**AIB (Arbori Indexați Binar)**

Cu ajutorul lor se pot rezolva acele probleme la care este nevoie de sume parțiale (de exemplu avem un vector a și avem nevoie foarte des de calculat sume parțiale între diverși indici) însă în mod FOARTE frecvent se modifică elemente în șir, modificări care ar produce recalcularea pentru foarte multe elemente din vectorul cu sume parțiale (prefixe).

Arborii indexați binar de fapt se memorează sub forma unui vector obișnuit, doar că fiecare element al vectorului va conține o sumă parțială, a unei bucăți din vectorul dat într-un mod în care sumele parțiale memorate în arbore, puse cap la cap, permit foarte rapid atât reconstituirea unei sume de tip prefix, cât și recalcularea rapidă a tuturor sumelor parțiale atunci când se modifică un element.

Ceea ce face ca acest vector să fie arbore este legătura de tip tată a fiecărui indice, către un alt element.

***Regula după care se stabilește care este tatăl unui nod i este următoarea:***

***- se face reprezentarea binară a lui i***

***- tatăl este dat din reprezentarea binară în care ultimul bit (1) din aceasta este resetat (acel bit devine 0).***

Exemplu: indicele 42 - să vedem ce tată are.

reprezentarea binară a lui 42 este: 101010.

Resetarea ultimului bit egal cu 1 ne dă: 101000 = 40.

SO, ta'su lu' 42 este 40.

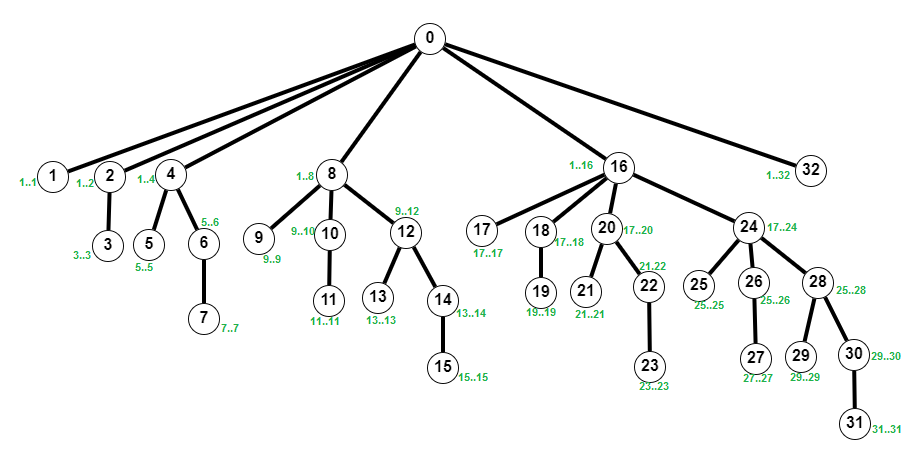
Cine este ta'su lu' 40: (40=101000, ta'su = 100000 = 32)

***Regula formării arborelui indexat binar:***

***un element memorat la indicele i va reține suma parțială a elementelor din șirul inițial aflat între indicii t[i]+1 și i***

Hai să facem o reprezentare a acestui arbore. Vom lua și un nod la indicele 0, doar pentru faptul că tații tuturor puterilor curate ale lui 2 (1, 2, 4, 8, 16, 32, 64,..) conform regulii dau 0.

Elementul respectiv nu va conține însă nicio sumă, dar respectă regula t[i]+1..i deci fiecare nod care are indice putere a lui 2 va memora suma tuturor valorilor din șirul inițial dintre 1 și acea putere.



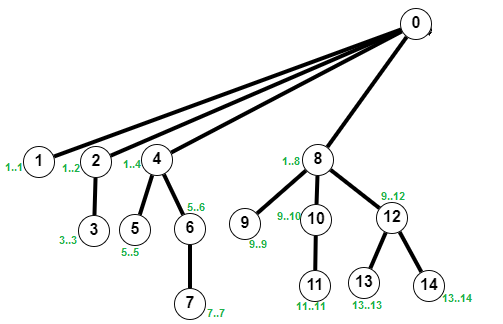
Astfel, dacă șirul nostru este a (cu 32 de elemente) arborele de mai sus ne dă schema după care vectorul cu sume parțiale va fi format.

Deci, numărul din nod (cel cu negru) reprezintă indicele vectorului AIB, iar cele două numere verzi reprezintă indicii secvenței din șirul dat pentru care s-a calculat suma parțială.

Exemplu: iată pentru un vector a cu 14 elemente cum arată vectorul AIB:

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| i | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 |
| a[i] | 5 | 9 | 1 | 17 | 11 | 18 | 7 | 13 | 15 | 5 | 2 | 6 | 8 | 1 |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| i | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 |
| aib[i] | 5 | 14 | 1 | 32 | 11 | 29 | 7 | 81 | 15 | 20 | 2 | 28 | 8 | 9 |



deci în aib[12] se reține, conform arborelui din stânga, suma valorilor din vectorul dat dintre indicii scriși în dreptul lui 12 cu verde, deci de la 9 la 12, adică 15+5+2+6=28

Pe baza lui aib se poate calcula în timp logaritmic foarte bun orice sumă a unui prefix a șirului: pur și simplu mergem în arbore pe lanțul tată și adunăm toate acele valori.

De exemplu, dacă ne trebuie suma primelor 10 valori, adunăm aib[\*] de la toate nodurile cu indici în arborele de mai sus pe traseul dintre 10 și rădăcină.

Adică doar aib[10]+aib[8]=20+81=101

SAU suma tuturor dintre indicii 1 și 14 = aib[14]+aib[12]+aib[8]=9+28+81= 118

Cum determinăm FOARTE rapid indicele nodului tată al unui nod **x**?

ne bazăm pe faptul că, dacă facem reprezentarea binară a unui număr și scădem 1, ultimul 1 din reprezentare devine 0, iar zerourile de după el devin 1.

Ex:

101101000- 101101000

1 101100111

========= =========

101100111 101100000

Făcând operația **&** pe biți între **x** și **x-1**

SO, ta'su lui x se face prin banala atribuire:

**tasu = (x&(x-1));**

**Modificarea unui element în șirul dat și refelxia sa asupra AIB**

Modificarea unui element al vectorului, de la un indice x va modifica TOATE sumele din AIB care includ acel indice x.

Plecând din nou de la definiția AIB-ului, mai precis a relației TATĂ-FIU și a sumelor care de fapt sunt cuprinse între TATĂ+1 și FIU, putem analiza câteva exemple și deduce regula:

x=1001010 = 74

1001100 = 72 .. 76

1001000 = 72 ... 64

1010000 = 64...80

1100000 = 64..96

10000000 = 1...128

cel mai limpede este că 74 este cuprins în nodul 74 (orice valoare x este cuprinsă în AIB[x]).

mai precis AIB[74] conține suma parțială 72..74

Ca un alt nod din AIB să conțină valoarea de la indicele 74, e clar că indicele trebuie să fie mai mare decât 74.

Noul indice se obține adunând x cu numărul format din ultimul bit egal cu 1 și restul cifrelor egale cu 0, la x.

Operația se repetă până când numărul obținut este mai mare decât n = nr. de valori din șir.

Exemplu:

74=1001100

îl adunăm mai întâi cu 100

1001010+

10

=======

1001100 = 76

100

=======

1010000 = 80

10000

=======

1100000 = 96

100000

=======

10000000 = 128

Regula de calcul este din nou una simplă:

noul indice = (x&(-x)+x)