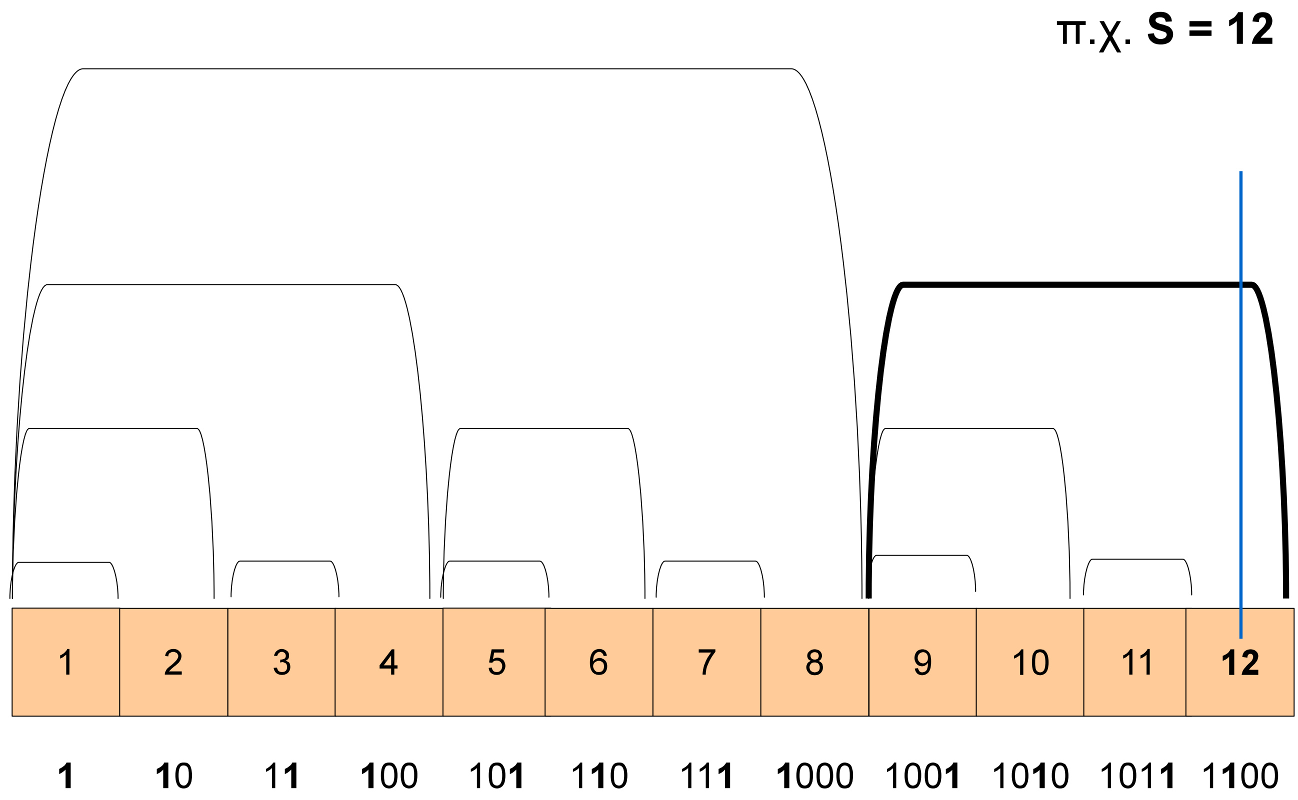


Binary Indexed Trees

π.χ. **S = 12**



Binary Indexed Trees



Binary Indexed Trees

- **Βήμα 1**

  Προσθέτουμε τον αριθμό X στην θέση S.

- **Βήμα 2**

  Προσθέτουμε τον μικρότερο (δεξιότερο) άσσο στην την δυαδική αναπαράσταση του S.

  (π.χ. 10**1**0 → 1100)

  Βήμα 3

  Επαναλαμβάνουμε μέχρι το S να ξεπεράσει το μέγεθος του πίνακα.

Binary Indexed Trees

+3

Θέση:	1	2	3	4	5	6	7	8	9	10	11	12
(Σπίρτα) Σ:	7	0	3	2	0	0	0	4	6	3	2	8
BIT:	7	7	3	12	0	0	0	16	6	9	2	10

Έστω ότι θέλουμε να προσθέσουμε 3 σπίρτα στην θέση 5:

Θέλουμε, δηλαδή, να εκτελέσουμε την πράξη:
ADD 5 3

Binary Indexed Trees

Θέση:	1	2	3	4	5	6	7	8	9	10	11	12
(Σπίρτα) Σ:	7	0	3	2	3	0	0	4	6	3	2	8
BIT:	7	7	3	12	**3**	0	0	16	6	9	2	10

Αρχικά προσθέτουμε το 3 στην θέση 5 του πίνακα BIT.

Binary Indexed Trees

Θέση:	1	2	3	4	5	6	7	8	9	10	11	12
(Σπίρτα) Σ:	7	0	3	2	3	0	0	4	6	3	2	8
BIT:	7	7	3	12	**3**	0	0	16	6	9	2	10

Η δυαδική αναπαράσταση του 5 είναι 10**1.** Προσθέτουμε, Λοιπόν, πρώτο άσσο:

```
  101
+ 001
-----
  110
```

$= 6_{10}$

Binary Indexed Trees

Θέση:	1	2	3	4	5	6	7	8	9	10	11	12
(Σπίρτα) Σ:	7	0	3	2	3	0	0	4	6	3	2	8
BIT:	7	7	3	12	**3**	**3**	0	16	6	9	2	10

Άρα η επόμενη θέση του πίνακα BIT στην οποία πρέπει να προστεθεί το 3 είναι η θέση 6.

Binary Indexed Trees

Θέση:	1	2	3	4	5	6	7	8	9	10	11	12
(Σπίρτα) Σ:	7	0	3	2	3	0	0	4	6	3	2	8
BIT:	7	7	3	12	**3**	**3**	0	16	6	9	2	10

Η δυαδική αναπαράσταση του 6 είναι 1**1**0**.** Προσθέτουμε, ξανά πρώτο άσσο:

```
  110
+ 010
-----
 1000   = 8₁₀
```

Binary Indexed Trees

Θέση:	1	2	3	4	5	6	7	8	9	10	11	12
(Σπίρτα) Σ:	7	0	3	2	3	0	0	4	6	3	2	8
BIT:	7	7	3	12	**3**	**3**	0	**19**	6	9	2	10

Έτσι το προσθέτουμε το 3 στο BIT[8] και γίνεται:

BIT[8] = 16 **+ 3 = 19**

Binary Indexed Trees

Θέση:	1	2	3	4	5	6	7	8	9	10	11	12
(Σπίρτα) Σ:	7	0	3	2	3	0	0	4	6	3	2	8
BIT:	7	7	3	12	**3**	**3**	0	**19**	6	9	2	10

Η δυαδική αναπαράσταση του 8 είναι **1**000**.** Προσθέτοντας τον πρώτο (και μοναδικό) άσσο παίρνουμε:

```
   1000
 + 1000
 ------
  10000
```

$= 16_{10}$

Binary Indexed Trees

Θέση:	1	2	3	4	5	6	7	8	9	10	11	12
(Σπίρτα) Σ:	7	0	3	2	3	0	0	4	6	3	2	8
BIT:	7	7	3	12	**3**	**3**	0	**19**	6	9	2	10

Το 16 όμως είναι μεγαλύτερο από το μέγεθος του πίνακα BIT, άρα η διαδικασία τερματίζει εδώ.

Binary Indexed Trees: Υλοποίηση της sum

```
int sum(int Y) {
    int answer = 0; /* μεταβλητή που κρατάει το ζητούμενο άθροισμα */
    while (Y > 0) {
        answer += BIT[Y];
        Y -= (Y & -Y);
    }
    return answer;
}
```

Binary Indexed Trees: Υλοποίηση της sum

```
int sum(int Y) {
    int answer = 0; /* μεταβλητή που κρατάει το ζητούμενο άθροισμα */
    while (Y > 0) {
        answer += BIT[Y];
        Y -= (Y & -Y);
    }
    return answer;
}
```

Τρικ

Binary Indexed Trees: Υλοποίηση της add

```
void add(int pos, int num) {
    while (pos <= N) {
        BIT[pos] += num;
        pos += (pos & -pos);
    }
}
```