

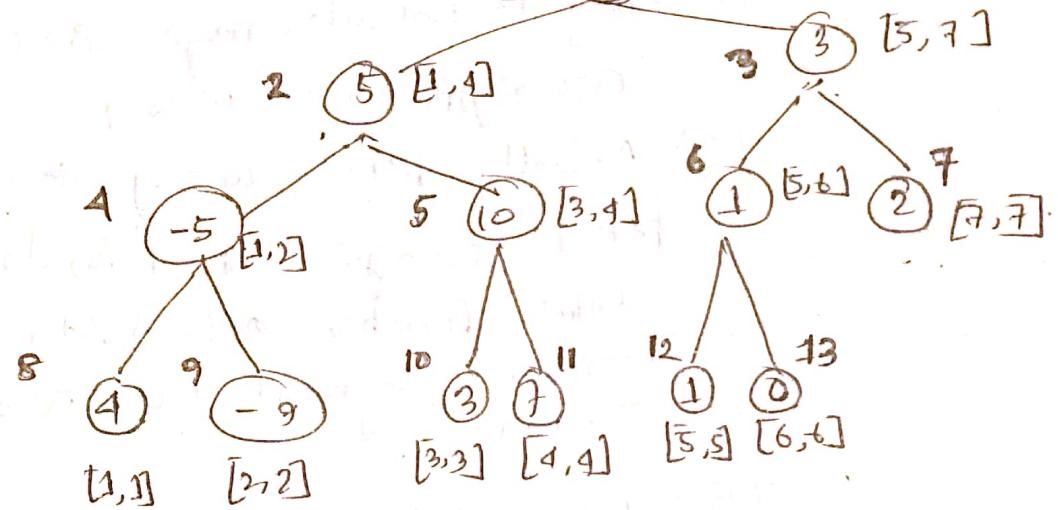
Segment Tree

- ① Array to particular range को कैसे बनाएं, यह,
इसकी त्रिकोणीयता क्या है?
 - ② Actually Segment Query, क्या इसका प्रयोग करके
किसी range के total sum का ज्ञान किया जा सकता है।
अगली function का लिखें। जैसा कि
आगला segment tree का बहुत ज्ञान
चाहिए।
 - ③ Range Minimum Query type का problem
क्या उसके लिए अल्गोरिदम बनाएं, इसके
लिए loop का उपयोग करके इसका विवरण
TLE का कारण क्या है? क्योंकि यह का
कारण यह है कि जब उसका लिए जाना चाहिए तो यह
time limit को काट लेता है जो 20 ms है।

સ્વાચ્છ એવી માર્ગ નિર્દેશિત અનાજો વળ ના;

ଏହା କିମ୍ବା 4-9 3 7 1 0 2
 ଏହା କିମ୍ବା 4-9 3 7 1 0 2
 ଏହା କିମ୍ବା 4-9 3 7 1 0 2
 ଏହା କିମ୍ବା 4-9 3 7 1 0 2

1	2	3	4	5	6	7
4	-9	3	7	1	0	2

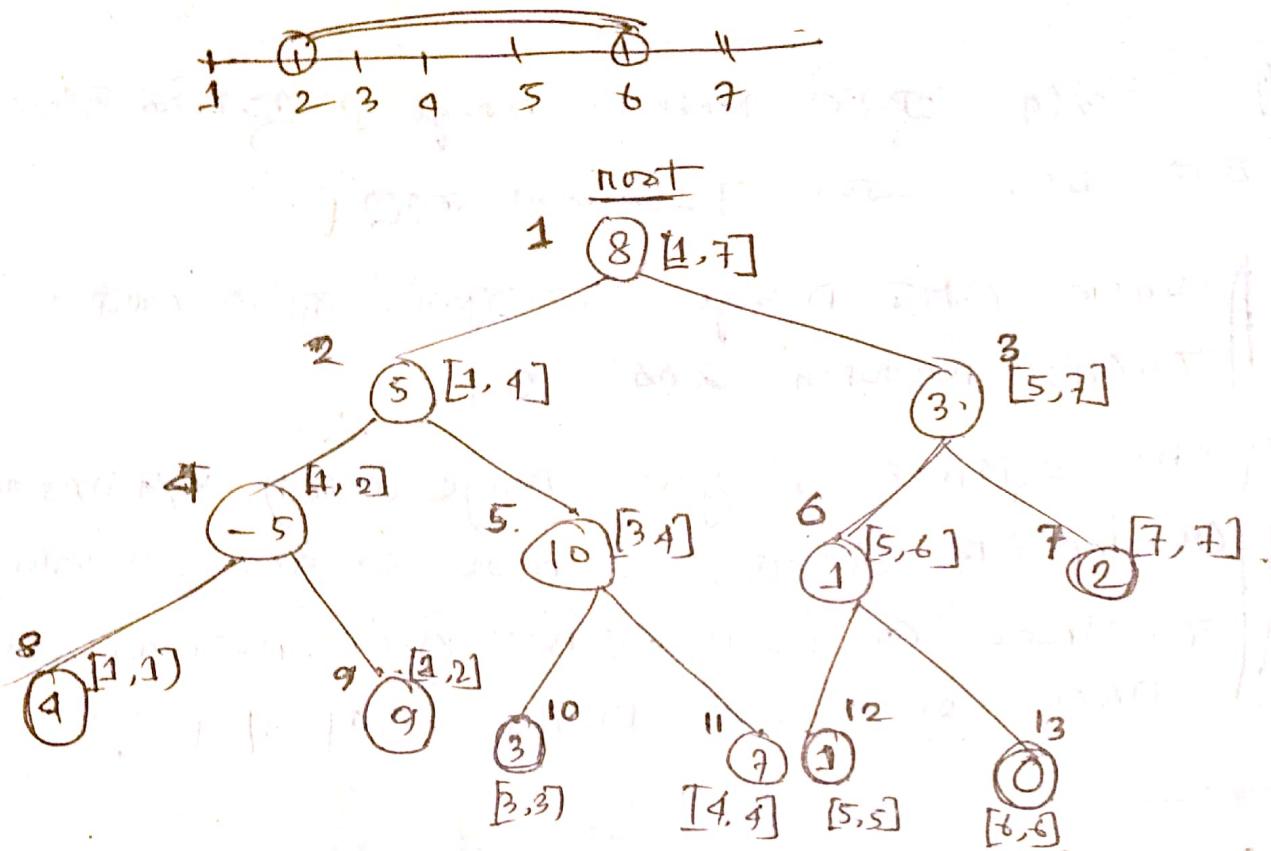


BST का गणना पूँछा - 27, मानक पूँछा index
 अद्वितीय एवं द्वितीय अन्य (अ) नहीं राखा 27।
 रजत, अक्षर, द्वितीय लड़्डी जो Child भवति, उसे
 left child एवं उसे right child.

$$\text{left} = \text{node} * 2;$$

$$\text{right} = \text{node} * 2 + 1$$

Ques 2. for 6 marks अन्य index वे 27 का क्या बनाएँ।

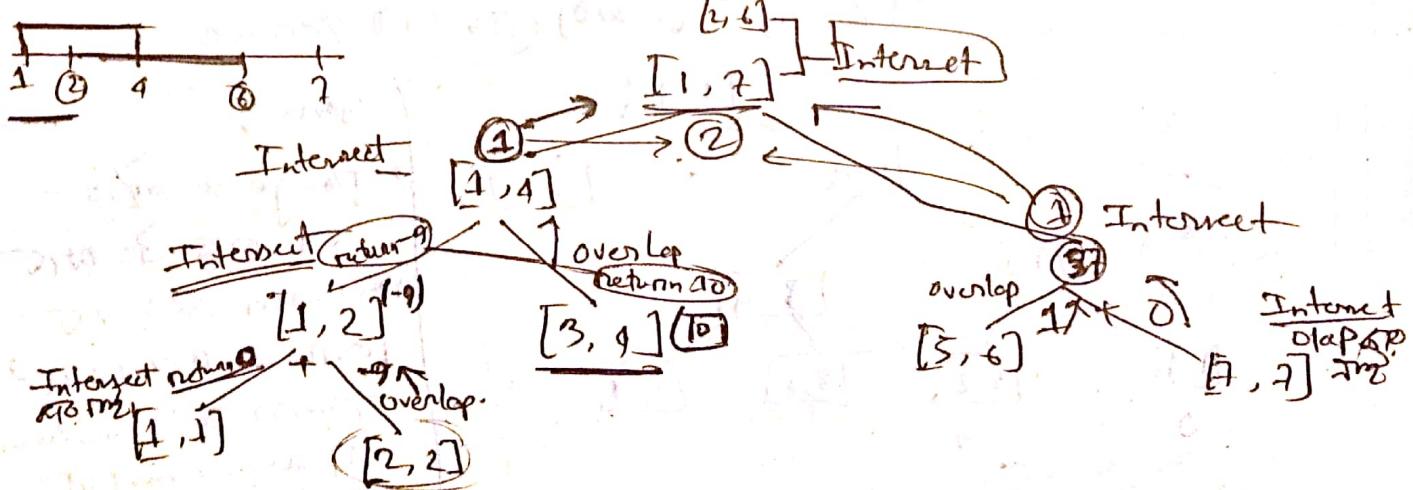


2. Curr. Range Index & Current Range

3. No node covers current root curr, root curr range

$(1, 7)$ intersects range $(2, 4)$ or, two ranges

- range 3. This will merge them



काम करने के लिए Note वो range के सुधारित बहाव
पर भर दें तो Internet करें।

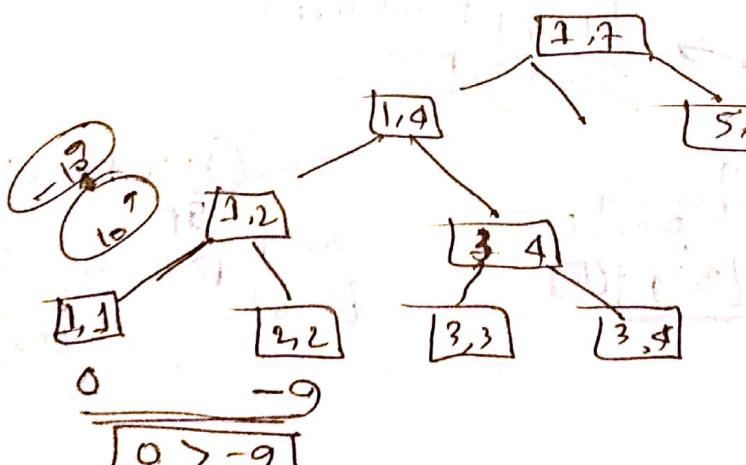
अब यह range के सुधारित गणीय होता है।
रसम्भव return देता है 0, 1, 2

आप इसके लिए given range के लिए प्राप्त
Overlap का अंतर्वर्ती वो note के मध्य के value
के free के fixed घटाने के लिए return देता है।
वास्तव में यह उसे Divide करता है।

यह Query का काम करने के लिए दो तरीके हैं।
एक अलग अलग फ़ाइल के लिए, एक और एक

Return करना वही करना चाहिए। ऐसा करना
leaf level का Example देता है। आइए:

(2-6) जो यही (कड़ी) कर देता है

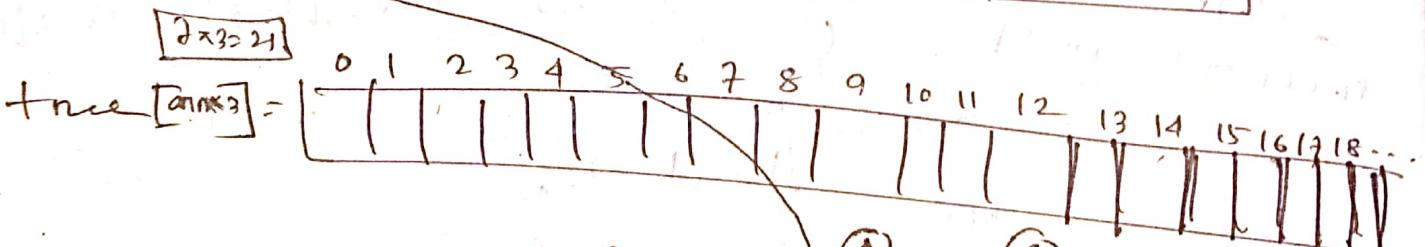
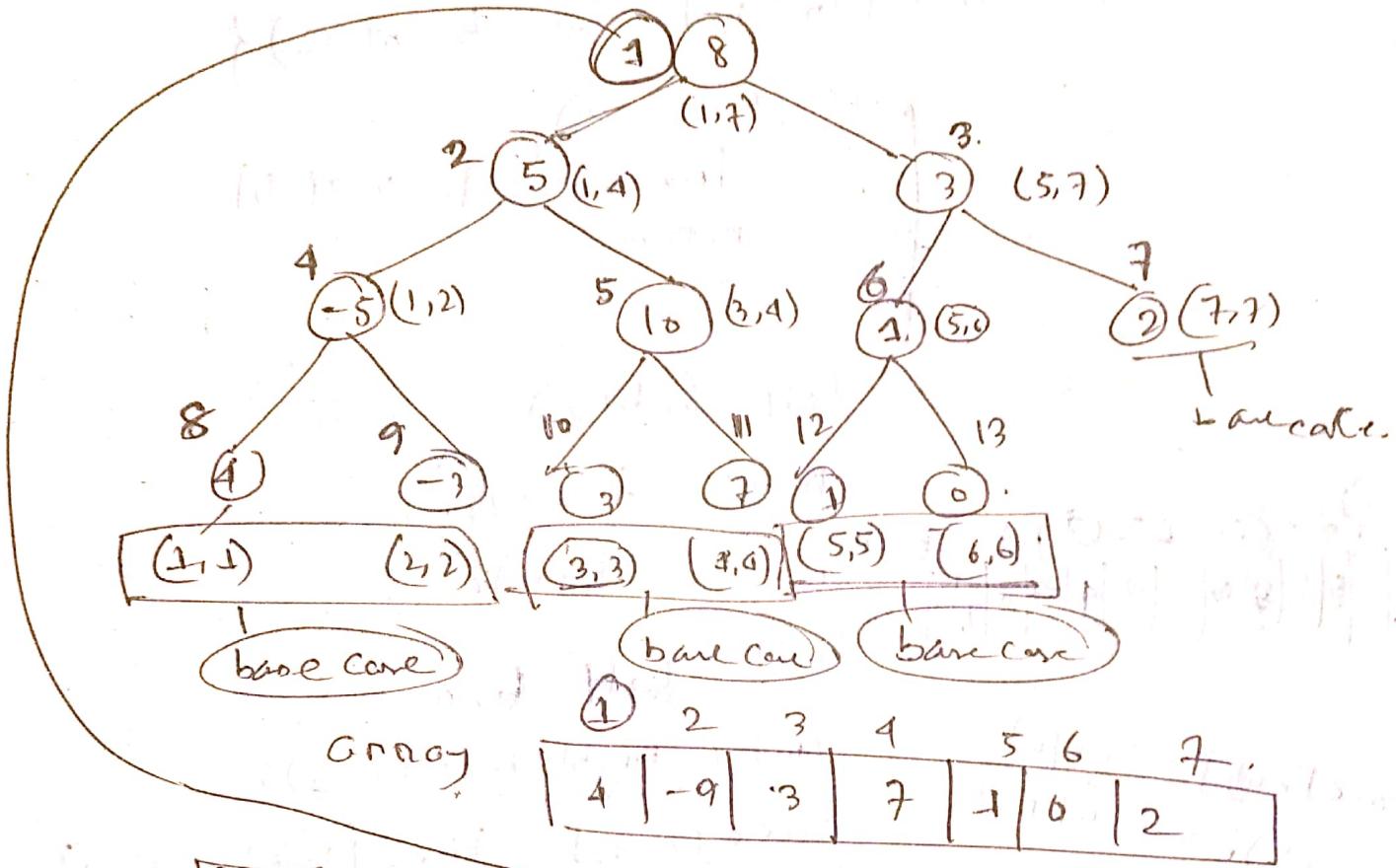


जबकि;
Range के लिए
उसे उसे करना चाहिए
Return करना 20
सिर्फ़ 1/1000 का लिए
ऐसा करना चाहिए।
उसे करना चाहिए।
उसे करना चाहिए।

यदि यही return करता था, तो उसे 2000
तो, 0 > -9 का लिए 1000 का

Implementation of Segment tree

base case
left = right



void func (int node, int left, int right)

if (b == e)

return;

left = node * 2 + 1

right = node * 2 + 2

mid = (left + right) / 2

$\text{arr} = \begin{bmatrix} 1 & 2 & 3 & 4 & 5 & 6 & 7 \\ 4 & 9 & 3 & 17 & 7 & 0 & 2 \end{bmatrix}$

$\text{tree} = [\text{arr} \times 3] = \begin{bmatrix} 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 & 11 & 12 & 13 & 14 & 15 & 16 & 17 \end{bmatrix}$

void init (int node, int b, int e) {

if ($b == e$) {

tree[node] = arr[b]

return;

left = (node * 2) =

right = (node * 2) + 1

mid = $(b+e)/2$;

init (left, b, mid);

init (right, mid+1, e);

tree[node] = tree[left] + tree[right];

tree[node] = arr[b].

⑧ = arr[1]

tree[9] = arr[2]=

tree[1] = arr[3]=

tree[2] = arr[4]=

tree[3] = arr[5]=

tree[4] = arr[6]=

tree[5] = arr[7]=

tree[6] = arr[8]=

tree[7] = arr[9]=

tree[8] = arr[10]=

tree[9] = arr[11]=

tree[10] = arr[12]=

tree[11] = arr[13]=

tree[12] = arr[14]=

tree[13] = arr[15]=

tree[14] = arr[16]=

tree[15] = arr[17]=

tree[16] = arr[18]=

tree[17] = arr[19]=

tree[18] = arr[20]=

tree[19] = arr[21]=

tree[20] = arr[22]=

tree[21] = arr[23]=

tree[22] = arr[24]=

tree[23] = arr[25]=

tree[24] = arr[26]=

tree[25] = arr[27]=

tree[26] = arr[28]=

tree[27] = arr[29]=

tree[28] = arr[30]=

tree[29] = arr[31]=

tree[30] = arr[32]=

tree[31] = arr[33]=

tree[32] = arr[34]=

tree[33] = arr[35]=

tree[34] = arr[36]=

tree[35] = arr[37]=

tree[36] = arr[38]=

tree[37] = arr[39]=

tree[38] = arr[40]=

tree[39] = arr[41]=

tree[40] = arr[42]=

tree[41] = arr[43]=

tree[42] = arr[44]=

tree[43] = arr[45]=

tree[44] = arr[46]=

tree[45] = arr[47]=

tree[46] = arr[48]=

tree[47] = arr[49]=

tree[48] = arr[50]=

tree[49] = arr[51]=

tree[50] = arr[52]=

tree[51] = arr[53]=

tree[52] = arr[54]=

tree[53] = arr[55]=

tree[54] = arr[56]=

tree[55] = arr[57]=

tree[56] = arr[58]=

tree[57] = arr[59]=

tree[58] = arr[60]=

tree[59] = arr[61]=

tree[60] = arr[62]=

tree[61] = arr[63]=

tree[62] = arr[64]=

tree[63] = arr[65]=

tree[64] = arr[66]=

tree[65] = arr[67]=

tree[66] = arr[68]=

tree[67] = arr[69]=

tree[68] = arr[70]=

tree[69] = arr[71]=

tree[70] = arr[72]=

tree[71] = arr[73]=

tree[72] = arr[74]=

tree[73] = arr[75]=

tree[74] = arr[76]=

tree[75] = arr[77]=

tree[76] = arr[78]=

tree[77] = arr[79]=

tree[78] = arr[80]=

tree[79] = arr[81]=

tree[80] = arr[82]=

tree[81] = arr[83]=

tree[82] = arr[84]=

tree[83] = arr[85]=

tree[84] = arr[86]=

tree[85] = arr[87]=

tree[86] = arr[88]=

tree[87] = arr[89]=

tree[88] = arr[90]=

tree[89] = arr[91]=

tree[90] = arr[92]=

tree[91] = arr[93]=

tree[92] = arr[94]=

tree[93] = arr[95]=

tree[94] = arr[96]=

tree[95] = arr[97]=

tree[96] = arr[98]=

tree[97] = arr[99]=

tree[98] = arr[100]=

tree[99] = arr[101]=

tree[100] = arr[102]=

tree[101] = arr[103]=

tree[102] = arr[104]=

tree[103] = arr[105]=

tree[104] = arr[106]=

tree[105] = arr[107]=

tree[106] = arr[108]=

tree[107] = arr[109]=

tree[108] = arr[110]=

tree[109] = arr[111]=

tree[110] = arr[112]=

tree[111] = arr[113]=

tree[112] = arr[114]=

tree[113] = arr[115]=

tree[114] = arr[116]=

tree[115] = arr[117]=

tree[116] = arr[118]=

tree[117] = arr[119]=

tree[118] = arr[120]=

tree[119] = arr[121]=

tree[120] = arr[122]=

tree[121] = arr[123]=

tree[122] = arr[124]=

tree[123] = arr[125]=

tree[124] = arr[126]=

tree[125] = arr[127]=

tree[126] = arr[128]=

tree[127] = arr[129]=

tree[128] = arr[130]=

tree[129] = arr[131]=

tree[130] = arr[132]=

tree[131] = arr[133]=

tree[132] = arr[134]=

tree[133] = arr[135]=

tree[134] = arr[136]=

tree[135] = arr[137]=

tree[136] = arr[138]=

tree[137] = arr[139]=

tree[138] = arr[140]=

tree[139] = arr[141]=

tree[140] = arr[142]=

tree[141] = arr[143]=

tree[142] = arr[144]=

tree[143] = arr[145]=

tree[144] = arr[146]=

tree[145] = arr[147]=

tree[146] = arr[148]=

tree[147] = arr[149]=

tree[148] = arr[150]=

tree[149] = arr[151]=

tree[150] = arr[152]=

tree[151] = arr[153]=

tree[152] = arr[154]=

tree[153] = arr[155]=

tree[154] = arr[156]=

tree[155] = arr[157]=

tree[156] = arr[158]=

tree[157] = arr[159]=

tree[158] = arr[160]=

tree[159] = arr[161]=

tree[160] = arr[162]=

tree[161] = arr[163]=

tree[162] = arr[164]=

tree[163] = arr[165]=

tree[164] = arr[166]=

tree[165] = arr[167]=

tree[166] = arr[168]=

tree[167] = arr[169]=

tree[168] = arr[170]=

tree[169] = arr[171]=

tree[170] = arr[172]=

tree[171] = arr[173]=

tree[172] = arr[174]=

tree[173] = arr[175]=

tree[174] = arr[176]=

tree[175] = arr[177]=

tree[176] = arr[178]=

tree[177] = arr[179]=

tree[178] = arr[180]=

tree[179] = arr[181]=

tree[180] = arr[182]=

tree[181] = arr[183]=

tree[182] = arr[184]=

tree[183] = arr[185]=

tree[184] = arr[186]=

tree[185] = arr[187]=

tree[186] = arr[188]=

tree[187] = arr[189]=

tree[188] = arr[190]=

tree[189] = arr[191]=

tree[190] = arr[192]=

tree[191] = arr[193]=

tree[192] = arr[194]=

tree[193] = arr[195]=

tree[194] = arr[196]=

tree[195] = arr[197]=

tree[196] = arr[198]=

tree[197] = arr[199]=

tree[198] = arr[200]=

tree[199] = arr[201]=

tree[200] = arr[202]=

tree[201] = arr[203]=

tree[202] = arr[204]=

tree[203] = arr[205]=

tree[204] = arr[206]=

tree[205] = arr[207]=

tree[206] = arr[208]=

tree[207] = arr[209]=

tree[208] = arr[210]=

tree[209] = arr[211]=

tree[210] = arr[212]=

tree[211] = arr[213]=

tree[212] = arr[214]=

tree[213] = arr[215]=

tree[214] = arr[216]=

tree[215] = arr[217]=

tree[216] = arr[218]=

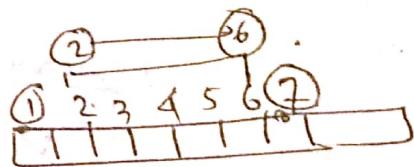
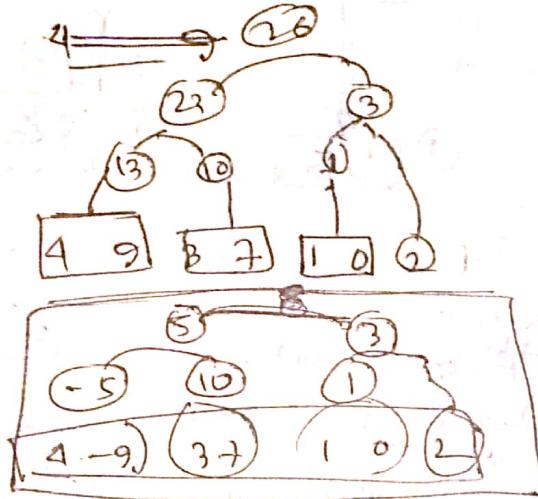
tree[217] = arr[219]=

tree[218] = arr[220]=

tree[219] = arr[221]=

tree[220] = arr[222]=

tree



କୌଣସି tree କେବଳ ଏକ ଫେଲ୍, କିନ୍ତୁ tree
କେବଳ କାହାର କାମରେ ଯାଏନ୍ତି ଏହି tree କିମ୍ବା
Query କାମ, ଏଥିରେ ପରିଚାରକ (MST) କିମ୍ବା

i) କେବଳ value (range) (କୌଣସି ୨୩୦୩୦)
କିମ୍ବା ଏହି ପରିମାଣ କାମକରଣ କରିବାକୁ ବିଶେଷ.

କୌଣସି Query function to be used
1 7 2-6

```
int Query(int node, int b, int e, int i, int j)
```

if ($i > e \text{ or } j < b$)

return 0;

if ($b >= i \text{ and } e <= j$)

return tree[node];

left = node * 2;

right = node * 2 + 1;

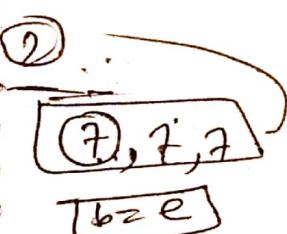
mid = $(b+e)/2$;

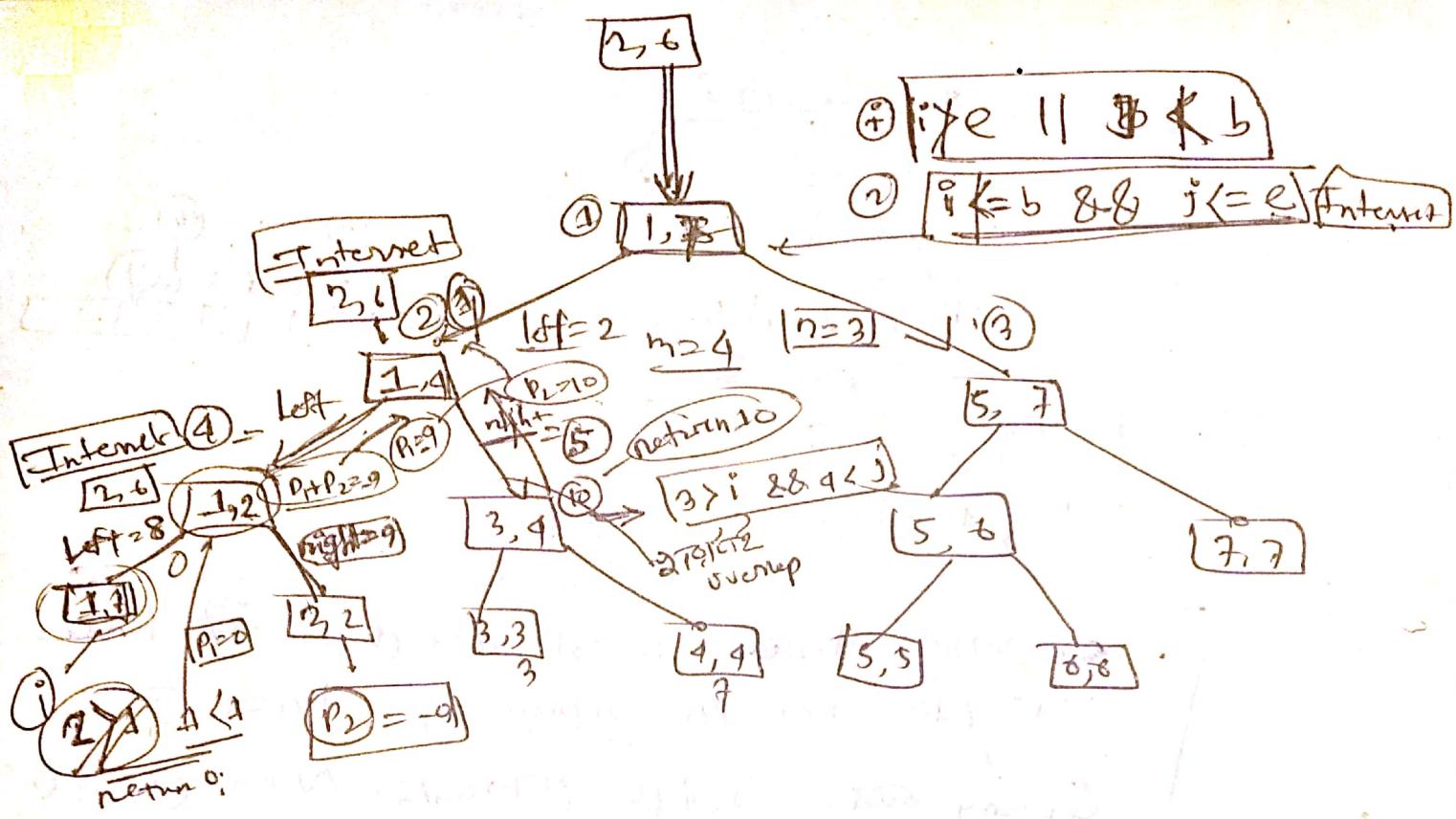


$P_1 = \text{Query}(\text{left}, b, \text{mid}, i, j)$

$P_2 = \text{Query}(\text{right}, \text{mid}+1, e, i, j)$

return $P_1 + P_2$;





एक रेक्यूरेटिव रेंज सेक्यूरिटी ब्रॉडकास्टिंग त्री द्वारा किसी भी विशेष रेंज के लिए अनुसन्धान किया जाता है।

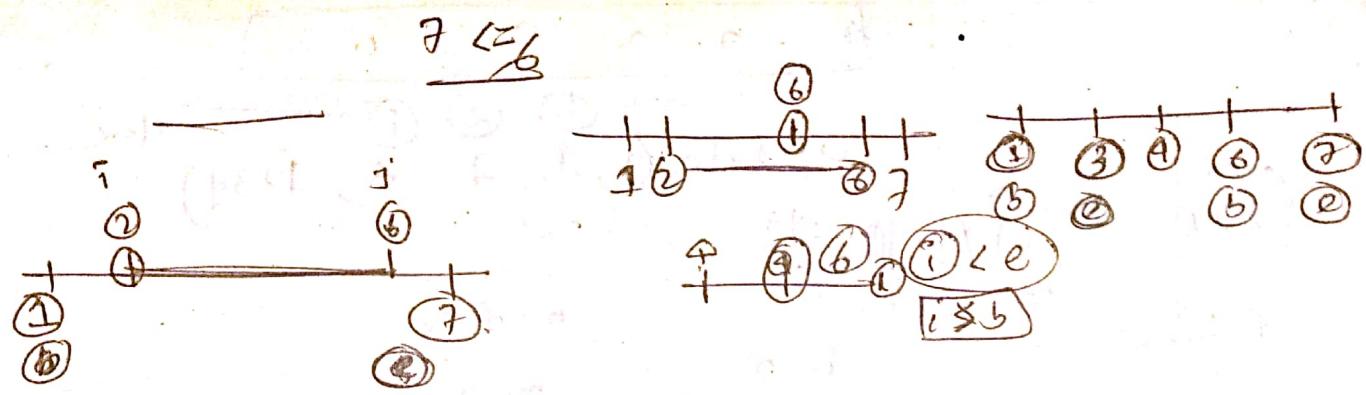
यह रेक्यूरेटिव रेंज सेक्यूरिटी त्री का उपयोग इसकी विशेषताओं के लिए किया जाता है कि इसमें विशेष रेंज के लिए अनुसन्धान करने की कठिनी नहीं है।

(d) $T_1 \cap T_2$

दो विशेष रेंजों का अनुसन्धान

(d) $T_1 \cup T_2$ या $\neg T_1 \cap \neg T_2$

दो विशेष रेंजों का अनुसन्धान



Update एवं Array element Segment tree द्वारा

କୁଣ୍ଡଳ ଏଥିର ଗାଁଳ ନାମ - T. myro tree

Tree कोड में तो, then \rightarrow tree & update
array कोड जिसके position & tree[node] दह
Value change होता है। ऐसी प्रौद्योगिकी दह
आजकल value change करने के लिए
नया value दे सकता है तो वहाँ 270।

Upstate (note, b, e, i, val)

if ($b > i$ || $e < i$) return;

if (b == i && e == i) {

4. tree(node) < val;
return;

left = node*2

$$\text{right} = \text{node}^{\star 2+1}$$

$$\text{mid} = (b+e)/2;$$

update = (left, b, mid[i].val)

update(night, mdp, val);

$\text{tree}[\text{node}] = \text{tree}[\text{left}] + \text{tree}[\text{right}]$.

4

4 - 9 3 7 1 0 2

note (b) (e) (i) Position

value

update (1, 1, 7, 6, 1234)

eLi
4 < 6

return

Update 2, 1, 4, 6, 1234

1 6 8 6 7
1 6 8 8 6 7 7

l = 2

m = 4

n = 3

m = 4

update 3, 5, 7, 6, 1234

Update

l = 6
m = 6

l = 4 m = 2

Update (4
1, 2, 6, 1234)

8
1 7

l = 8
m = 1

return

l = 12 m = 5

1 2 5, 5, 6, 1234

1 2 4 [3] = 1234

update (13, 6, 6) 1234

Satisfied

Implementation

using stack

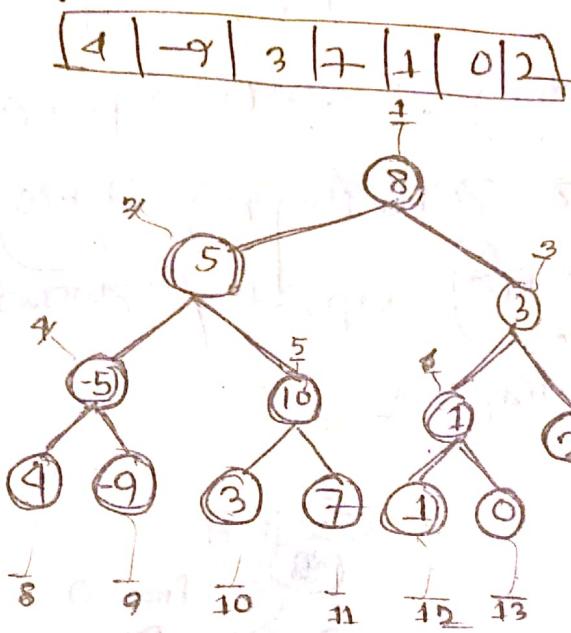
using queue

using linked list

using binary search tree

Segment tree with lazy propagation

Let's have a segment tree:



এই tree টা আমরা এডি করে কোথায় কিন্তু গোপনীয় রেখা, কোথা element র মাঝে দোস্ত শিখাব। এই হোট, সুবিধি কর্ম করা/হোট এবং কোথা কোথা, নির্দিষ্ট Range র মধ্যে গোপনীয় করে দেও। এবং এল; (সম্পৃক্ত আমরা পুরুষ সংখ্যার ক্ষেত্রে কেবল সংজ্ঞা), এই tree টা আমরা উদ্দেশ্য করে কোথায় কোথা দোস্ত শিখাব।

ବ୍ୟାପକ ଯଦି ଜାମାଦୀ ଲୋଗ ହେବୁ (ନିରାପଦ ହେବୁ
ଏବଂ ଆମ ଏବୁ ଏକ range ରେ କୌଣସିବା
କଣ୍ଠେ ମୋତେ ଯାଏଣ ଯଦି ଏହି ଏକିଏ କୌଣସିବା

ପାଇଁ କିମ୍ବା କିମ୍ବା କିମ୍ବା କିମ୍ବା କିମ୍ବା କିମ୍ବା କିମ୍ବା

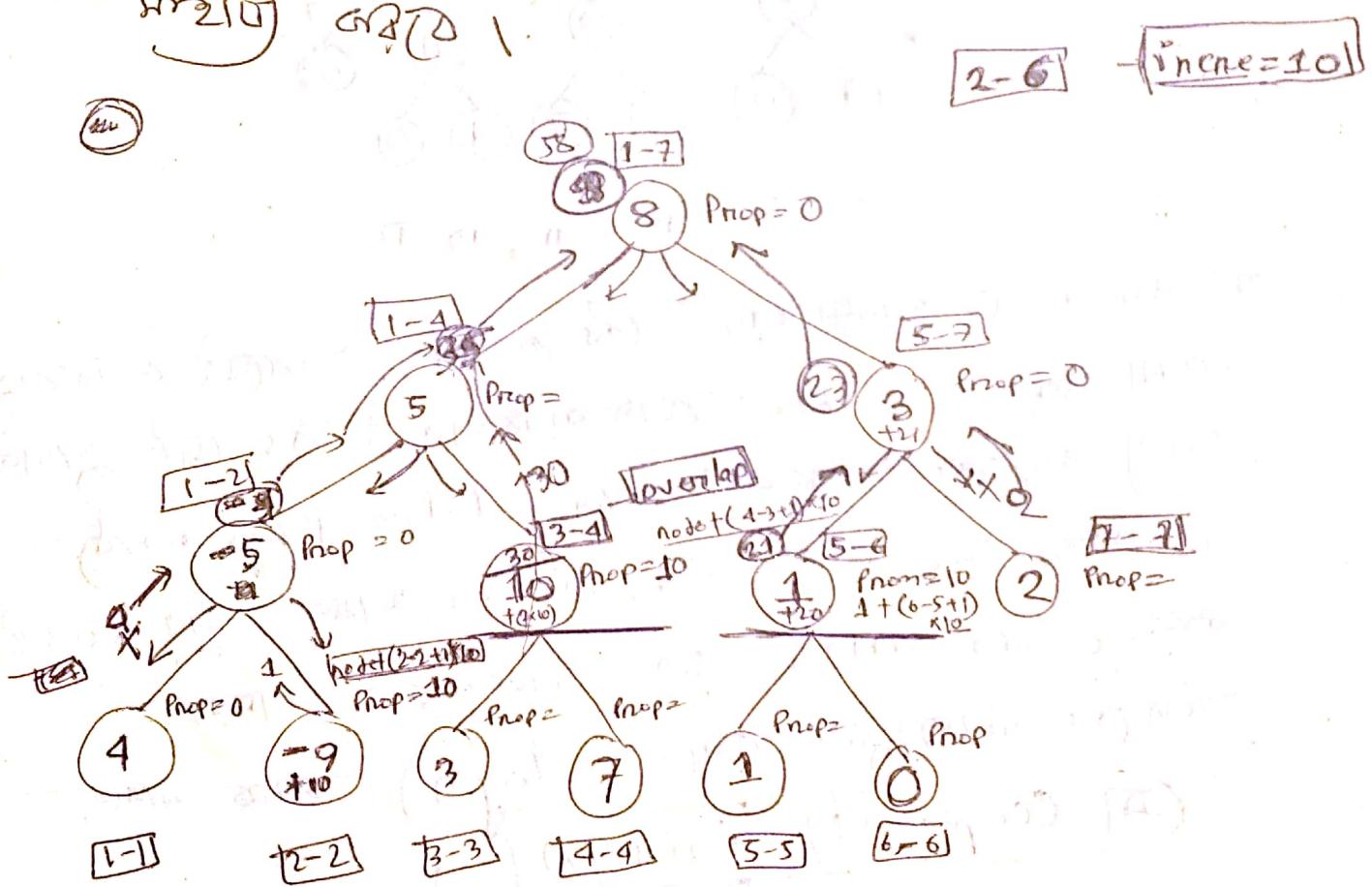
ଆମଣୀ ଗ୍ରାମୀ କଲେଜ୍ ପାଠ୍ୟକର୍ତ୍ତା ଗ୍ରାମୀ ନାମରେ ଏହାରେ କିମ୍ବା କିମ୍ବା କିମ୍ବା

worst case \rightarrow O(N^2) range query complexity

दाखिला($n \log n$) : यहाँ एक अमर्त्य- वाक्ता Algorithm

କେବଳ ଅନ୍ତର୍ଗତ ପାଇଁ ଲାଗୁ ହେଲା

Range (०.५८) update / तरावा और अन्य (०.५८)



બેસ ટ્રી Basic tree. જવા હતી 25 [2-5]

ରେ Range ଏହା ମାତ୍ର କୁଣ୍ଡ ଦୟା ଏବଂ ପାଇଁ

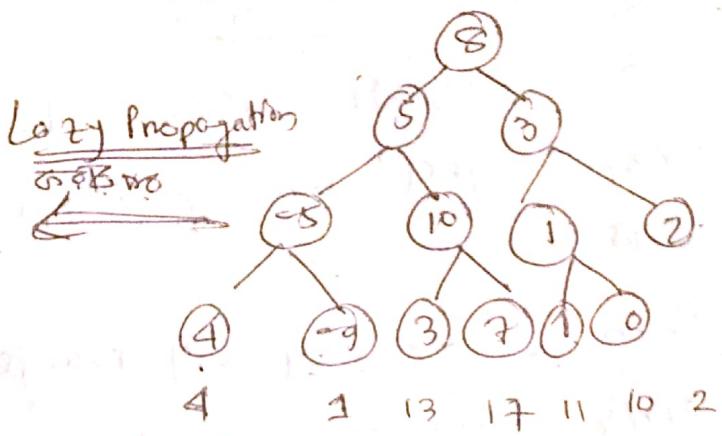
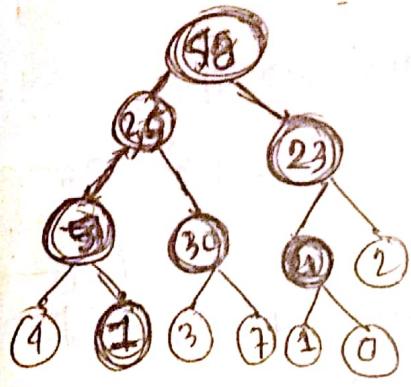
ମେଟ୍ ପାତ୍ର କରସାମେ ପ୍ରତିକ୍ରିୟା

କ୍ଷେତ୍ର ଆମ୍ବୁଟ କିମ୍ବା?

ଗାମଣ ଏବଂ ପାଶା - Leaf node ହେଉ ଥାଏ,
ଏବଂ ଯାଣ overlap ହୋଇ Range କିମ୍ବା ଅନ୍ତର୍ଭେଦ ହୋଇ
ଥାଏ,

then \rightarrow overlap ହେବାରେ ଉପରିବାଟି leaf note
କିମ୍ବା ହେବାରେ (କୌଣସି, then \rightarrow increment କରିବାକୁ
 \rightarrow କ୍ଷେତ୍ର କିମ୍ବା / note କିମ୍ବା update କରିବାକୁ) କିମ୍ବା ଆମଣ
କାହାର କିମ୍ବା Increment କରିଲାମ ତାକୁ \rightarrow କ୍ଷେତ୍ର କିମ୍ବା / note

ଏବୁ propagate value କିମ୍ବା ଟ୍ରେସାଇବୁ, ଏବୁ
ଅନ୍ତର୍ଭେଦ କିମ୍ବା କରିବାକୁ ଆମଣ କରିବାକୁ node overlapped
Range କାହାର କିମ୍ବା update କରି କିମ୍ବା କରିଲାମ ତାକୁ
update କରିବାକୁ କିମ୍ବା କରିବାକୁ (ପ୍ରଶନ୍ନାଳୀ Query କିମ୍ବା
update କରିବାକୁ କିମ୍ବା କରିବାକୁ), ଏବୁ କିମ୍ବା କରିବାକୁ କିମ୍ବା
କିମ୍ବା value increment କରିବାକୁ, ଏବୁ କିମ୍ବା
 \rightarrow overlap Range \rightarrow belong କରିବାକୁ
ଆପଣଙ୍କ \rightarrow range note କିମ୍ବା update କରିବାକୁ
ଏବୁ update କରିବାକୁ କିମ୍ବା କରିବାକୁ total prop.
କିମ୍ବା କରିବାକୁ



આમારું (-6) range (0 to 10) increment કરીને

એવી રીતે કેવી આપ્યું હોય એવી અપડેટ કરીને update કરી જાને

$$\underline{9 \ 14 \ 13 \ 7 \ 1 \ 0} \Rightarrow \text{total} = 58$$

$$\underline{9 \ 14 \ 13 \ 7 \ 17 \ 11 \ 10 \ 2} \Rightarrow \text{total} = 58$$

જેવી રીતે કેવી માટે update કરીને કેવી રીતે range(0 to)

update કરીને એવી સ્થિરીય update કરીને એવી કેવી રીતે,

Root node કરીને Array કરીને

Updated sum કરીને | એવી કોઈ કોઈ struct

નથી કરીને tree કરીને array કરીને |

struct info {

int sum;

int prop;

} tree[mx * 4];

int array[mx];

Array - કાણી 28
4 25 25 25 |

```

    void update(int node, int b, int e, int n, int y) {
        if (b > e || e < n) return; [out of range]
        if (b >= n & e <= y) {
            tree[node].sum = tree[node].sum + (e - b + 1) * val;
            tree[node].prop += val;
            return;
        }
        int left, right, mid;
        left = node * 2;
        right = node * 2 + 1;
        mid = (b + e) / 2;
        update(left, b, mid, n, y, val);
        update(right, mid + 1, e, n, y, val);
        tree[node].sum = tree[left].sum + tree[right].sum;
        tree[node].prop += (tree[node].prop * (e - b + 1));
    }

```

इसी तरह update करते हैं।

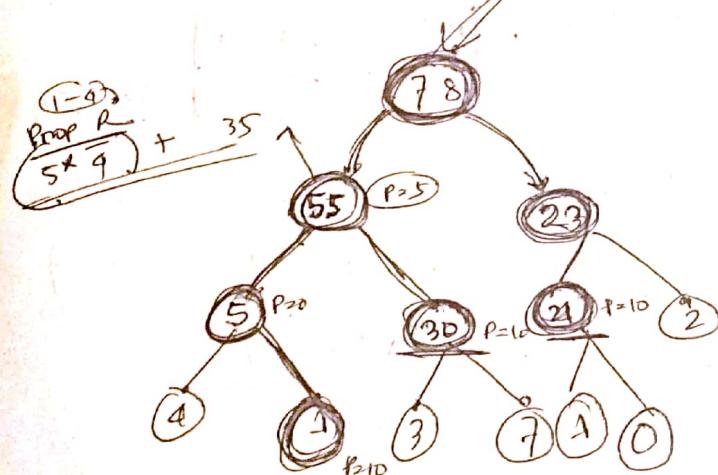
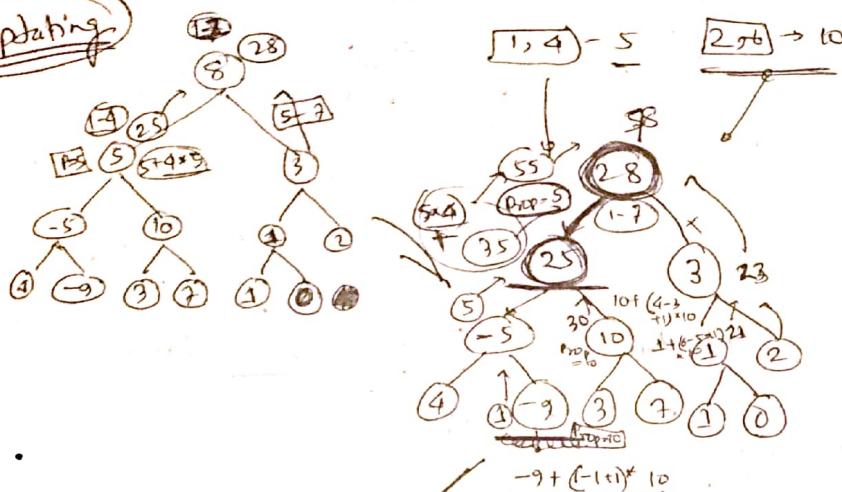
परन्तु यह क्या काम करता है? Query क्या है?

यह काम करता है कि 2-5 Range का किसी भी विशेषज्ञता का काम करता है।

After update 702 value का क्या बदलाव होगा?

दोषों का काम करता है।

Updating



Data structure Segment tree Lazy propagation.

Carry

$$[4 \quad 9 \quad 3 \quad 7] \quad 1 \quad 0 \quad 2$$

$$9 \quad [4 \quad 8 \quad 12 \quad 1 \quad 0] \quad 2 = \underline{\underline{1, 4} + 5} \\ = 28$$

$$9 \quad 4 \quad 18 \quad 22 \quad 11 \quad 10 \quad 3 = \underline{\underline{2, 6} + 10} \\ = 78$$

বৈধ কী? update কোরি এখন কোরি দেখুন

array -> একটা স্লিপেজ মানেগুলো ফার্গ ট্ৰি

(এখনো কোথো ফার্গ তাৰ বিষ্ণু, Tree কোথো
বৈধ কোৱা Page - 6০ টত।

একে কোৱা কৈ $(2-5)$ কোৱা $(3-2)$ element
মুলখণ্ড, স্লিপেজ কোৱা কৈ কৈ | কৈ কৈ |
সময় | কোৱা কোৱা | $(3 \text{ কো } 4, 5)$ ৩২ element
মুলখণ্ড কোৱা | update - ২ কোৱা কোৱা | tree |
৩২ update | কোৱা কোৱা কোৱা কোৱা | কোৱা | কোৱা |
কোৱা কোৱা ?

প্রশ্ন Query কোৱা কোৱা কোৱা | Carry use
কোৱা | ২৫ | Carry - কোৱা কোৱা?

Carry - কোৱা কোৱা | element |
কোৱা | Increment কোৱা কোৱা কোৱা |
আসা | কোৱা | Carry কোৱা কোৱা ?

ତାଙ୍କ || prop ଏବଂ ଉପରେ ଦୟାରେ ଡାକ୍ତିଲାମ୍ ମାନସରେ
ଆହାନ ଦିଲା କାହିଁ Increment କଥାମ କି ପ୍ରୋଫ୍ଟ୍ରୁଏୟା।

ଏହା carry କି ଅଧିକ୍ଷତ୍ତୁ ଜେଣେ (element / leaf) କୌଣସି

କହିଲୁ prop କିମ୍ବା ଏହା କିମ୍ବା ଏହାରେ କିମ୍ବା ଏହାରେ

leaf କି ଯାର ତେ Increment କରାବାବୁ carry

ବ୍ୟାଖ୍ୟା । Carry = prop; -ଆହା -ଏହାରେ କିମ୍ବା

leaf value କି କାହାରେ କମାଇଲୁଛି କାହାରେ ଗୋଟିଏ ।

ଏହାକିମ୍ବା ଏହାରେ Return କିମ୍ବା ଫିରିବା କିମ୍ବା କିମ୍ବା ।



int

~~Query~~ Query(int node, int b, int e, int x, int y){

if(b>y || e < x) return 0;

if(b>=x & e <= y){

return tree[node].sum + (e-b+1)*carry;

int left, right, mid;

left = node*2;

right = node*2+1;

mid = (b+e)/2;

int p1 = (left, b, mid, x, y, carry + tree[node].prop);

int p2 = (right, mid+1, e, x, y, carry + tree[node].prop);

return p1+p2;

7



Query Process

After Update :

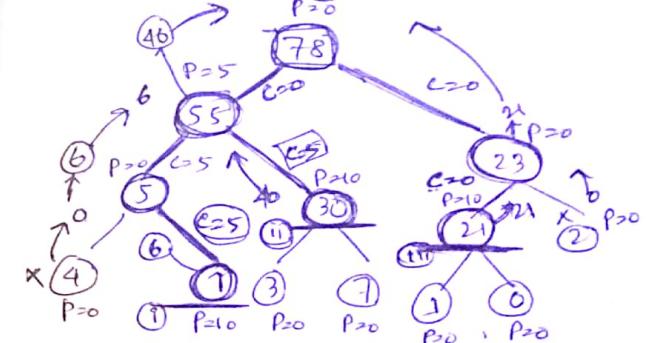
[1, 4] + 5

[2, 6] + 10

(6, 7)

P=0

↓ → overlo



$$\textcircled{1} \quad 1 + (c * (e-b+1)) \\ 1 + (5 * 1) = \textcircled{6}$$

$$\textcircled{11} \quad 30 + (c * (e-b+1)) \\ 30 + (5 * 2) = 40.$$

$$\textcircled{11} \quad 21 + (c * (e-b+1)) \\ 21 + (0 * 2) = 21$$