Data Structure 2
RMQ, BIT, Segment Tree

Jonathan Irvin Gunawan
National University of Singapore

prerequisite

bisa ngitung

udah mandi tadi pagi

motivation problem

buat hari ini

dikasih array N, dikasih query Q. tiap query bisa either update isi array (bisa range) atau cari jumlah dari sebuah range

1. gak ada update
2. semua updatenya di awal
3. updatenya cuma bisa satu titik doang
4. no additional constraint

1. gak ada update

2. semua updatenya di awal

3. updatenya cuma bisa satu titik doang

4. no additional constraint

prefix sum doang lah

ya

1. gak ada update

2. semua updatenya di awal

3. updatenya cuma bisa satu titik doang

4. no additional constraint

papan tulis aja lah ya

since ini bukan fokus hari ini, gw gak bakal bahas ini detil

1. gak ada update
2. semua updatenya di awal
3. updatenya cuma bisa satu titik doang
4. no additional constraint

BIT

BIT

Buseeeeet Irvin Tampan

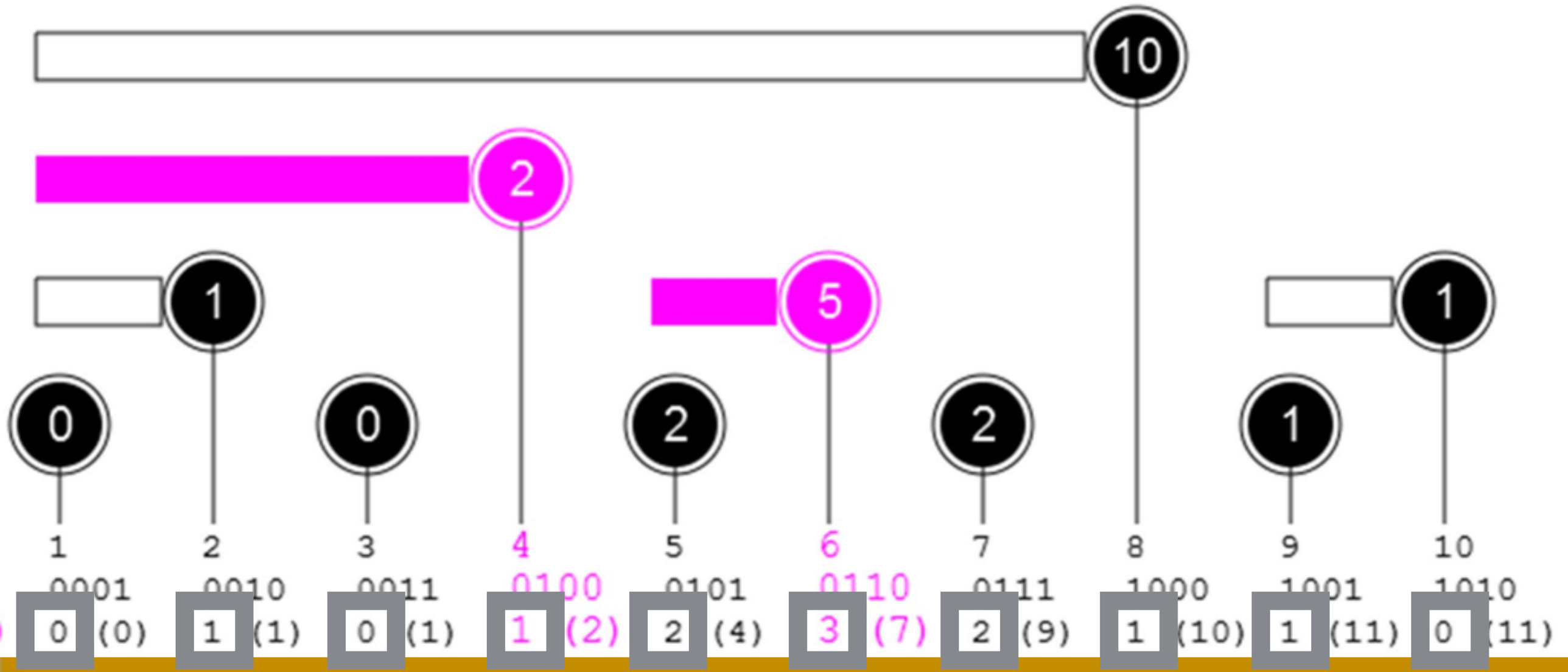
Binary Indexed Tree

a.k.a. Fenwick Tree

bikin array BIT

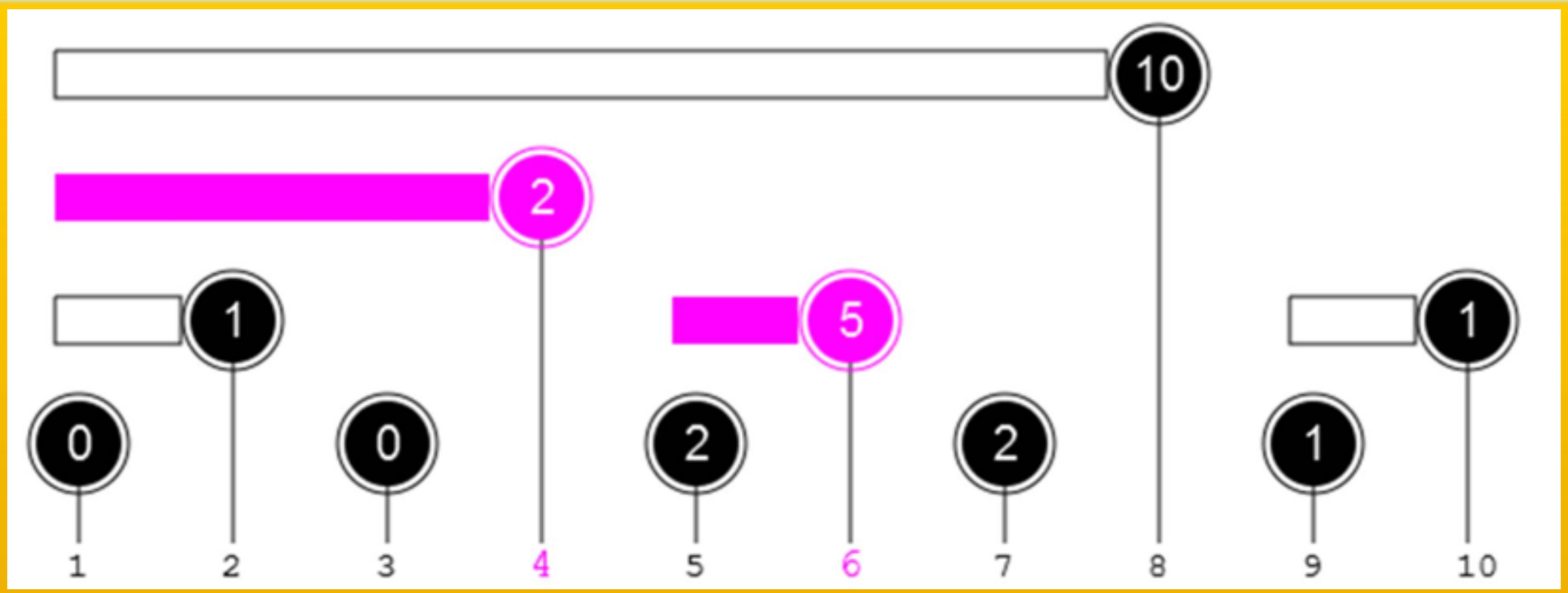
BIT[x] = sum dari A[x-k+1,x]
dimana k = x & (-x)

x = 1, k = 1, BIT[1] = A[1]
x = 2, k = 2, BIT[2] = A[1] + A[2]
x = 3, k = 1, BIT[3] = A[3]
x = 4, k = 4, BIT[4] = A[1] + A[2] + A[3] + A[4]
x = 5, k = 1, BIT[5] = A[5]
x = 6, k = 2, BIT[6] = A[5] + A[6]
x = 7, k = 1, BIT[7] = A[7]
x = 8, k = 8, BIT[8] = A[1] + A[2] + … + A[8]


10
2
1
5
1
0
2
0
2
1
1 0001 0 (0)
2 0010 1 (1)
3 0011 0 (1)
4 0100 1 (2)
5 0101 2 (4)
6 0110 3 (7)
7 0111 2 (9)
8 1000 1 (10)
9 1001 1 (11)
10 1010 0 (11)

kalo pake BIT querynya bisa query A[1] + A[2] + … + A[X]

gimana caranya?



$$S[x] = A[1] + A[2] + \ldots + A[x]$$

$S[1] = BIT[1]$
$S[2] = BIT[2]$
$S[3] = BIT[3] + BIT[2]$
$S[4] = BIT[4]$
$S[5] = BIT[5] + BIT[4]$
$S[7] = BIT[7] + BIT[6] + BIT[4]$

in general

$$S[k] = BIT[k] + S[k - k \,\&\, (-k)]$$

karena tadi BIT[k] cuma nyimpen sum dari k elemen terakhir kan

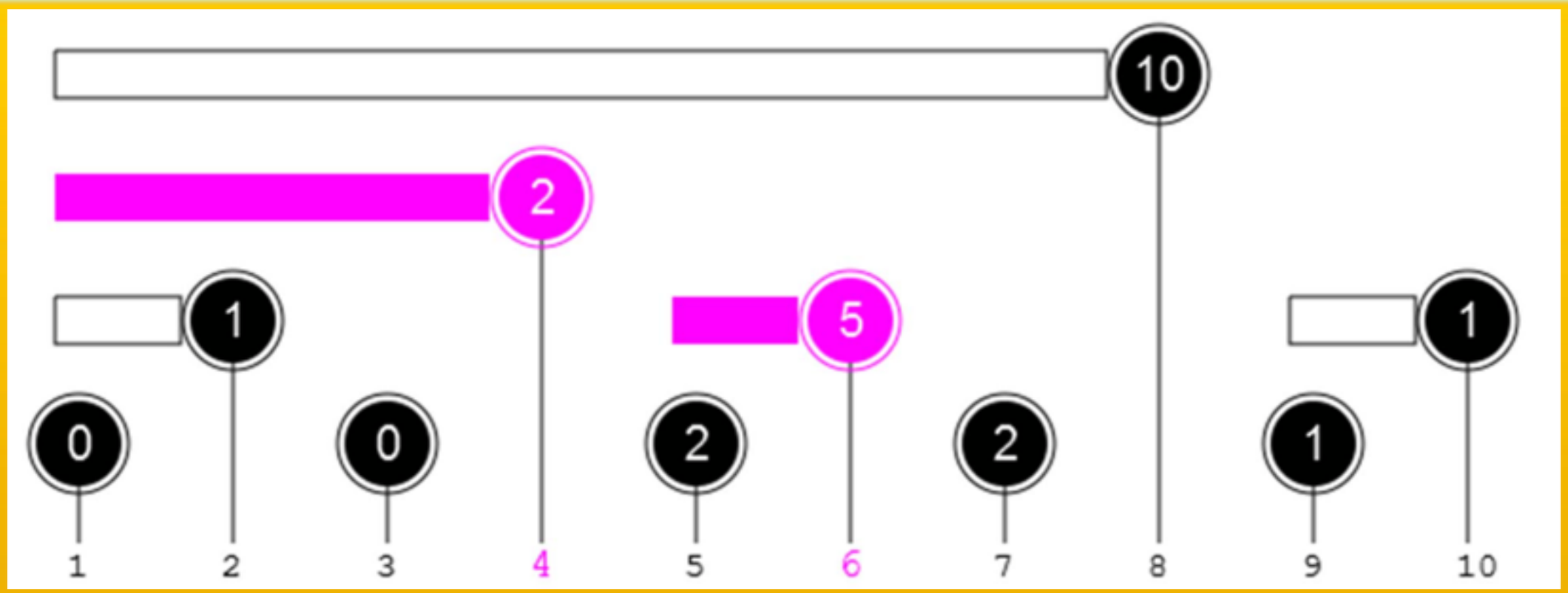
query(int x)

```
if (x == 0) return 0;
return BIT[x] += query(x - (x & -x))
```

```
// atau

int ret = 0;
for (int i = x; i > 0; i -= (i & -i)) {
  ret += BIT[i]
}
```

ngupdatenya gimana?
kita harus mastiin update(x)
bakal update semua BIT[y]
dimana $y - (y \& -y) < x \leq y$



kalo update(5), BIT[5], BIT[6], BIT[8] berubah

update(int x, int val)

```
for (int i = x; i <= MAXN; i += (i & -i)) {
  BIT[i] += val;
}
```

kalo lu panggil update(0)
dia bakal infinite loop

gimana kalo mau query(A,B)?

query(A,B)

query(B) - query(A-1)

bisa buat query max(A[1],A[2],...A[X]) juga

BIT nya bakal nyimpen maksimum instead of sum

query(int x)

```
if (x == 0) return 0;
return max(BIT[x],query(x - (x & -x)));
```

```
// atau

int ret = 0;
for (int i = x; i > 0; i -= (i & -i)) {
  ret = max(ret, BIT[i])
 }
```

update(int x, int val)

```
for (int i = x; i <= MAXN; i += (i & -i)) {
   BIT[i] = max(BIT[i], val)
}
```

aplikasi : DP LIS

quick review LIS :
cari subset dari sebuah array
yang increasing yang
panjangnya maksimum

DP O(N^2)

```
int ret = 0;
for (int i = 0; i < N; ++i) {
  dp[i] = 1;
  for (int j = 0; j < i; ++j) {
    if (A[j] < A[i]) {
      dp[i] = max(dp[i], dp[j] + 1);
    }
  }
  ret = max(ret, dp[i]);
}
return ret;
```

DP O(N^2)
sambil ilustrasi di papan tulis deh

```
int ret = 0;
for (int i = 0; i < N; ++i) {
  dp[i] = 1;
  for (int j = 0; j < A[i]; ++j) {
    dp[i] = max(dp[i], val[j] + 1);
  }
  val[A[i]] = max(val[A[i]], dp[i]);
  ret = max(ret, dp[i]);
}
return ret;
```

DP O(N lg N)

```
int ret = 0;
for (int i = 0; i < N; ++i) {
  // for (int j = 0; j < A[i]; ++j) {
  //   dp[i] = max(dp[i], val[j] + 1);
  // }
  dp[i] = query(A[i] - 1) + 1;
  // val[A[i]] = max(val[A[i]], dp[i]);
  update(A[i], dp[i]);
  ret = max(ret, dp[i]);
}
return ret;
```

1. gak ada update
2. semua updatenya di awal
3. updatenya cuma bisa satu titik doang
4. no additional constraint

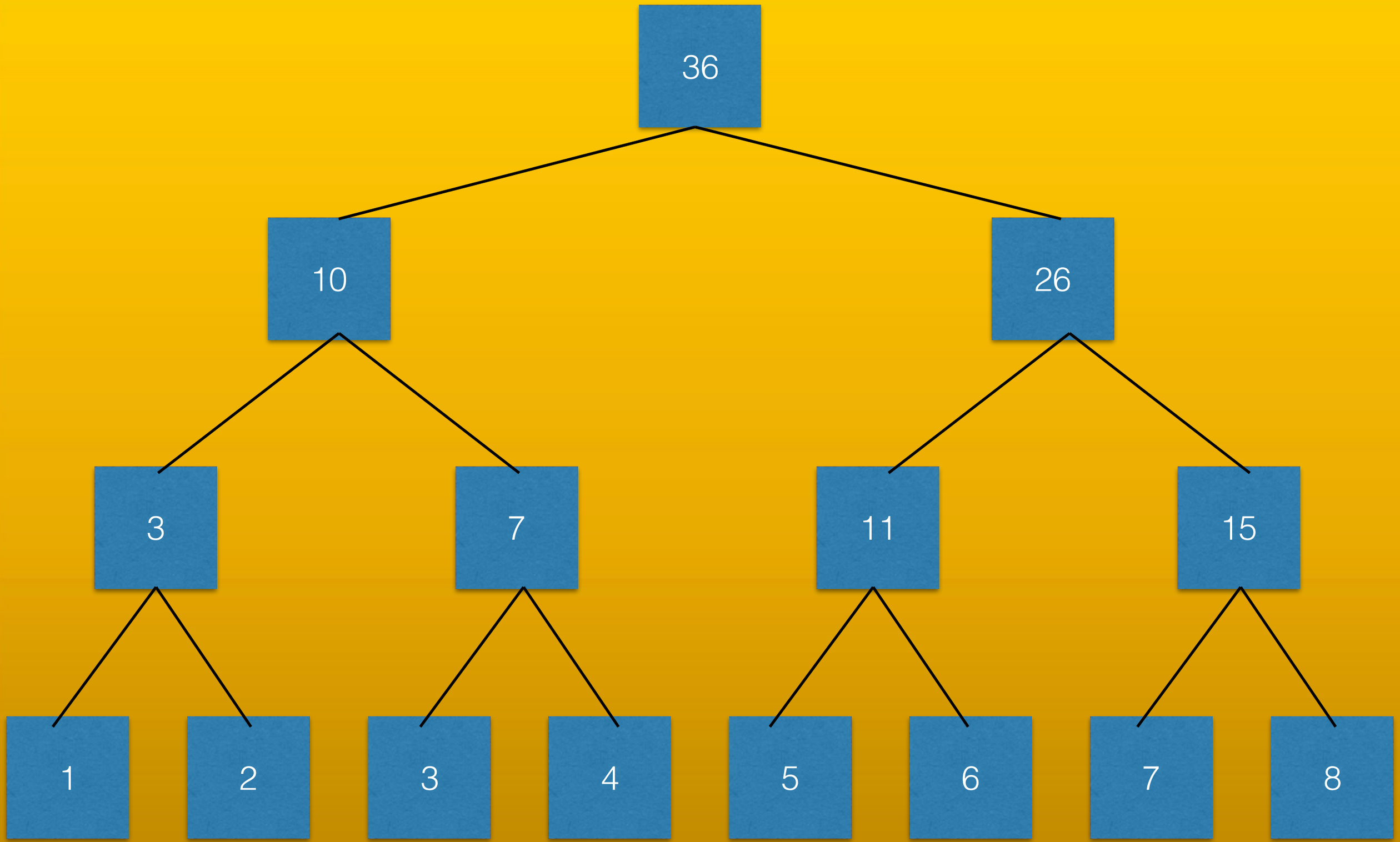
Segment Tree

buat belajar segment tree, kita balik ke problem 3 dulu (updatenya cuma bisa titik)

kalo di segment tree

ada node yang nyimpen sum dari subarray yang panjangnya A[N], A[N/2], A[N/4], ...

misal N = 8

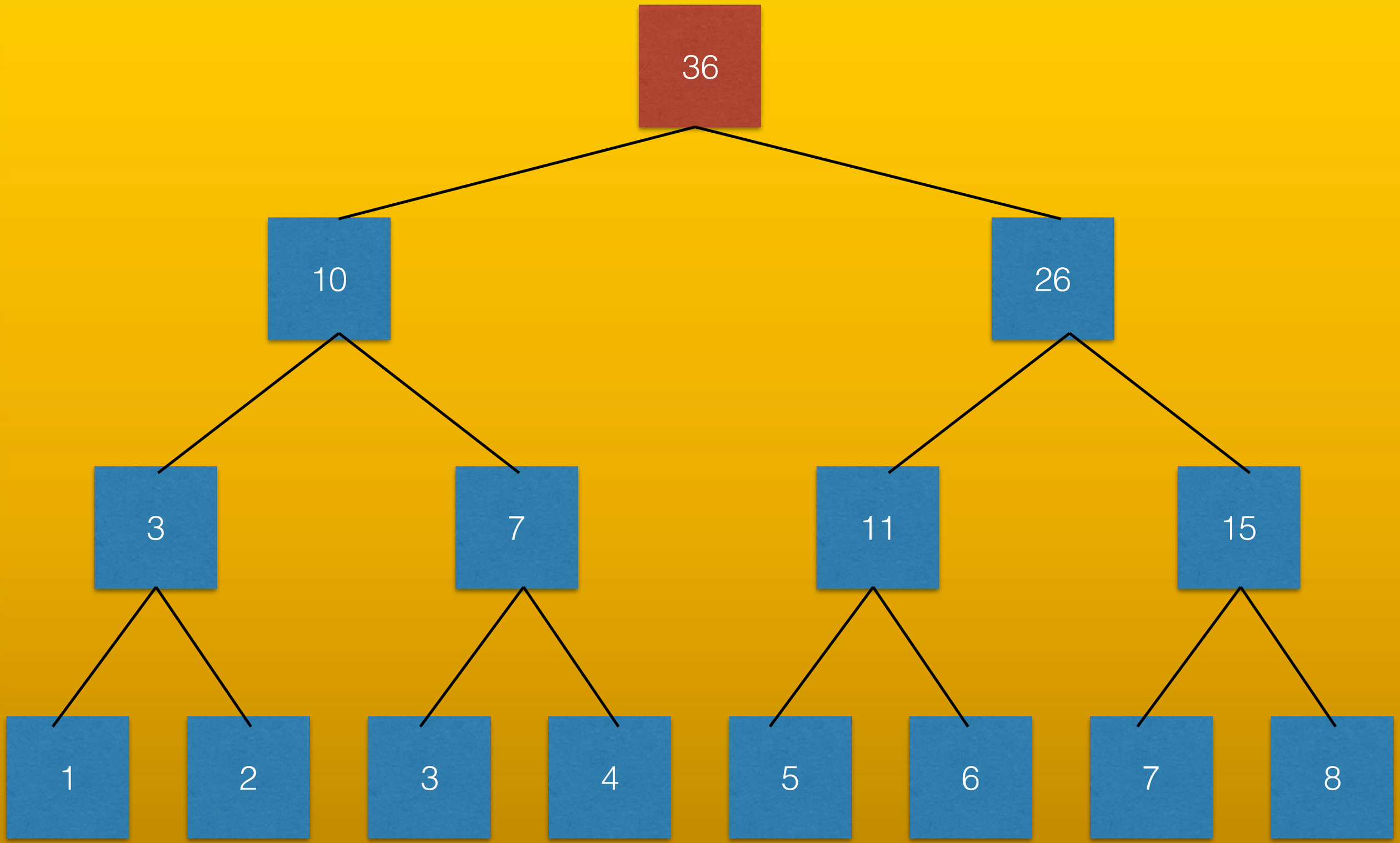

36
10
26
3
7
11
15
1
2
3
4
5
6
7
8

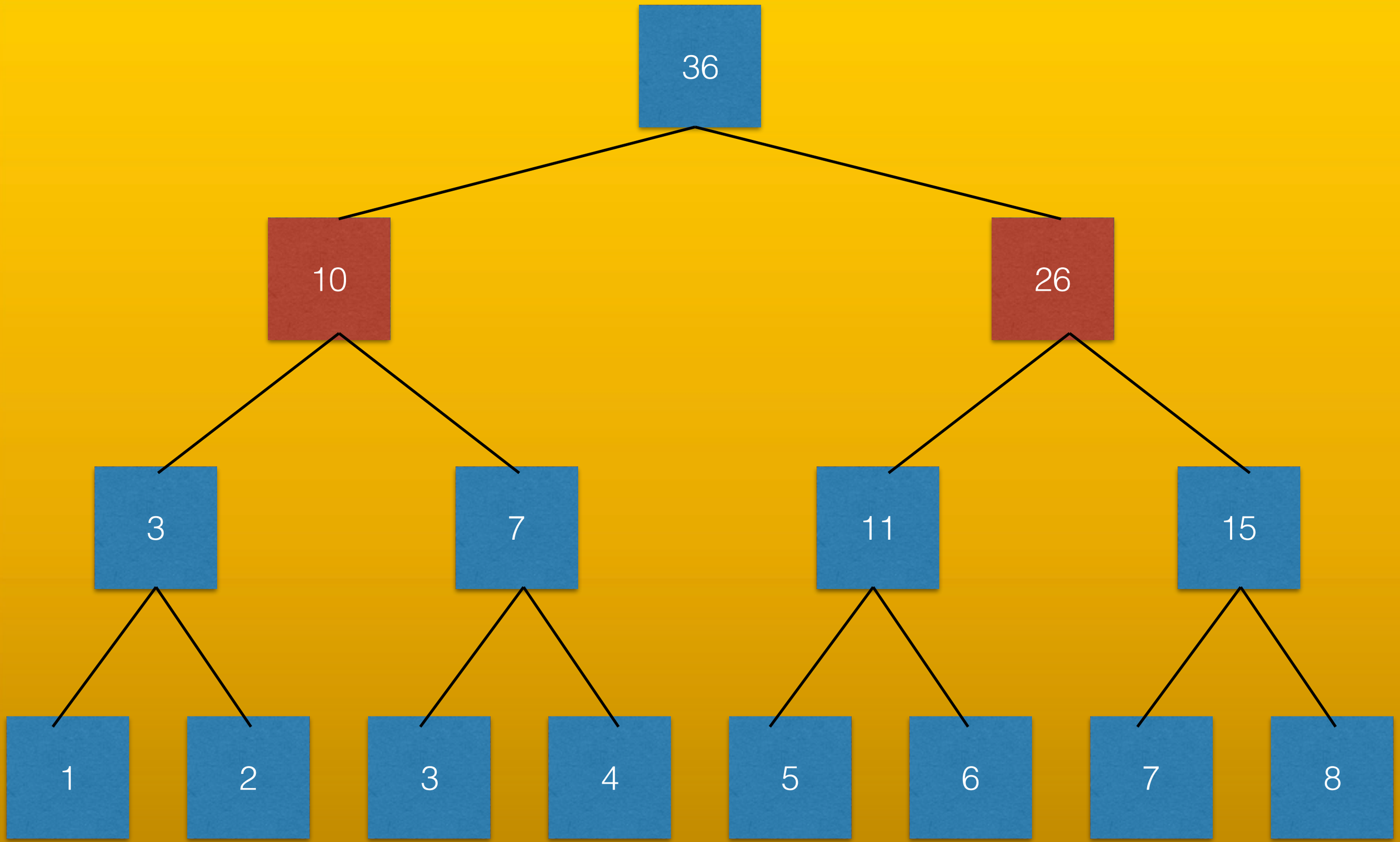
gimana cara buildnya

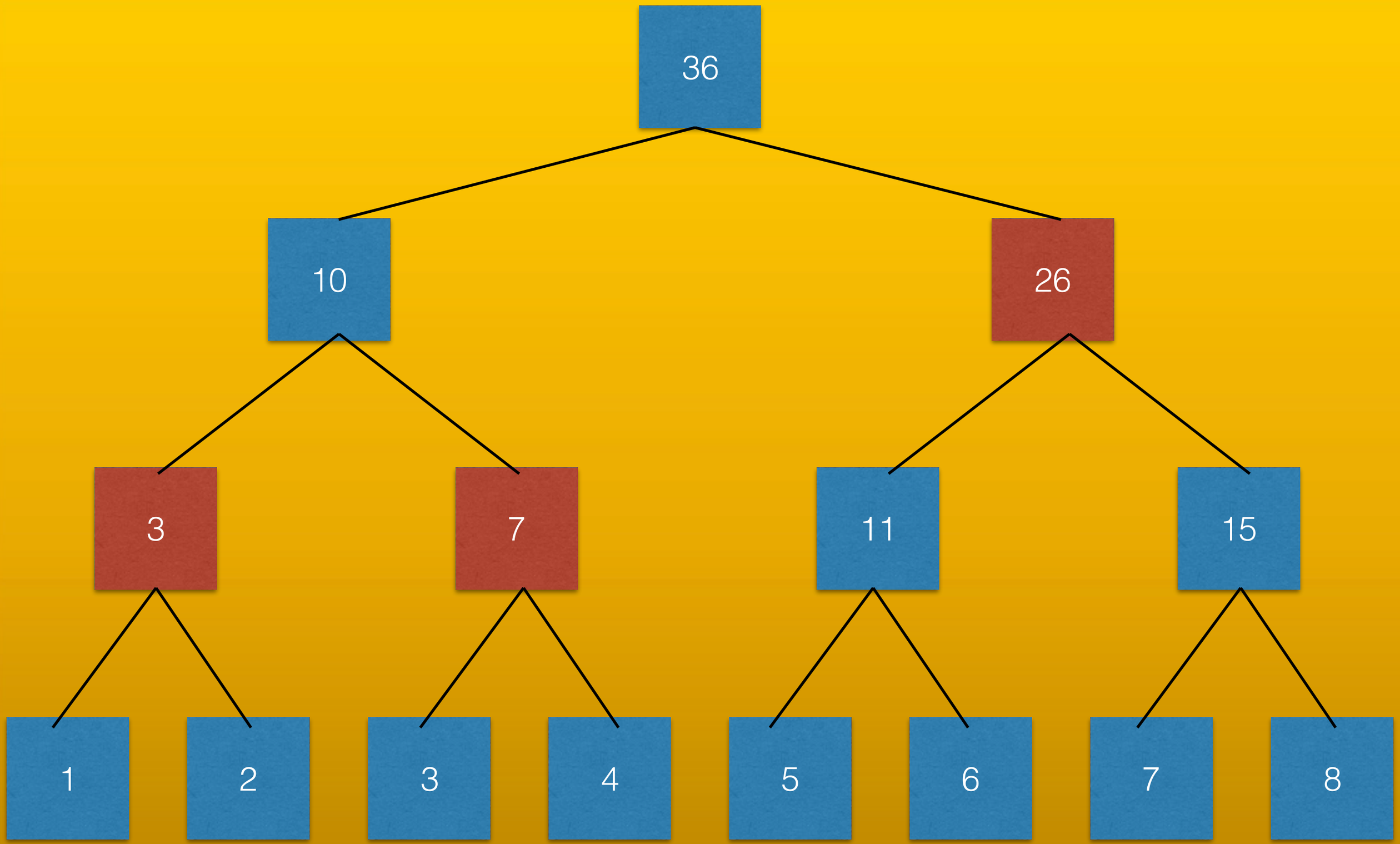
kita skip dulu yah

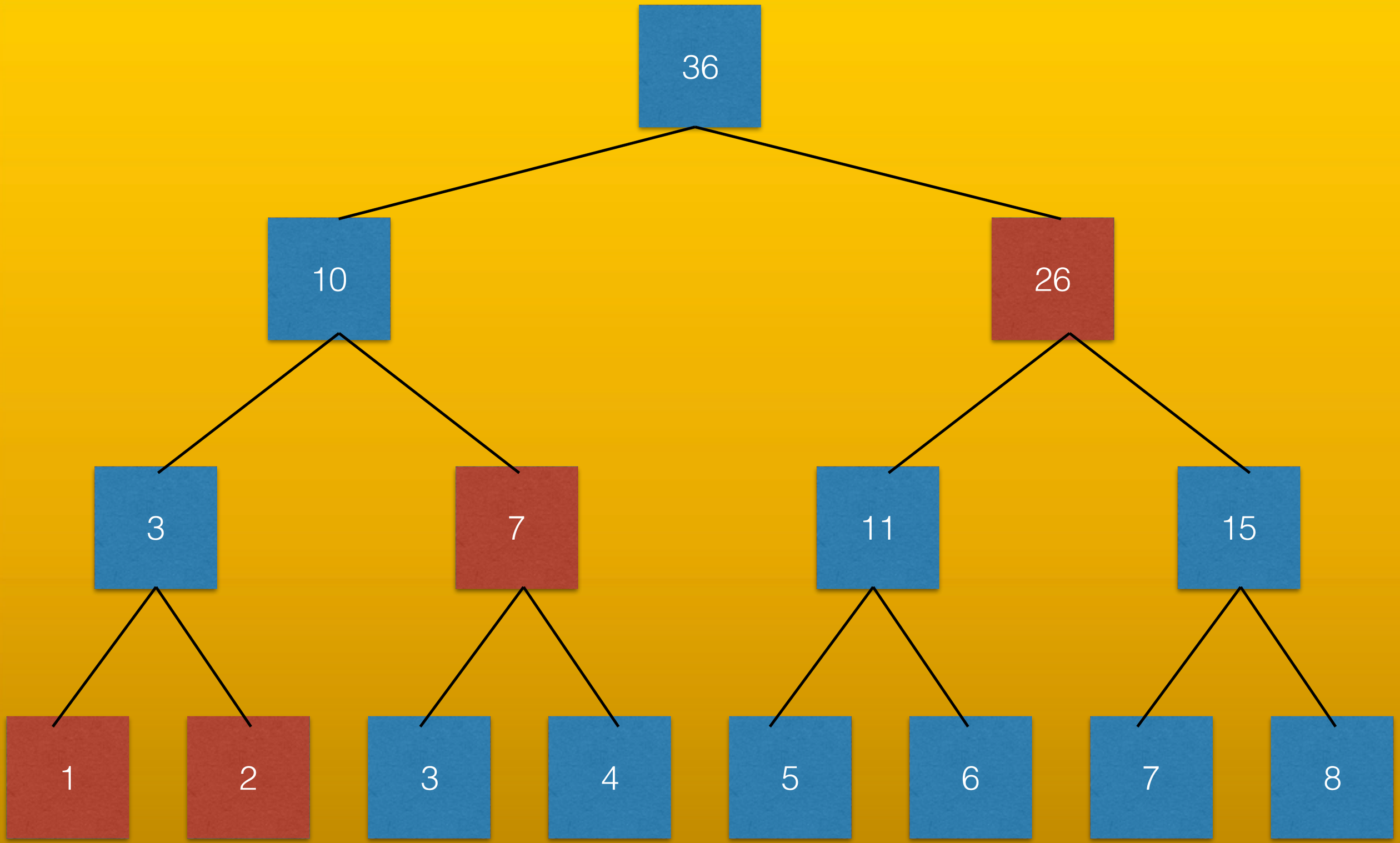
misalkan mau query

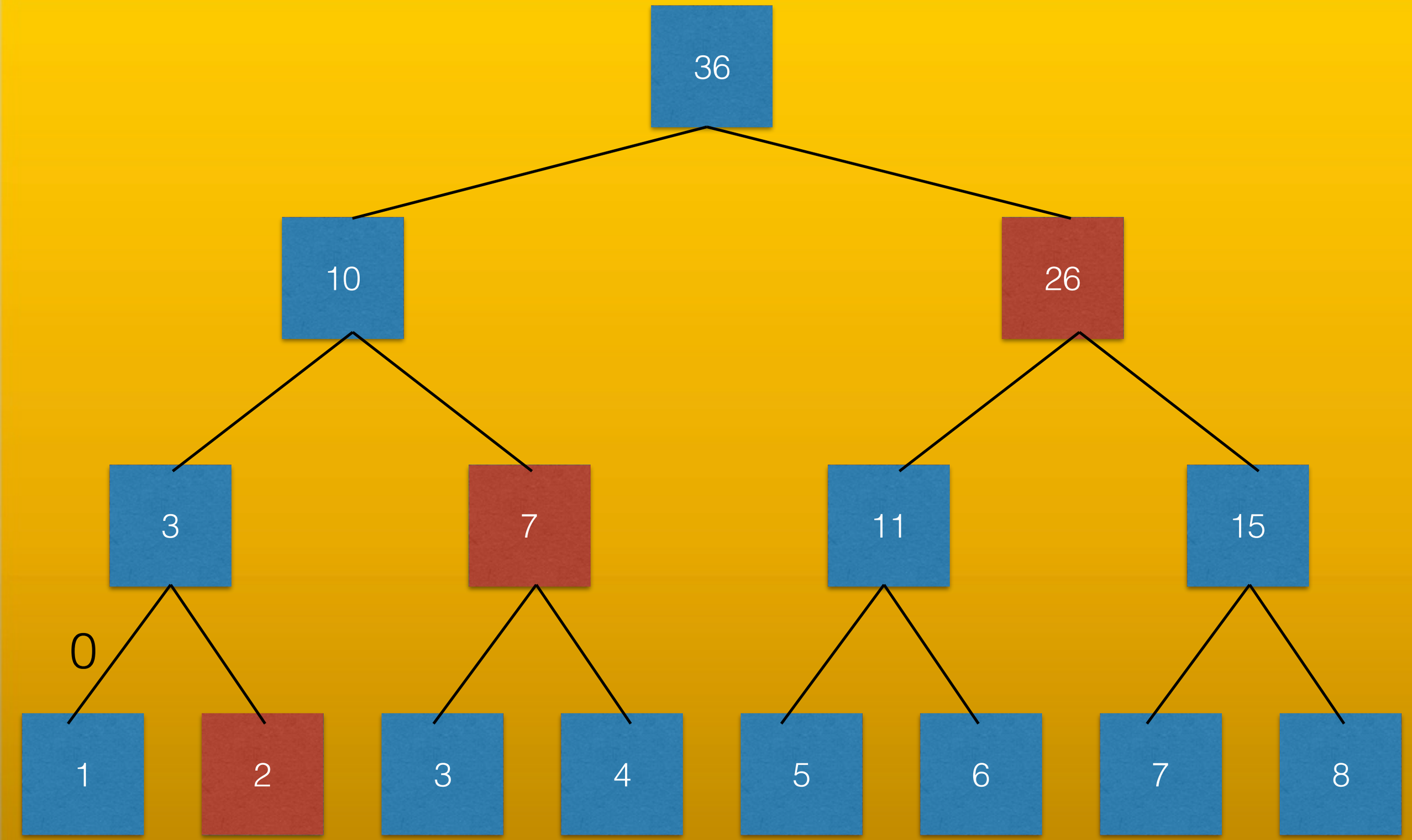
A[2] + A[3] + … + A[6]

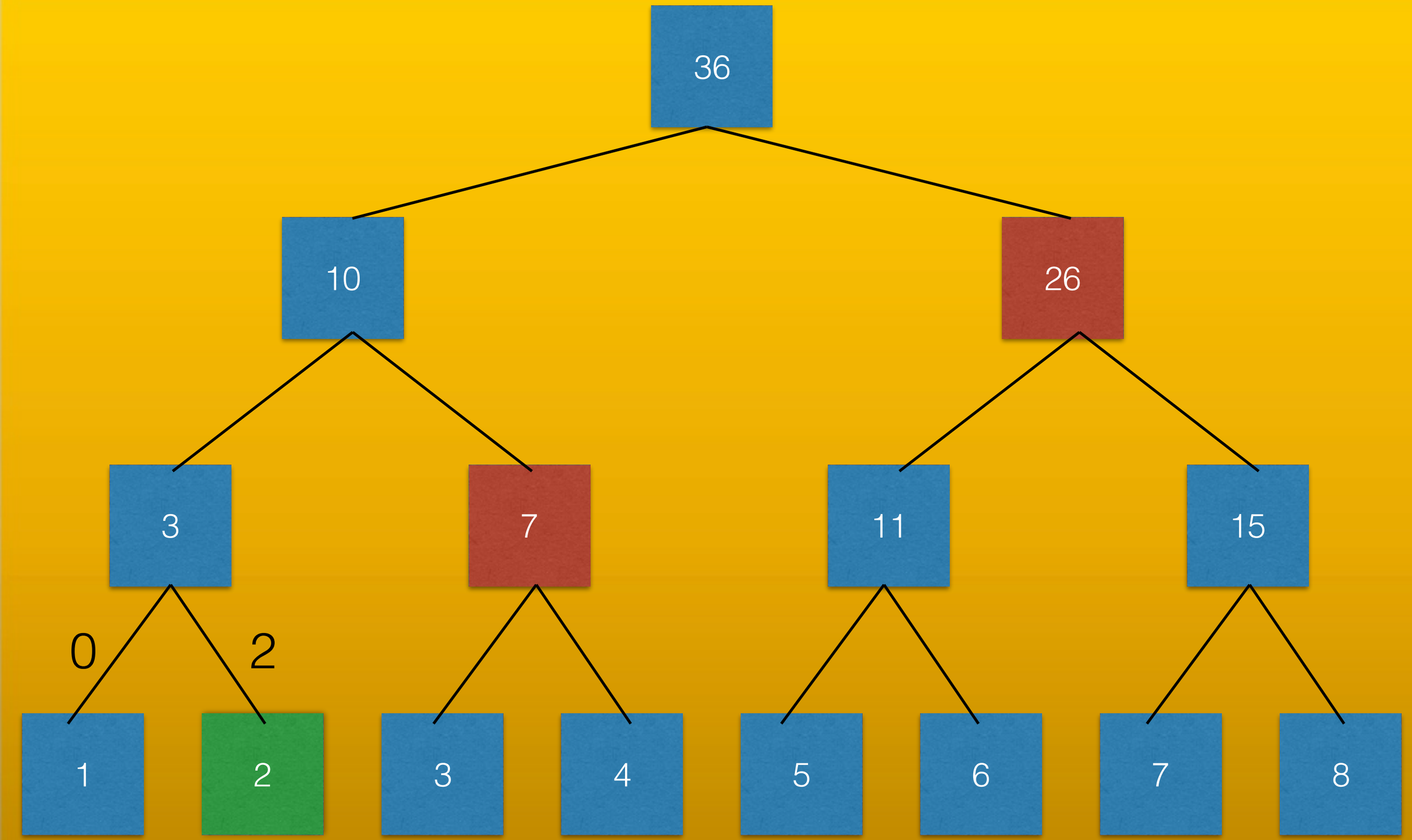

36
10
26
3
7
11
15
1
2
3
4
5
6
7
8

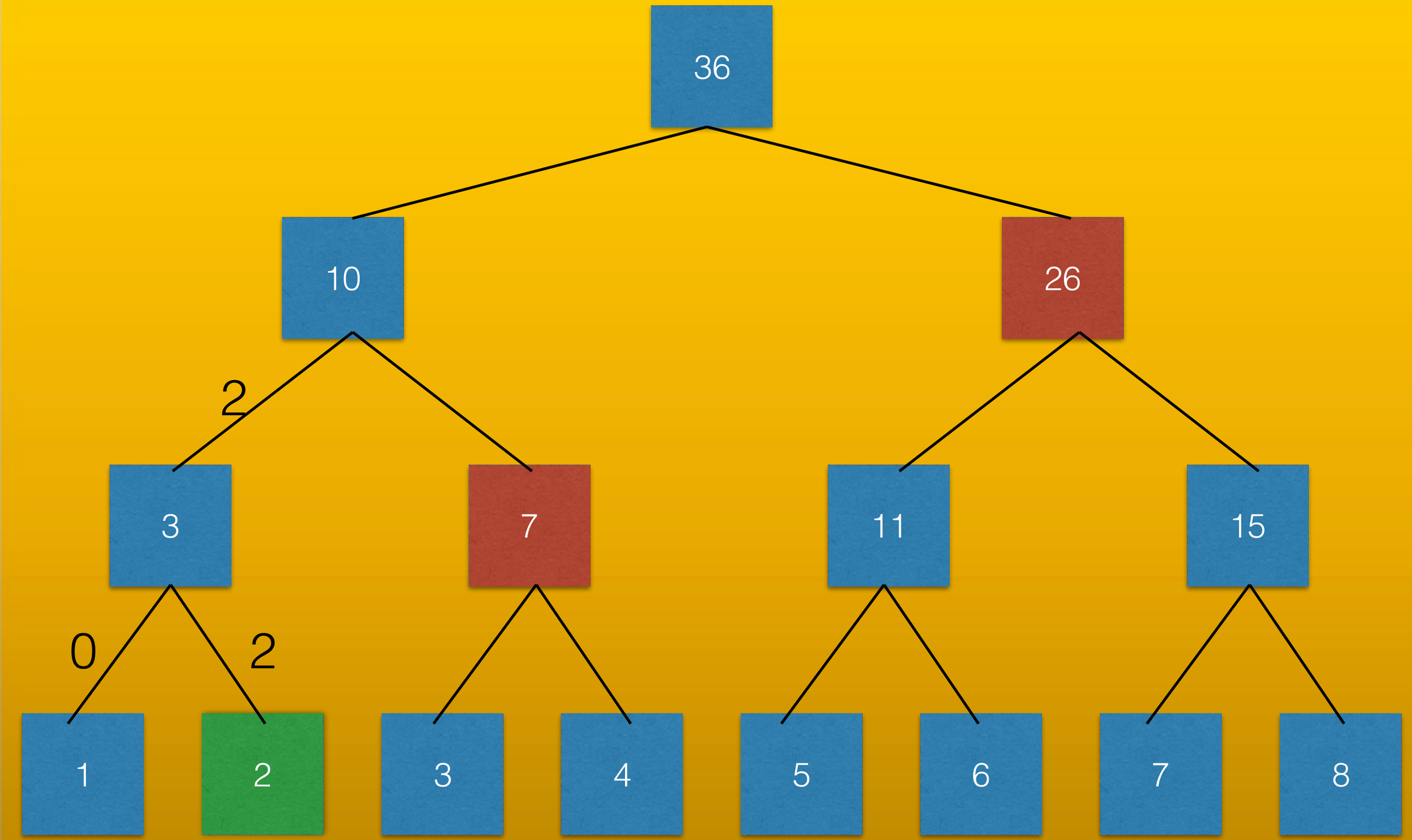

36
10
26
3
7
11
15
1
2
3
4
5
6
7
8

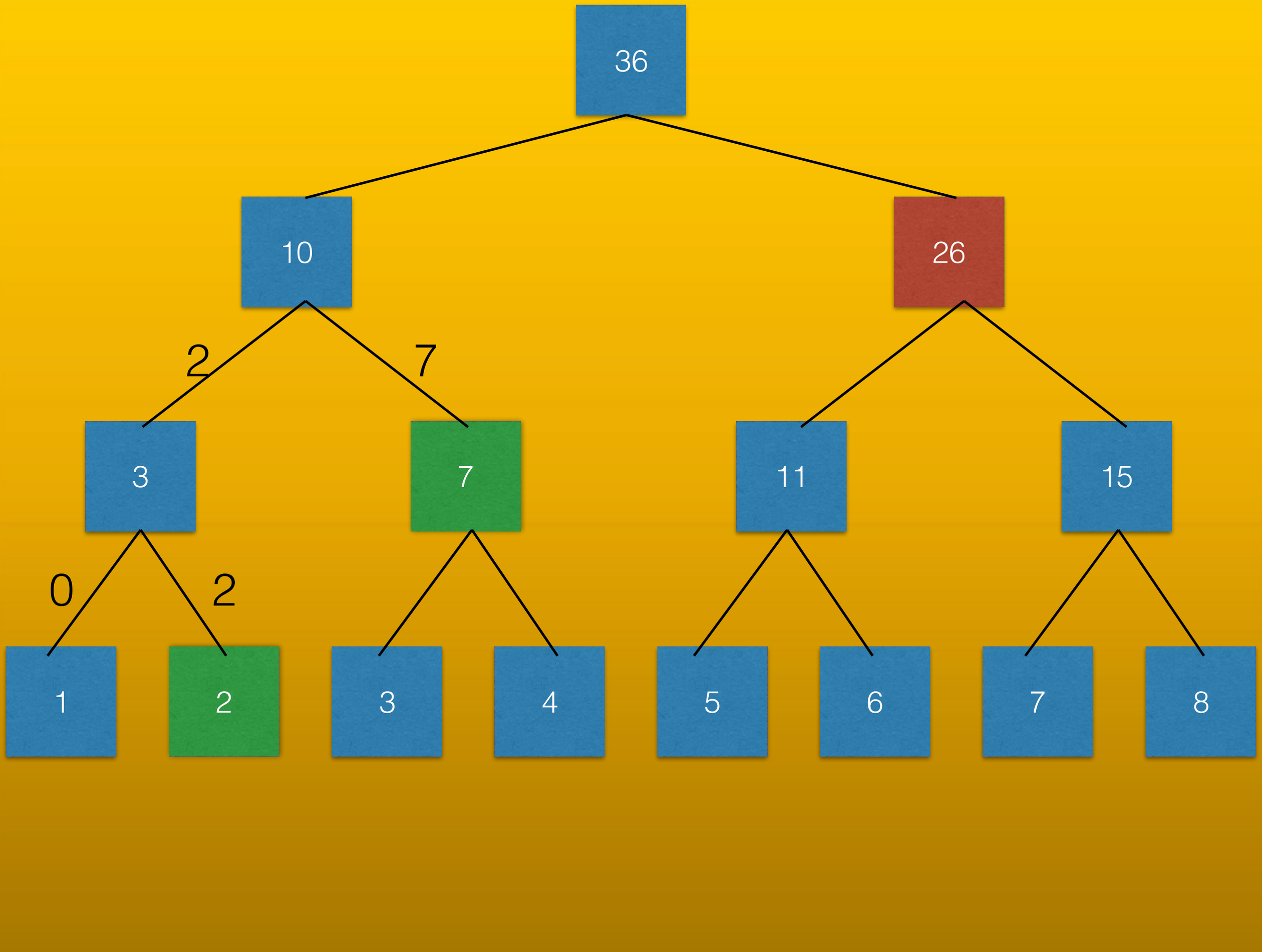

36
10
26
3
7
11
15
1
2
3
4
5
6
7
8

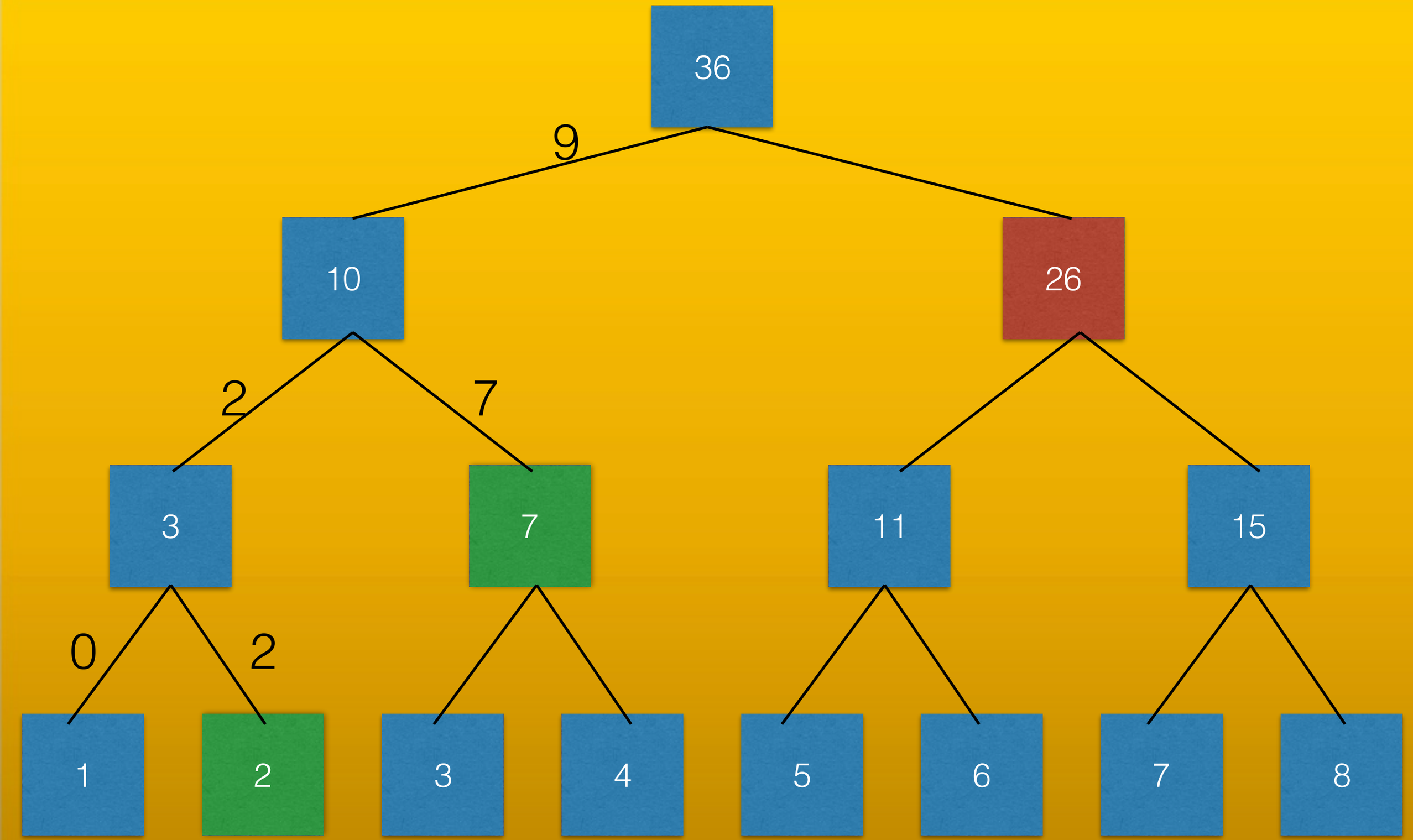

36
10
26
3
7
11
15
1
2
3
4
5
6
7
8

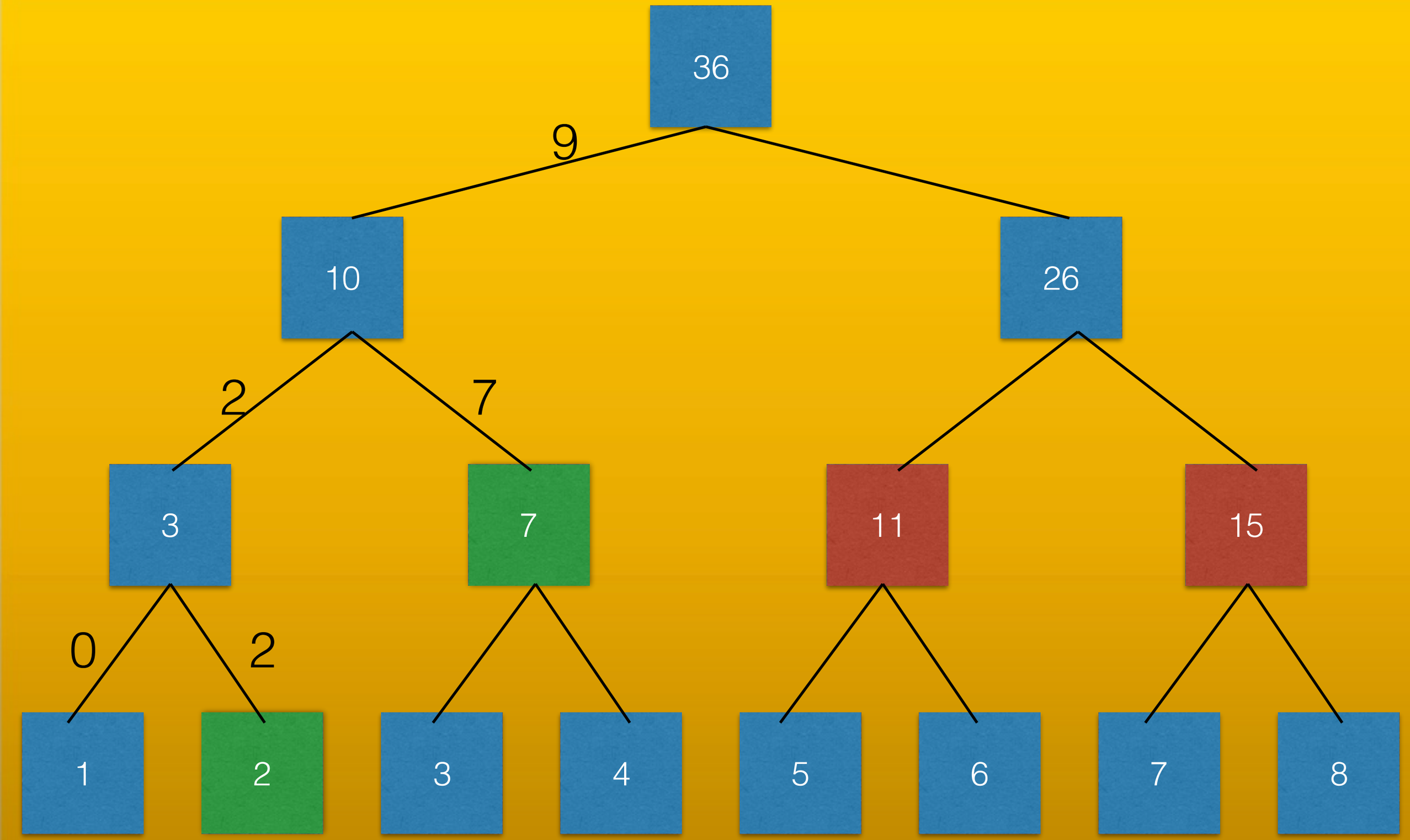

36
10
26
3
7
11
15
0
1
2
3
4
5
6
7
8

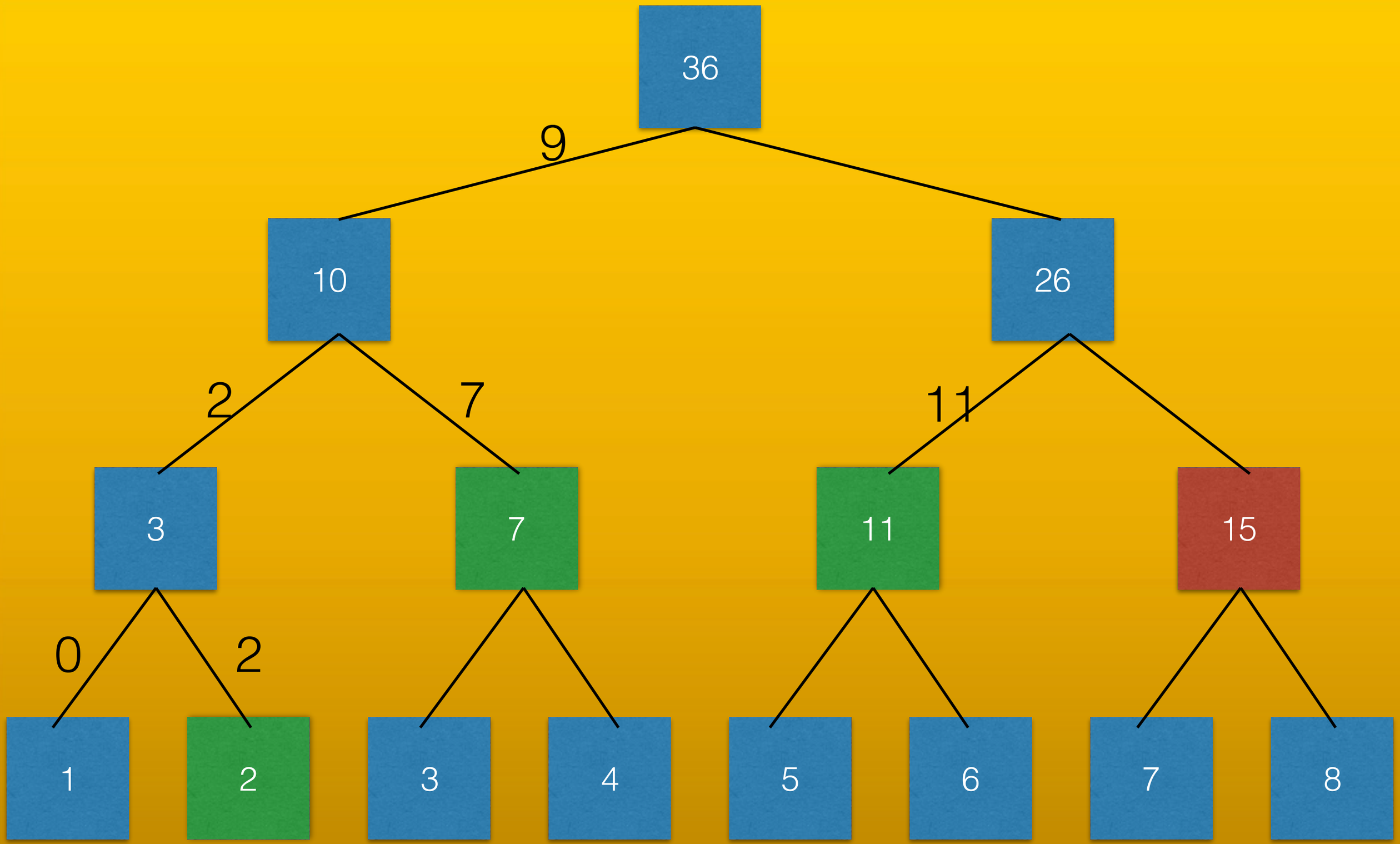

36
10
26
3
7
11
15
0
2
1
2
3
4
5
6
7
8

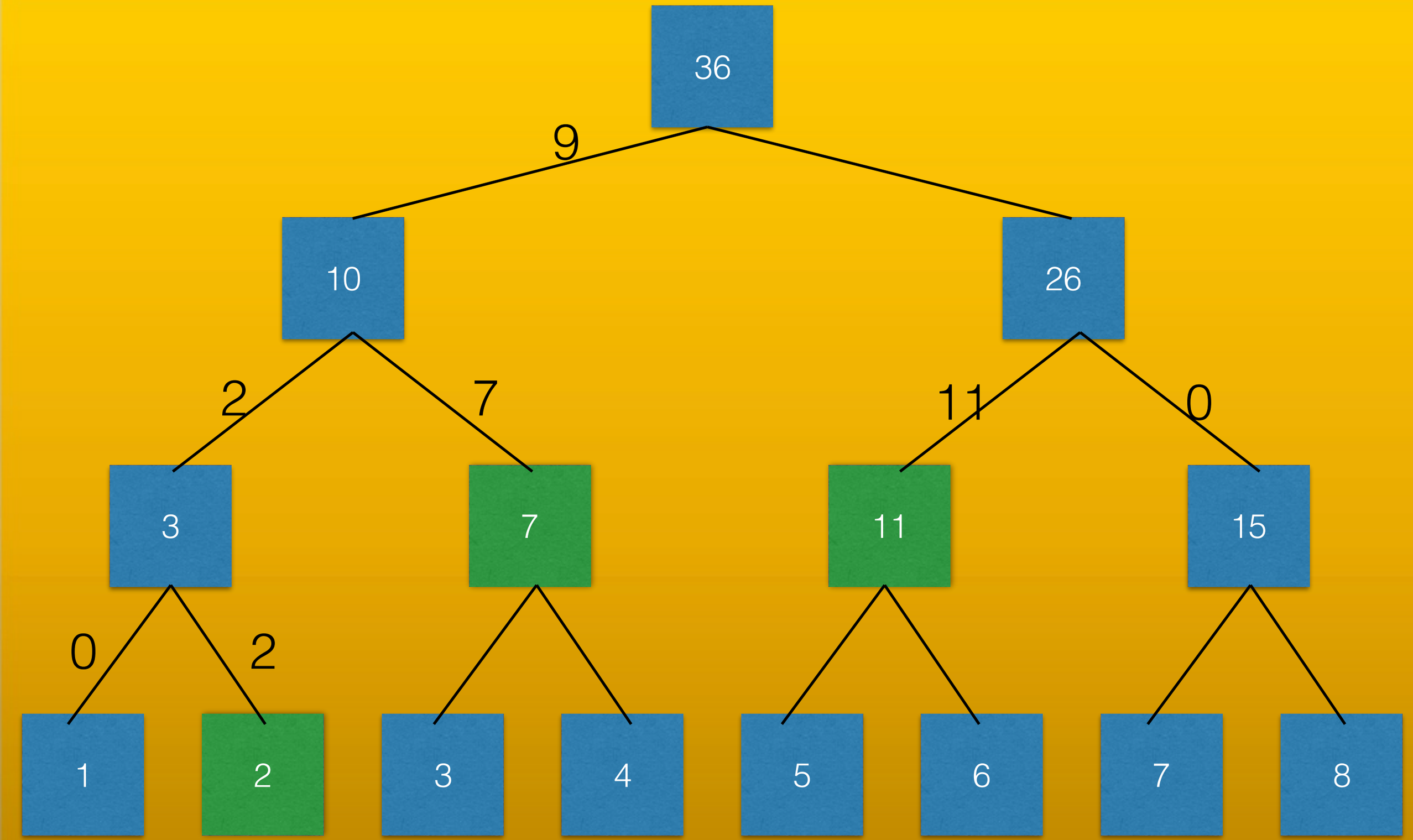

36
10
26
2
3
7
11
15
0
2
1
2
3
4
5
6
7
8

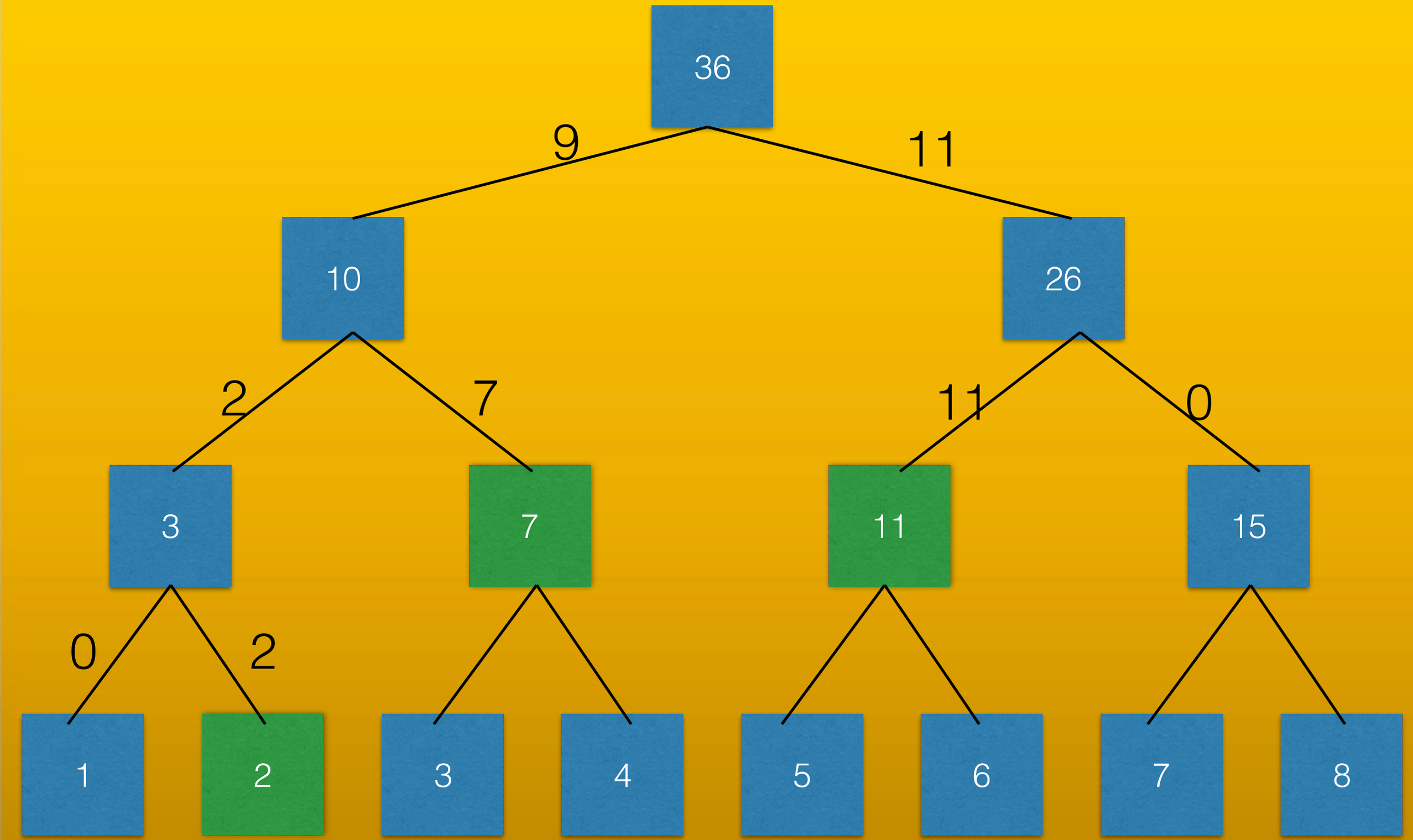

36
10
26
2
7
3
7
11
15
0
2
1
2
3
4
5
6
7
8


36
9
10
26
2
7
3
7
11
15
0
2
1
2
3
4
5
6
7
8


36
9
10
26
2
7
3
7
11
15
0
2
1
2
3
4
5
6
7
8


36
9
10
26
2
7
11
3
7
11
15
0
2
1
2
3
4
5
6
7
8


36
9
10
26
2
7
11
0
3
7
11
15
0
2
1
2
3
4
5
6
7
8


36
9
11
10
26
2
7
11
0
3
7
11
15
0
2
1
2
3
4
5
6
7
8

query(now, L, R, x, y)

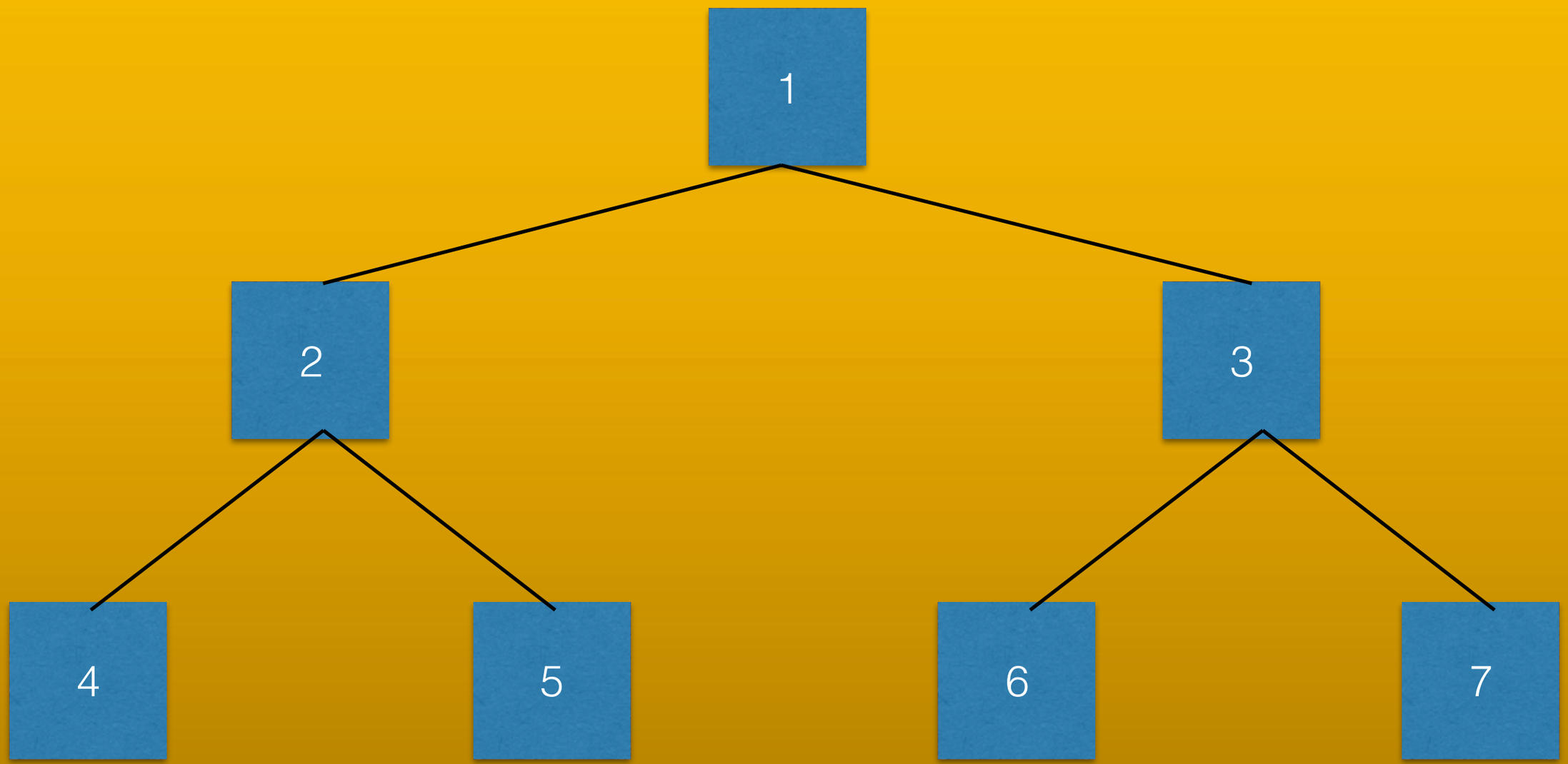
(x,y) = range yang diquery

(L,R) = range yang dicover oleh node now

query(now, L, R, x, y)

```
if (x <= L && R <= y) {
   // if everything is inside range query
   return node[now];
}
if (y < L || R < x) {
   // if not intersect with range query
   return 0;
}
int M = (L + R) >> 1;
return query(now.L, L, M, x, y)
     + query(now.R, M + 1, R, x, y);
```

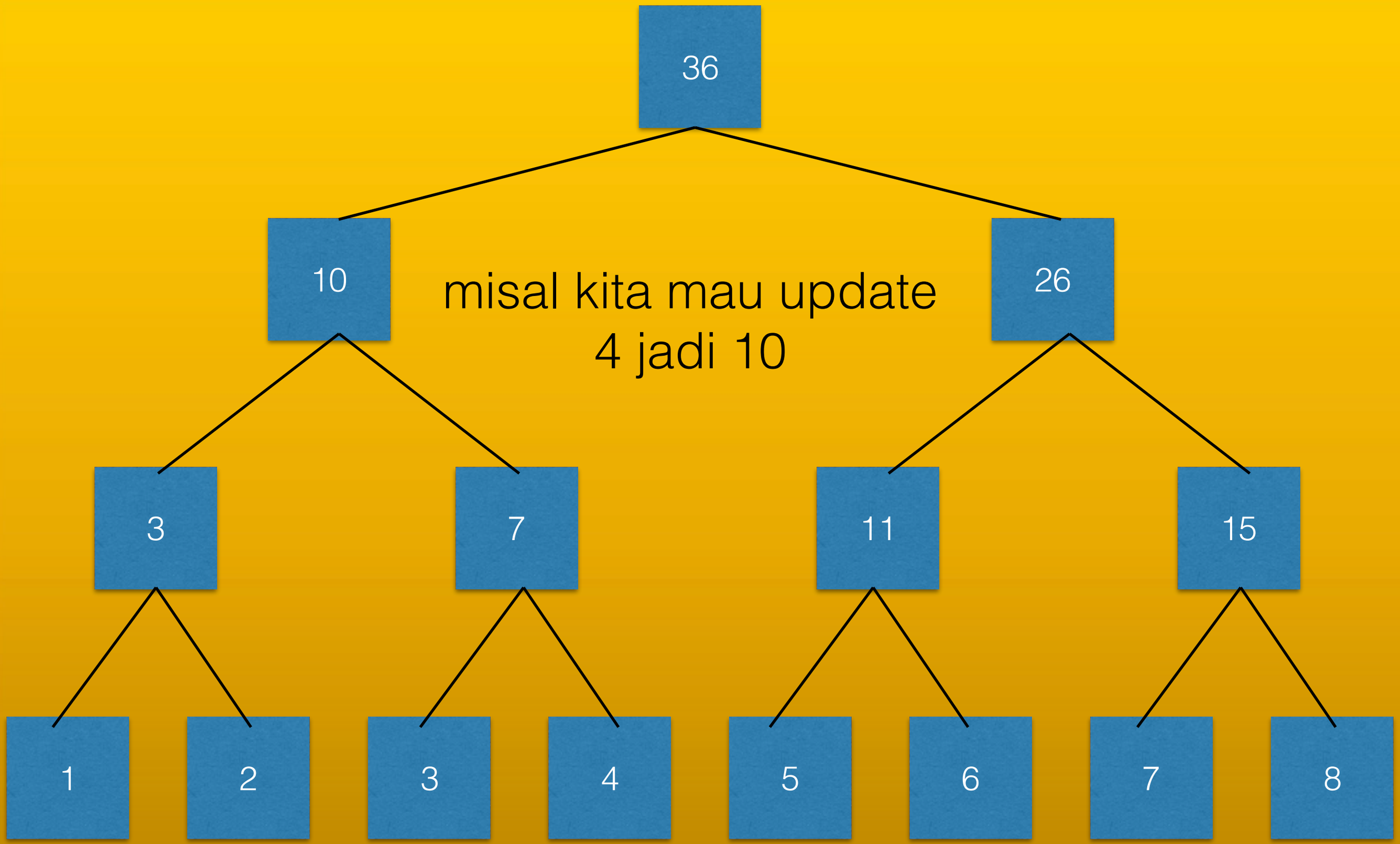
kalau kalian nomerin node2nya pake “BFS”



query(now, L, R, x, y)

```
if (x <= L && R <= y) {
   // if everything is inside range query
   return node[now];
}
if (y < L || R < x) {
   // if not intersect with range query
   return 0;
}
int M = (L + R) >> 1;
return query(now*2, L, M, x, y)
     + query(now*2 + 1, M + 1, R, x, y);
```

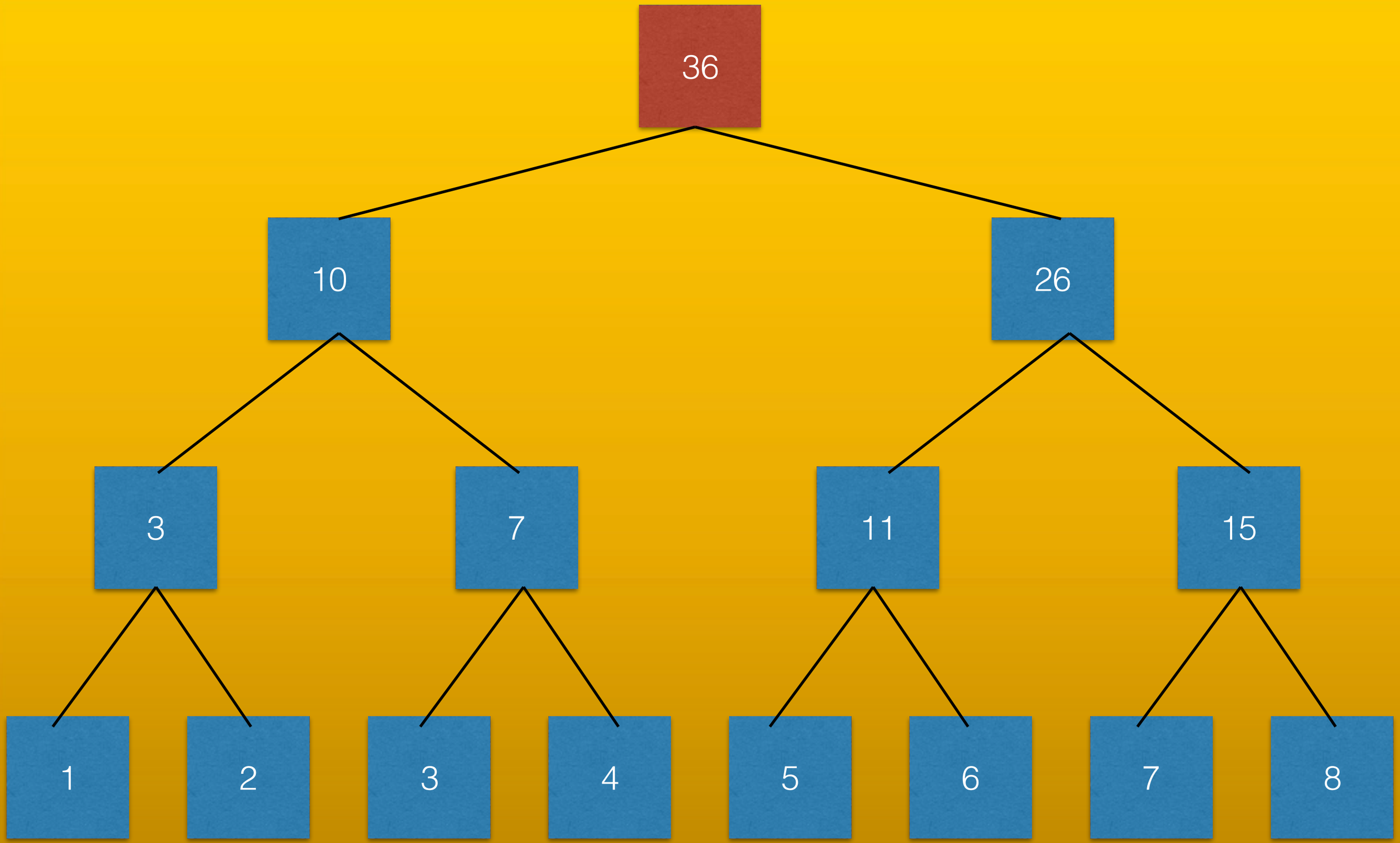
gimana cara updatenya?

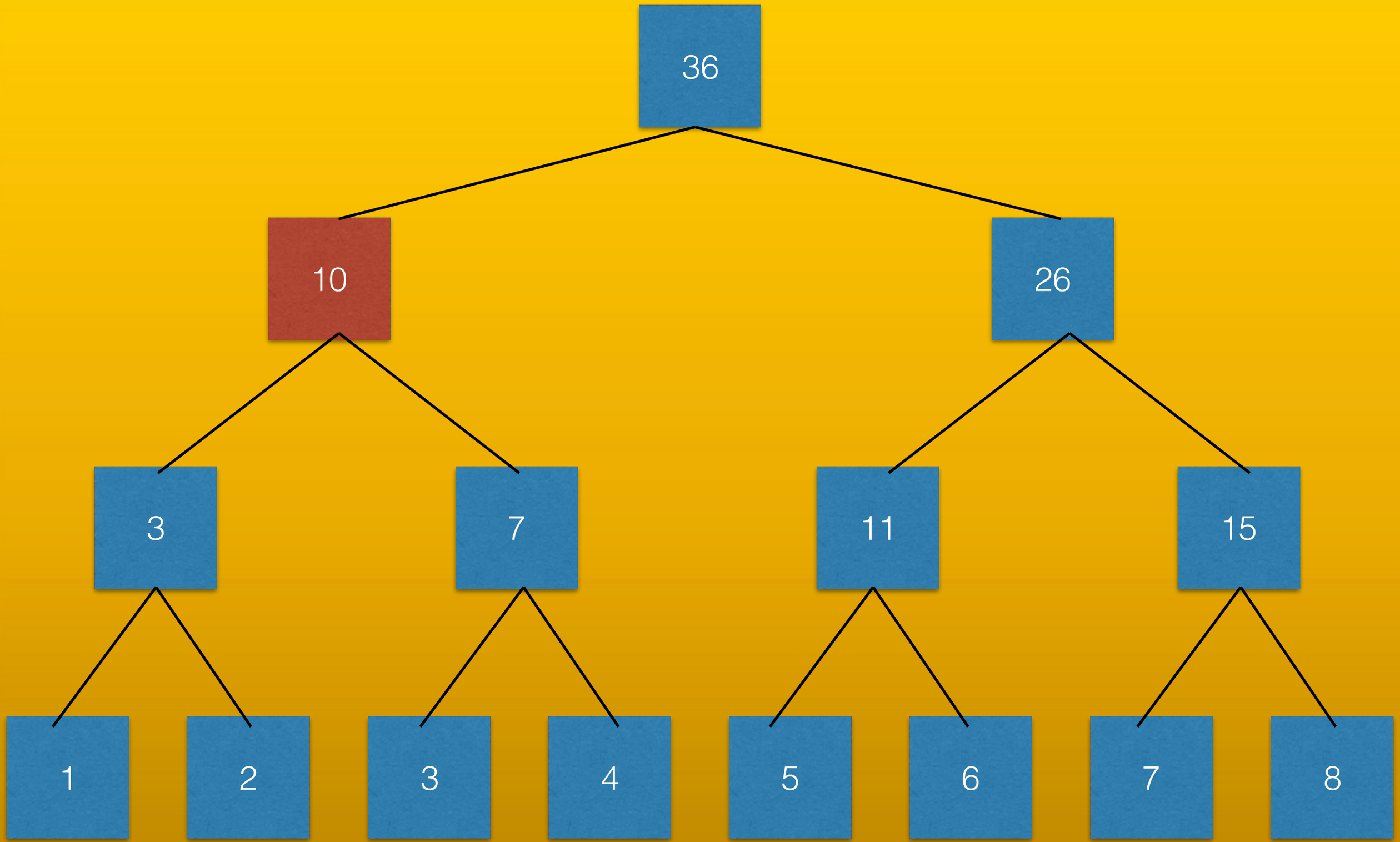

36
10
26
misal kita mau update
4 jadi 10
3
7
11
15
1
2
3
4
5
6
7
8

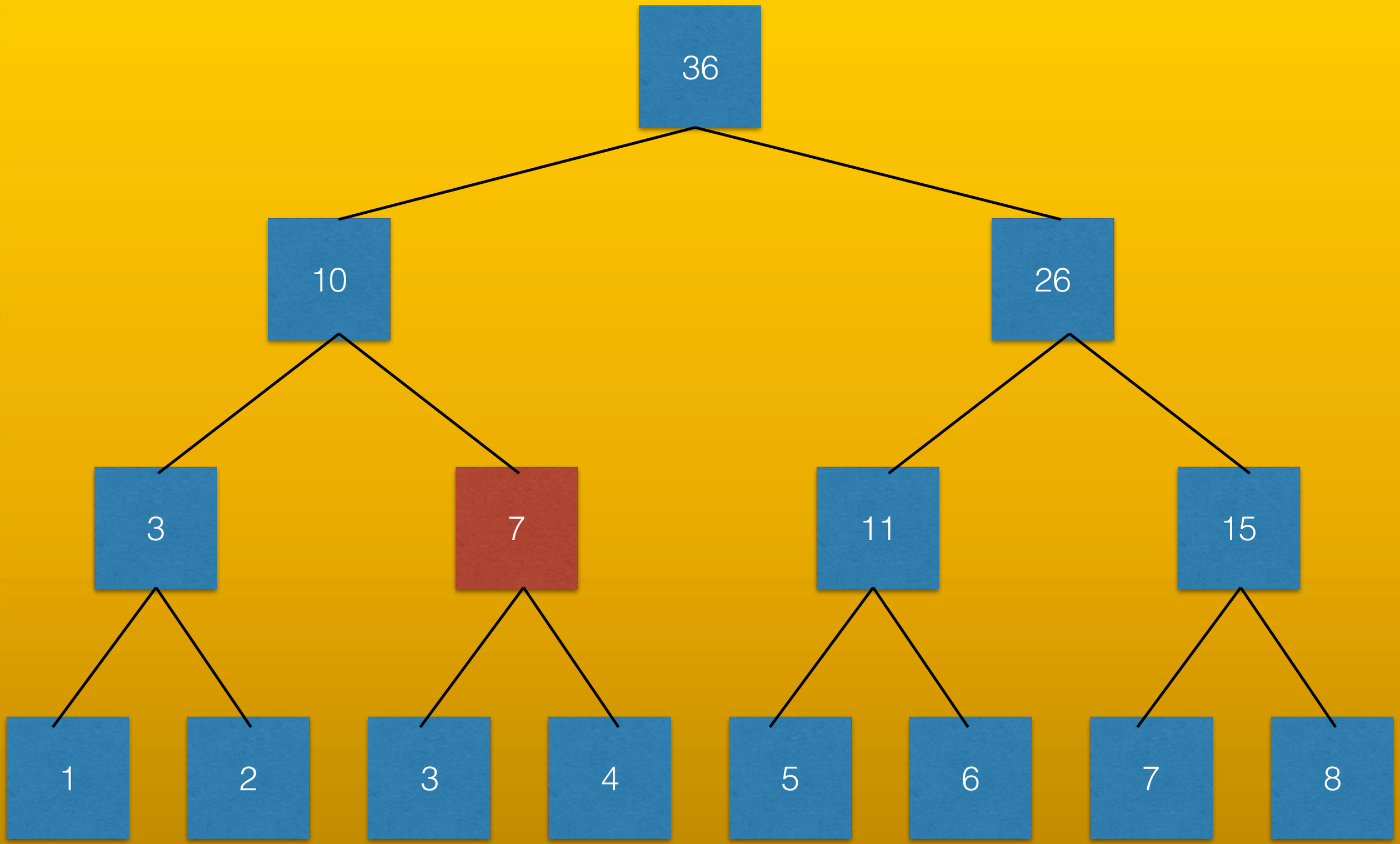

42
16
26
node2 merah ini
harus keupdate juga kan
3
13
11
15
1
2
3
10
5
6
7
8

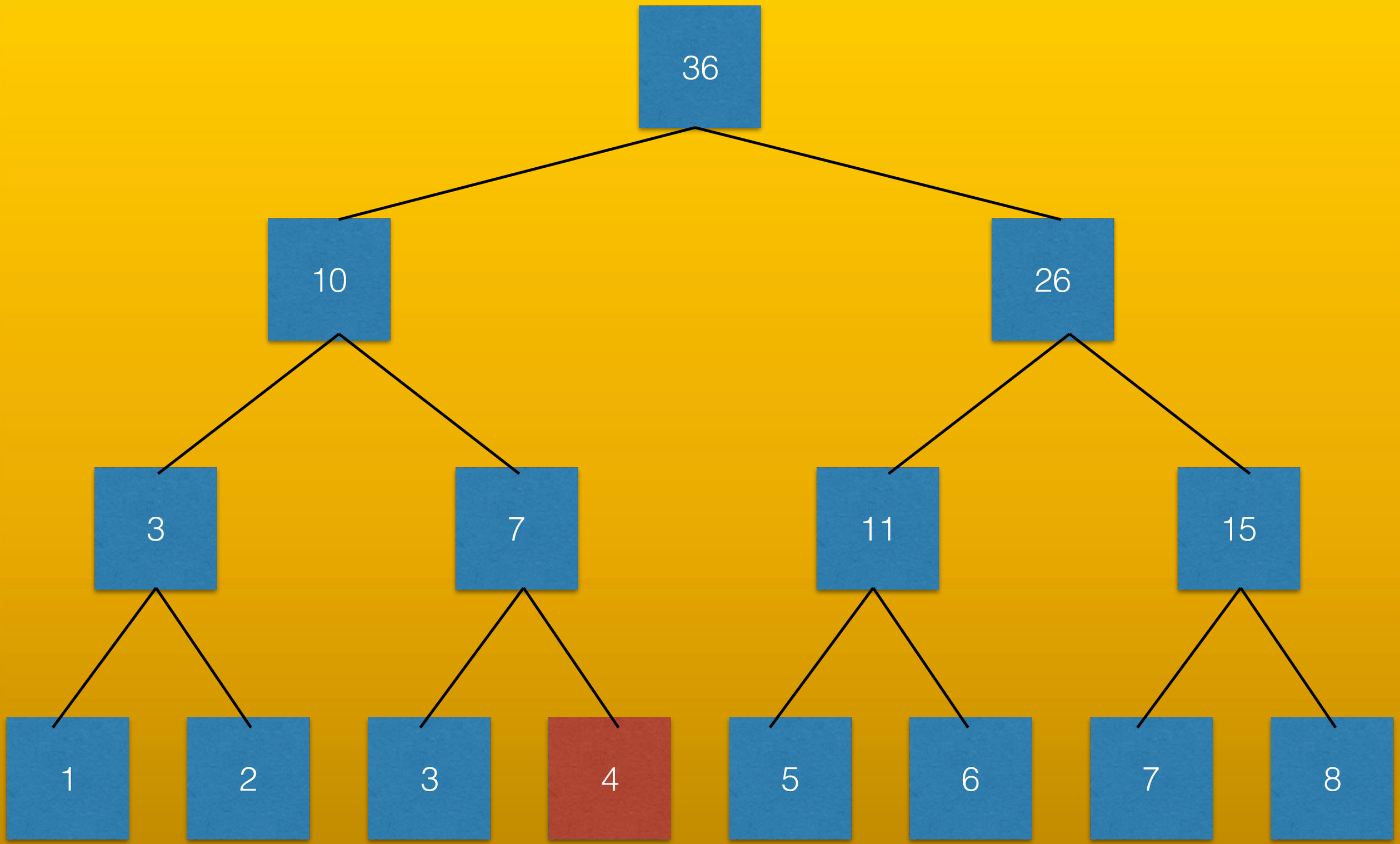
kita nyari path kebawah,

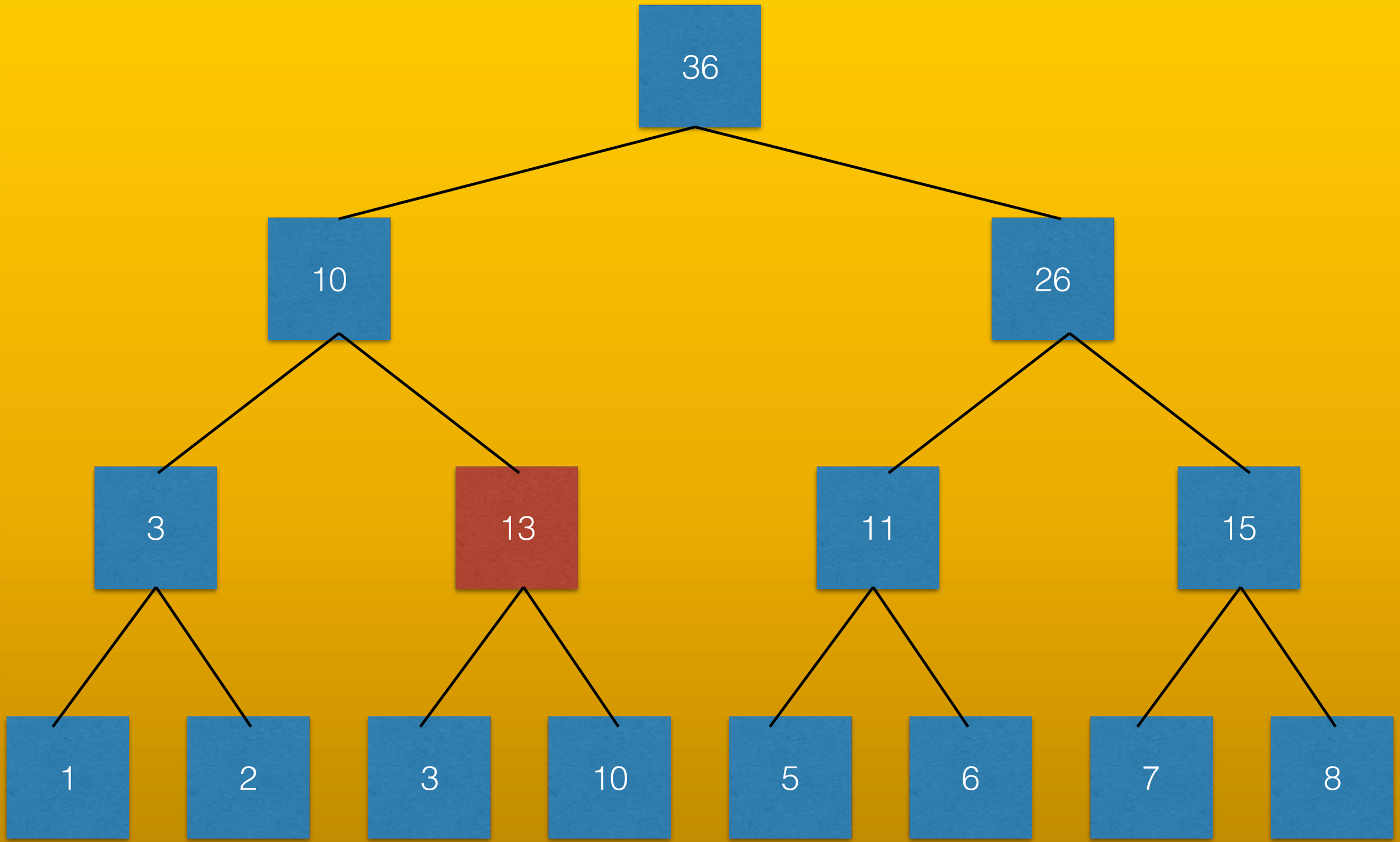
sambil update

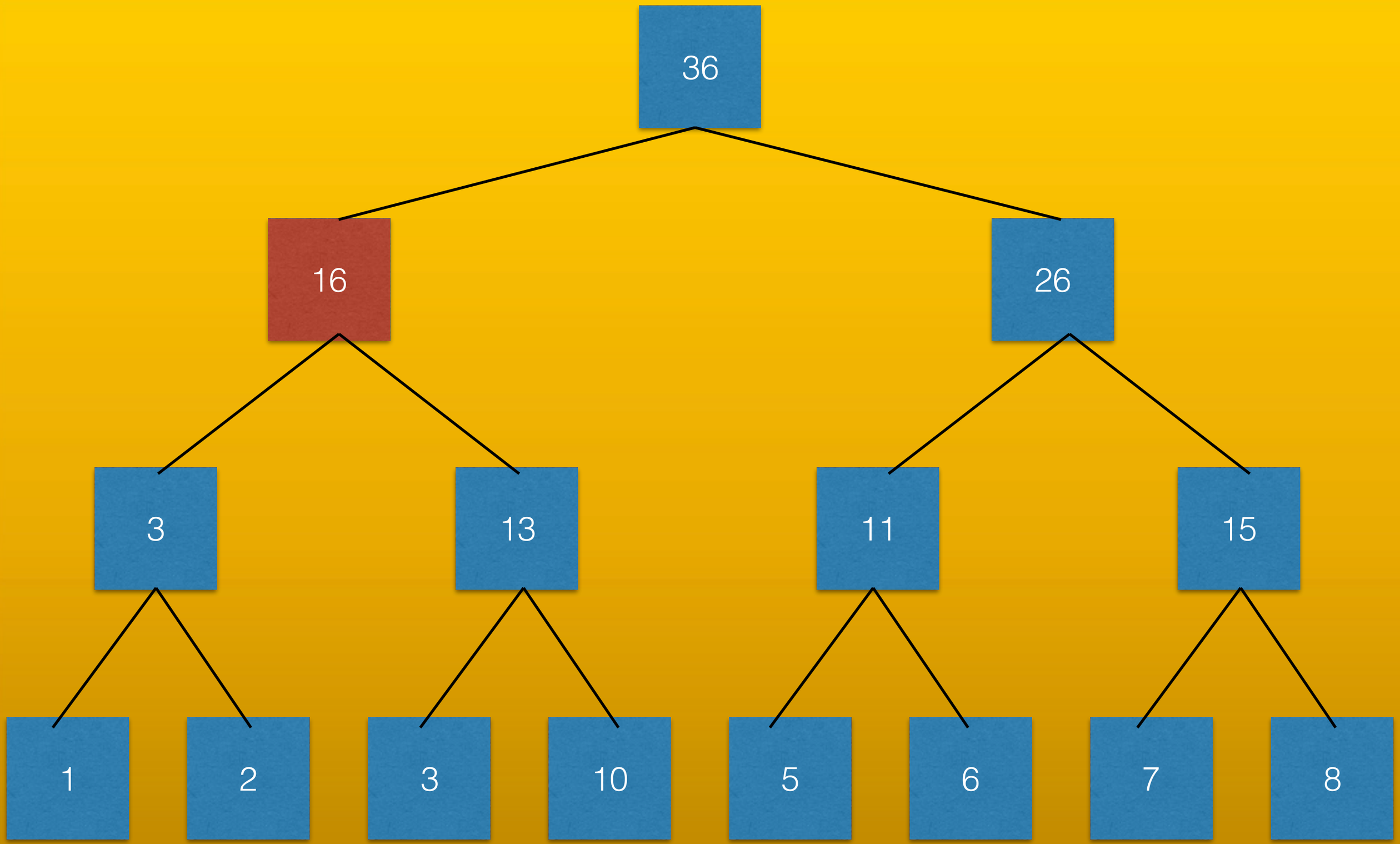

36
10
26
3
7
11
15
1
2
3
4
5
6
7
8

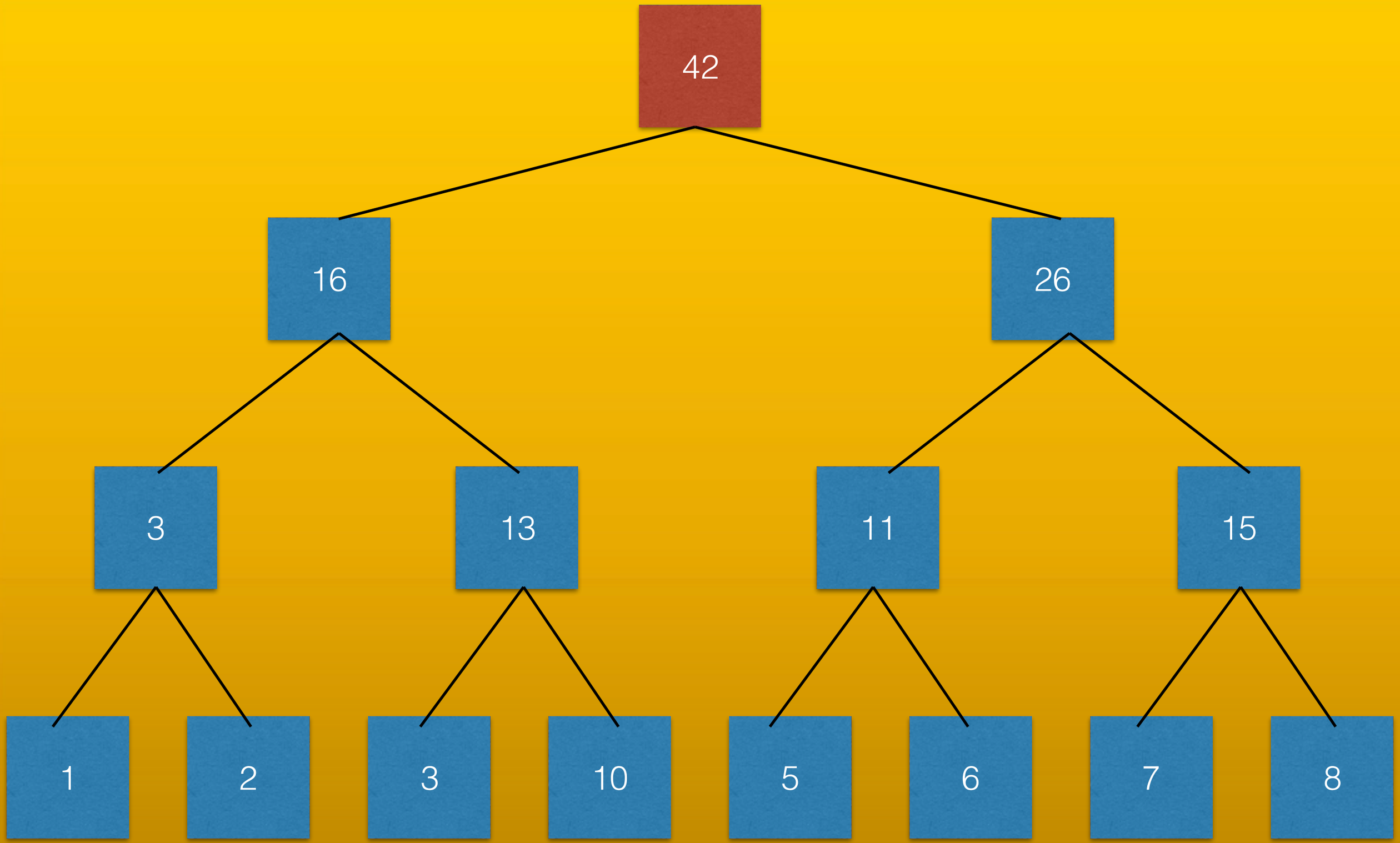

36
10
26
3
7
11
15
1
2
3
4
5
6
7
8


36
10
26
3
7
11
15
1
2
3
4
5
6
7
8


36
10
26
3
7
11
15
1
2
3
4
5
6
7
8


36
10
26
3
13
11
15
1
2
3
10
5
6
7
8


36
16
26
3
13
11
15
1
2
3
10
5
6
7
8


42
16
26
3
13
11
15
1
2
3
10
5
6
7
8

update(now, L, R, pos, val)

```
if (L == R) {
  tree[now] = val;
}
int M = (L + R) >> 1;
if (pos <= M) {
  update(now*2, L, M, pos, val);
} else {
  update(now*2+1, M+1, R, pos, val);
}
tree[now] = tree[now*2] + tree[now*2+1]
```

nah buat build :

1. bisa update satu2

2. bisa modify dikit dari kode updateya

build(now, L, R)

```
if (L == R) {
  tree[now] = data[L];
  return;
}
int M = (L + R) >> 1;
build(now*2, L, M);
build(now*2+1, M+1, R);
tree[now] = tree[now*2] + tree[now*2+1]
```

harus build kedua sisi

coba latihan dulu yah,
make sure you follow sebelum
getting more complicated

coba latihan dulu yah,
make sure you follow sebelum
getting more complicated

lagi, instead of cari sum, kita mau cari max

kalau mau cari max ditengah (bukan prefix max), gak bisa pake BIT

karena given
max(A[1] .. A[x]),
max(A[1] .. A[y]),
kita gak bisa deduce
max(A[x]..A[y])

basically mirip, tapi value sebuah node adalah **maksimum** dari value child2nya

update(now, L, R, pos, val)

```
if (L == R) {
  tree[now] = val;
  return;
}
int M = (L + R) >> 1;
if (pos <= M) {
  update(now*2, L, M, pos, val);
} else {
  update(now*2+1, M+1, R, pos, val);
}
tree[now] = max(tree[now*2],tree[now*2+1])
```

query(now, L, R, x, y)

```
if (x <= L && R <= y) {
   // if everything is inside range query
   return node[now];
}
if (y < L || R < x) {
   // if not intersect with range query
   return 0;
}
int M = (L + R) >> 1;
return max(query(now*2, L, M, x, y),
           query(now*2 + 1, M + 1, R, x, y);
```

coba yang agak susah dikit

querynya sekarang dikasih x dan y. tugas lu cari prefix dan suffix dari A[x..y] yang jumlahnya maksimum


10:00

tiap node simpan :

1. prefix maksimum di range

2. suffix maksimum di range

3. total elemen di range

sum
prefix
suffix

-4
4
0

-2
3
0

-2
6
0

3
3
3

-5
0
0

-1
6
0

-1
1
0

1
1
1

2
2
2

-1
0
0

-4
0
0

6
6
6

-7
0
0

1
1
1

-2
0
0

1

2

-1

-4

6

-7

1

-2

sum = left.sum + right.sum

prefix = ?

suffix = ?

sum = left.sum + right.sum

prefix = left.prefix

suffix = right.suffix

bener ga?

perlu consider case dimana max prefix nya itu motong perbatasannya

sum = left.sum + right.sum

prefix = max(left.prefix,**left.sum+right.prefix**)

suffix = max(right.suffix,**right.sum+left.suffix**)

update(now, L, R, pos, val)

```
if (L == R) {
   tree[now].sum = val;
   tree[now].pre = tree[now].suf = max(0,val);
   return;
}
int M = (L + R) >> 1;
if (pos <= M) {
   update(now*2, L, M, pos, val);
} else {
   update(now*2+1, M+1, R, pos, val);
}
tree[now].sum = tree[now*2].sum + tree[now*2+1].sum
tree[now].pre = max(tree[now*2].pre,
                    tree[now*2].sum+tree[now*2+1].pre)
tree[now].suf = max(tree[now*2+1].suf,
                    tree[now*2+1].sum+tree[now*2].suf)
```

query(now, L, R, x, y)

```
if (x <= L && R <= y) {
   return tree[now];
}
if (R < x || y < L) {
   return NULL;
}
int M = (L + R) >> 1;
left = query(now*2, L, M, x, y);
right = query(now*2+1, M+1, R, x, y);
if (left == NULL) return right;
if (right == NULL) return left;
temp.sum = left.sum + right.sum
temp.pre = max(left.pre,left.sum+right.pre);
temp.suf = max(right.suf,right.sum+left.suf);
return temp;
```

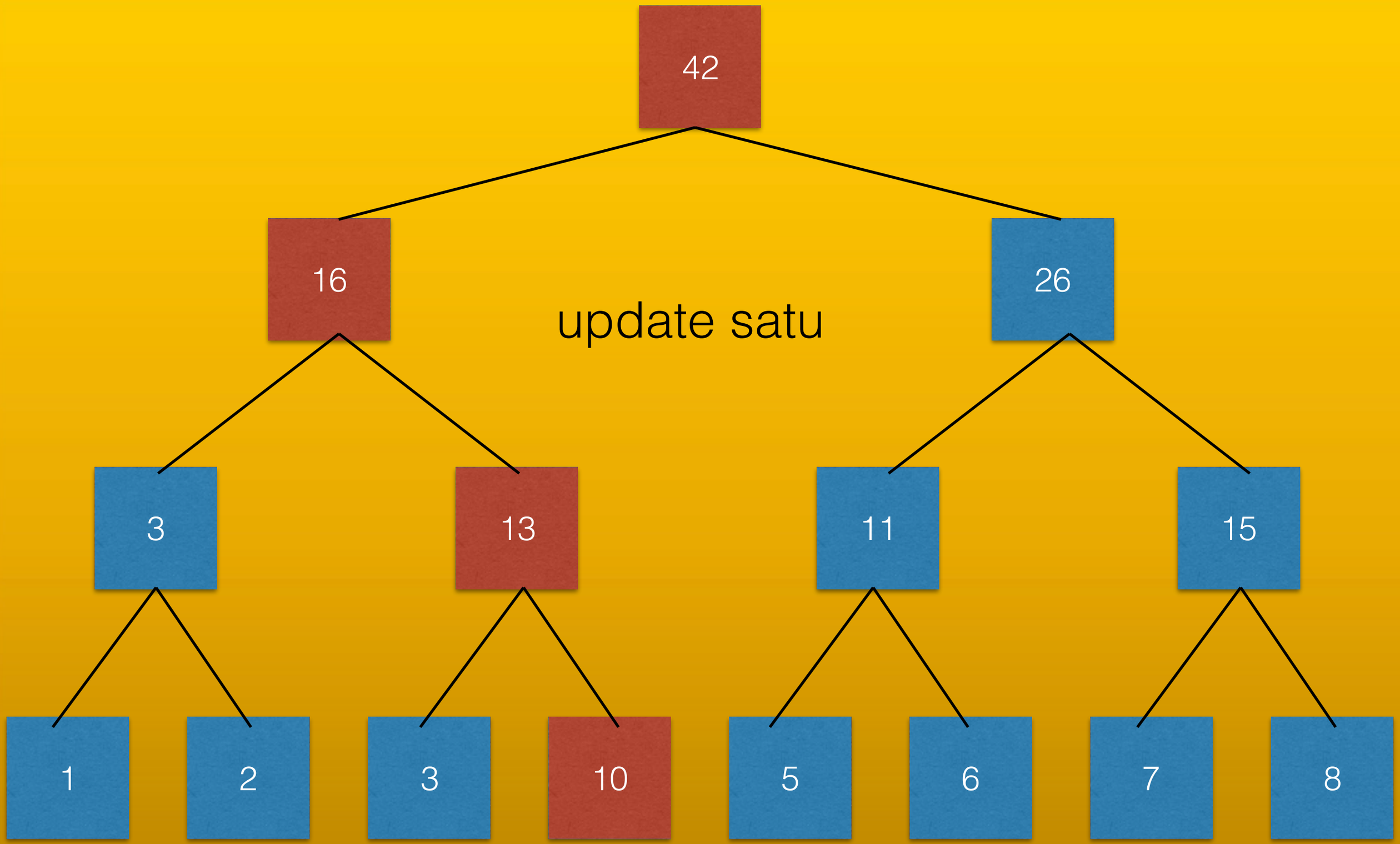
sebelum lanjut,
pertanyaan dulu?

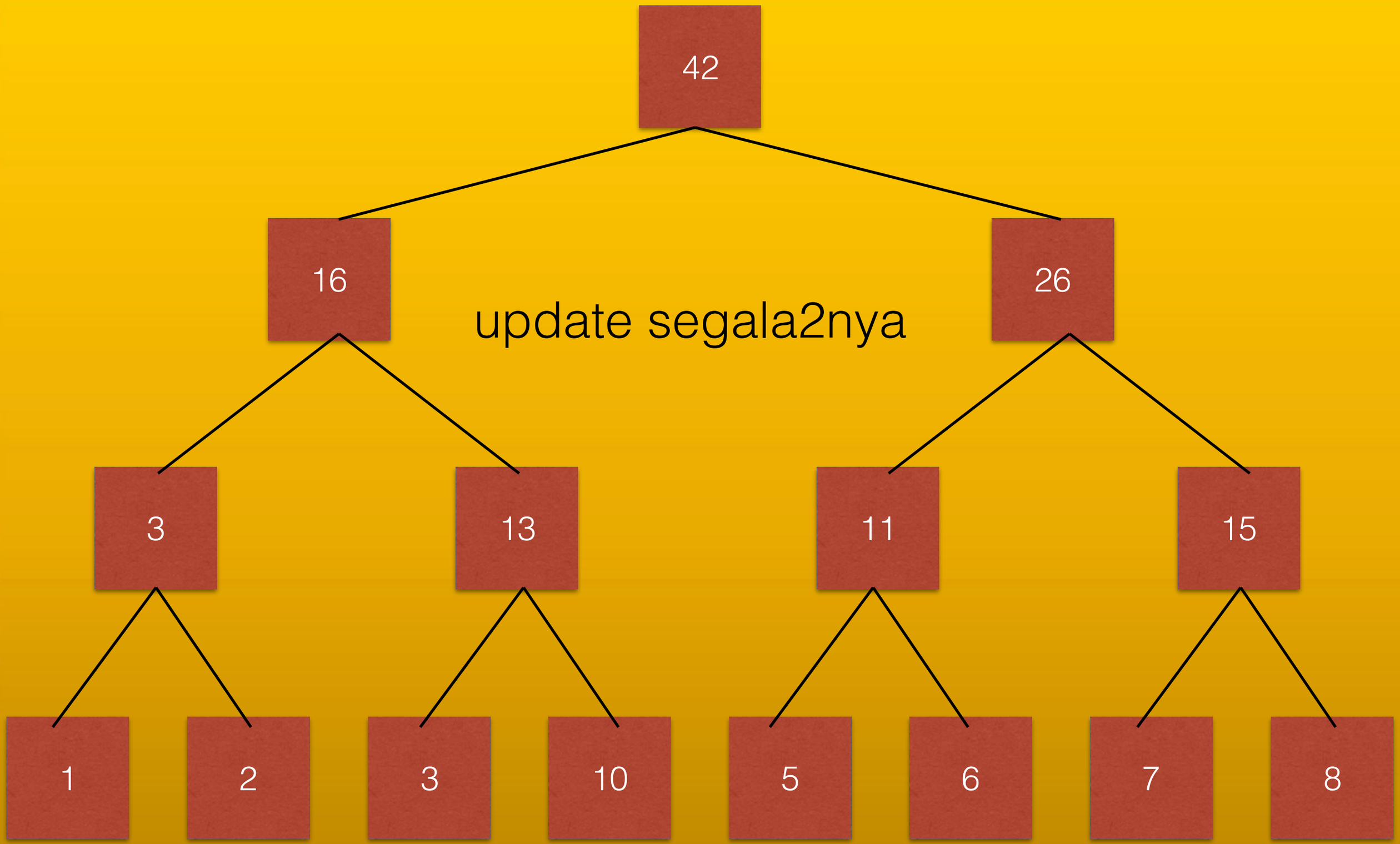
karena abis ini agak sedikit
susah

ini baru updatenya single point, tapi kan tadi problem statementnya

dikasih array N, dikasih query Q. tiap query bisa either update isi array **(bisa range)** atau cari jumlah dari sebuah range

apa yang terjadi kalau lu update segala2nya (yang ada di dunia ini)


42
16
26
update satu
3
13
11
15
1
2
3
10
5
6
7
8


42
16
26
update segala2nya
3
13
11
15
1
2
3
10
5
6
7
8

bisa O(n) per update,

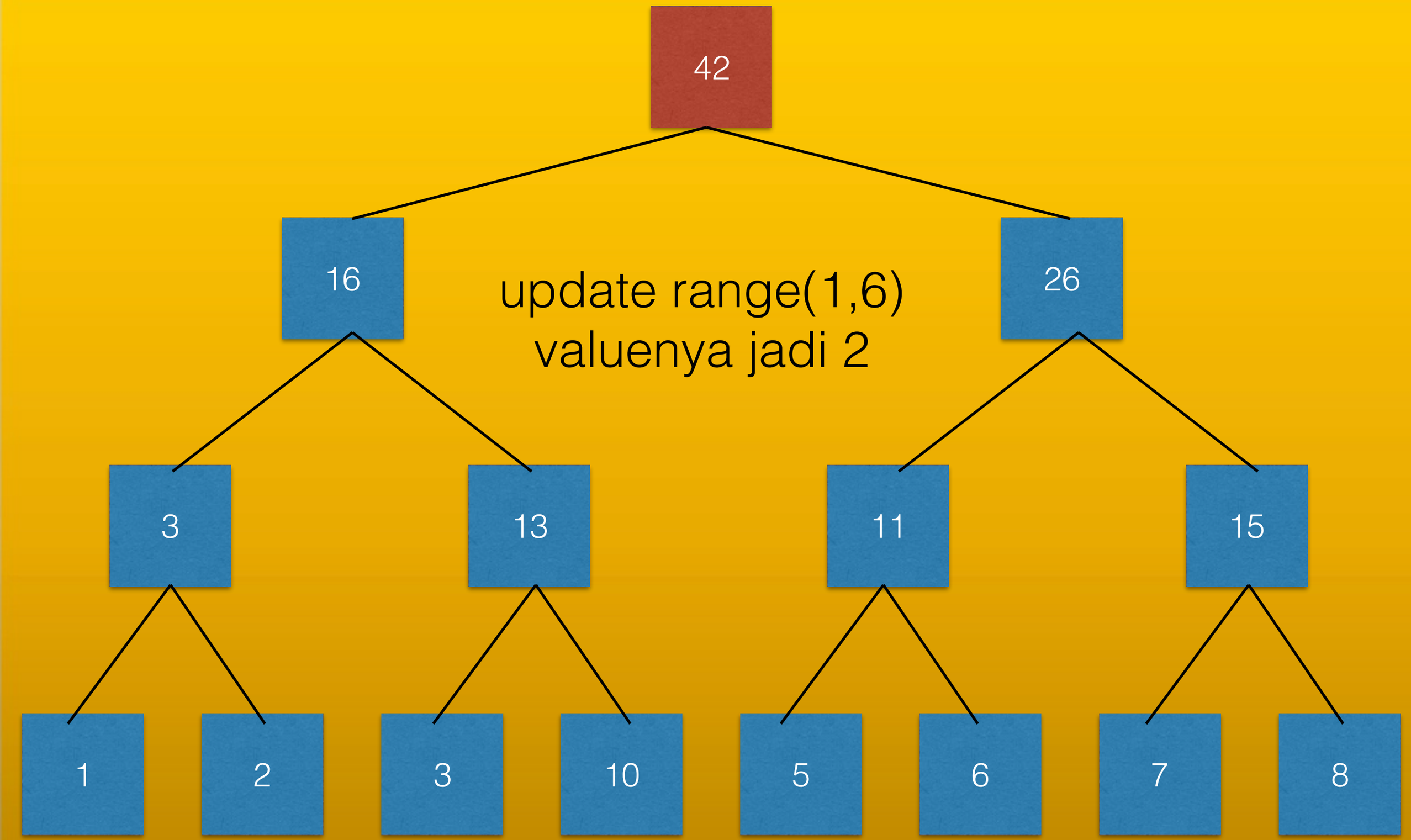
not better than bruteforce

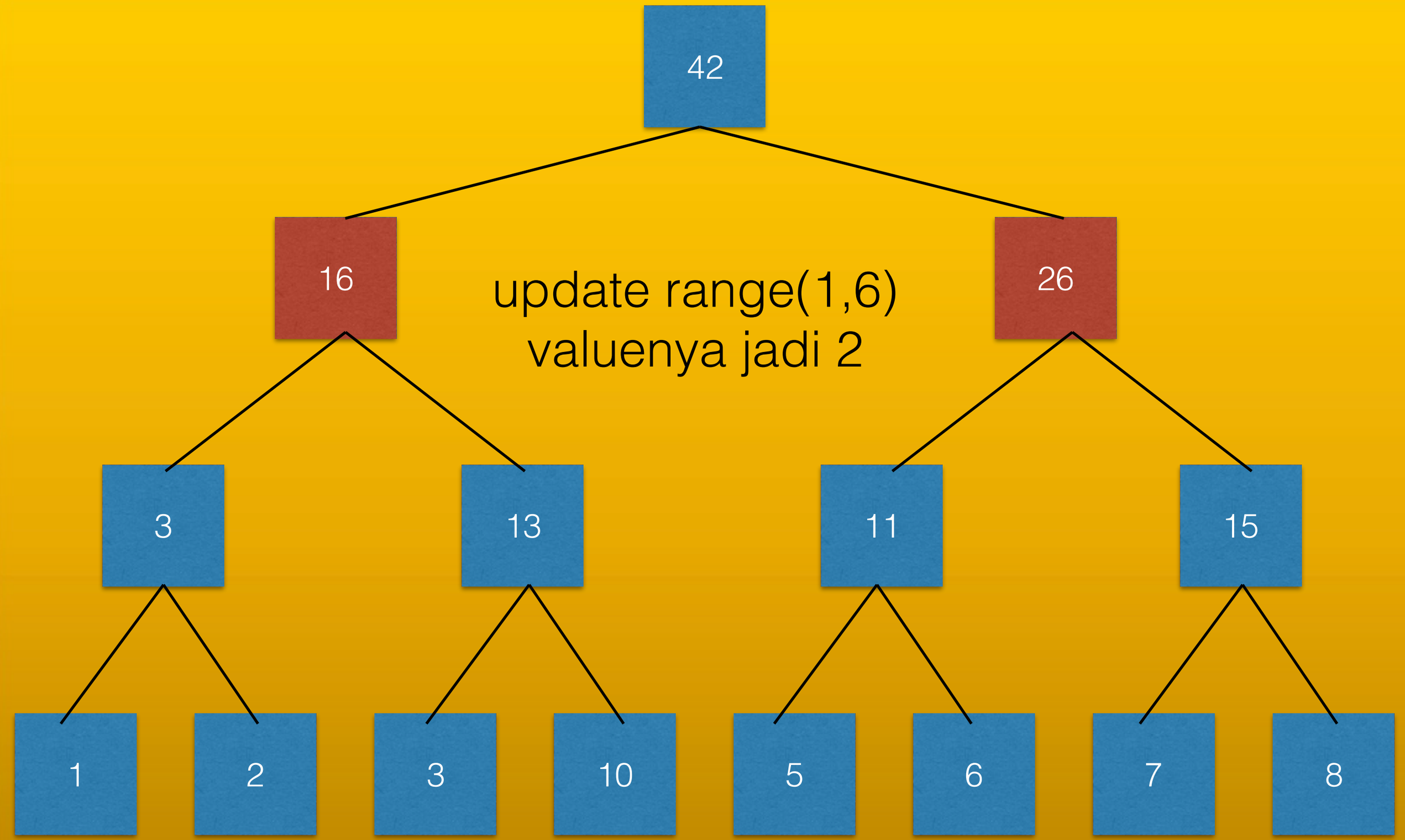
pake teknik yang namanya

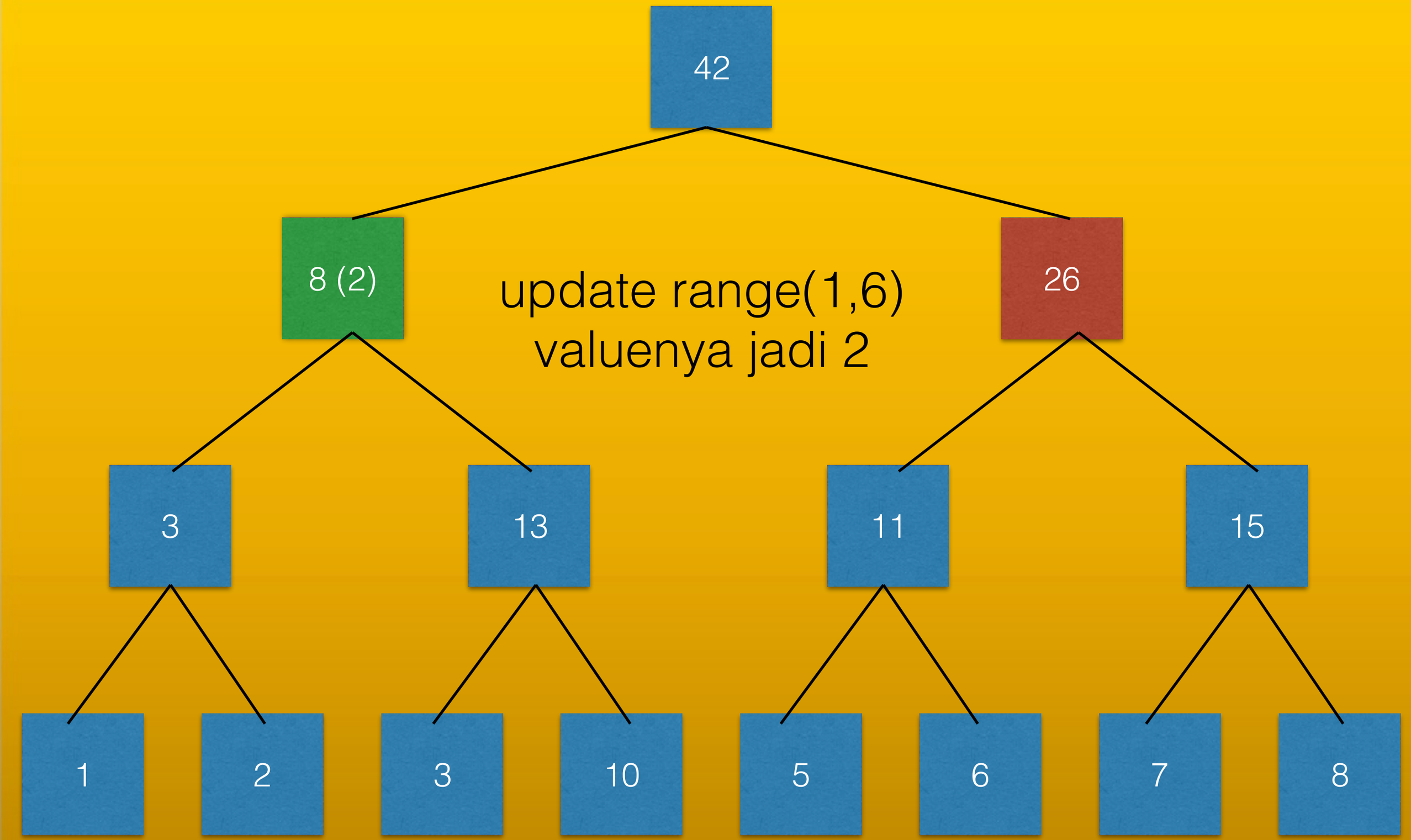
lazy propagation

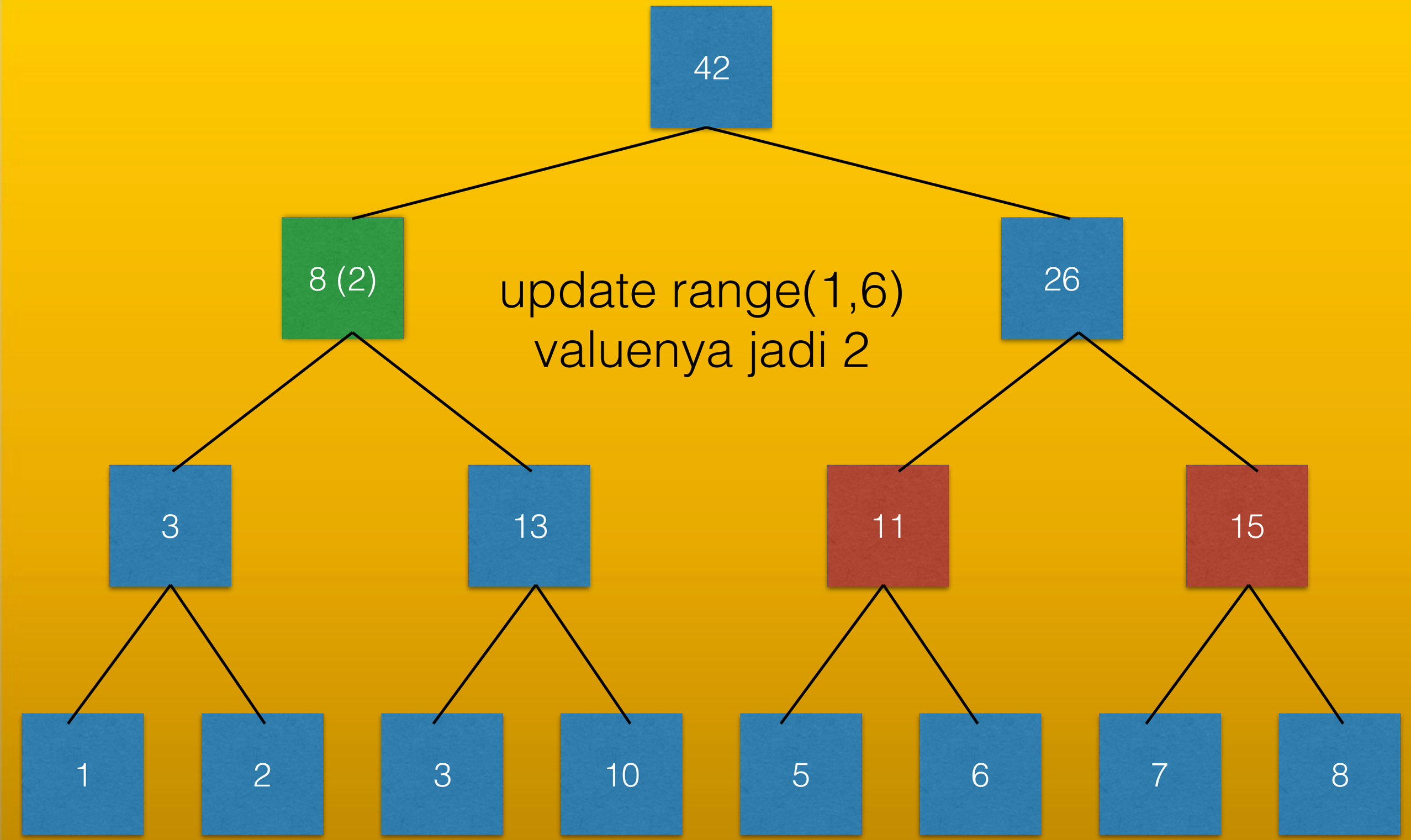
tiap node punya dua value :

1. sum dari range
2. update yang disimpen


42
16
26
update range(1,6)
valuenya jadi 2
3
13
11
15
1
2
3
10
5
6
7
8


42
16
26
update range(1,6)
valuenya jadi 2
3
13
11
15
1
2
3
10
5
6
7
8


42
8 (2)
26
update range(1,6)
valuenya jadi 2
3
13
11
15
1
2
3
10
5
6
7
8


42
8 (2)
26
update range(1,6)
valuenya jadi 2
3
13
11
15
1
2
3
10
5
6
7
8

42

8 (2)

26

update range(1,6)
valuenya jadi 2

3

13

4 (2)

15

1

2

3

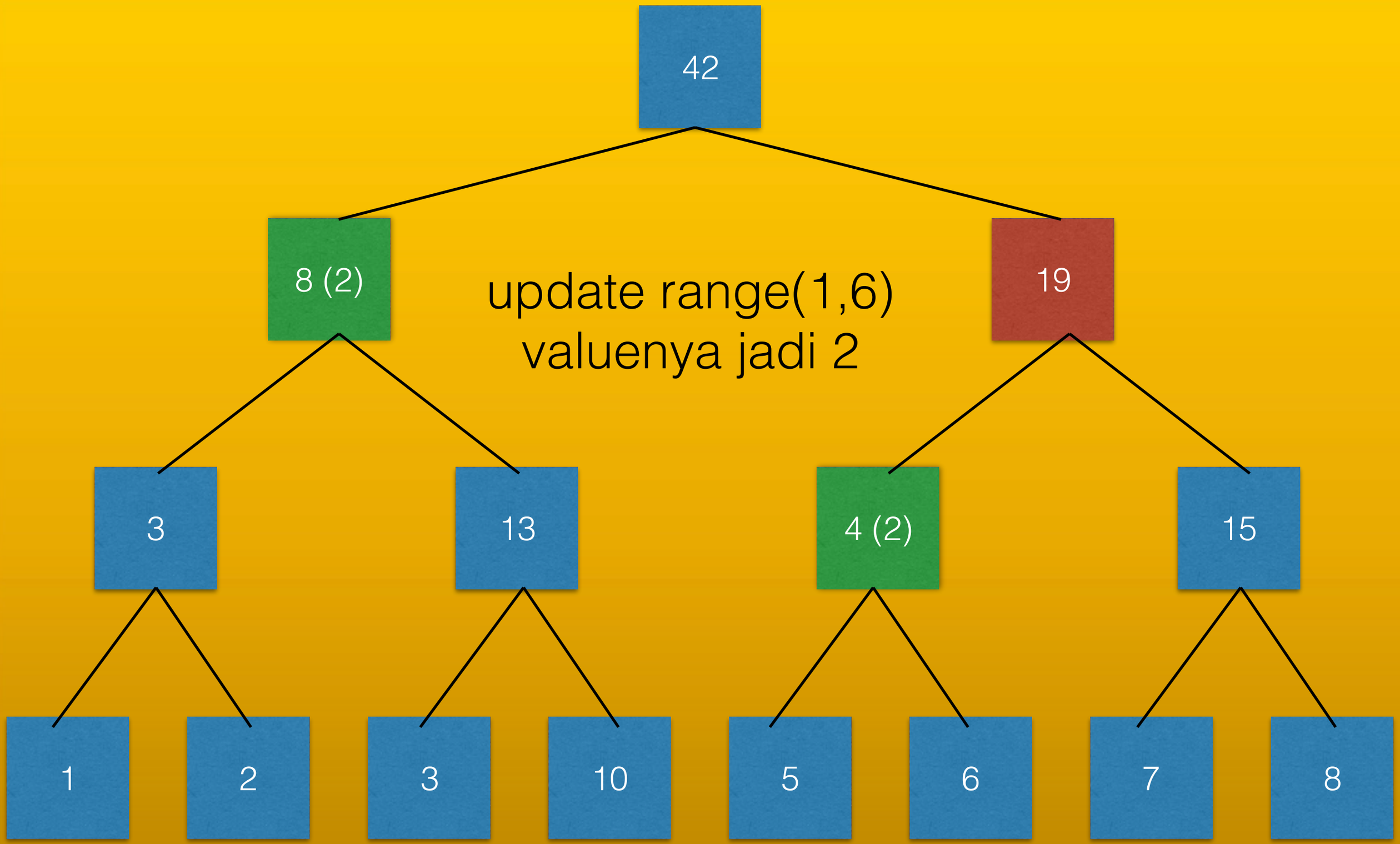
10

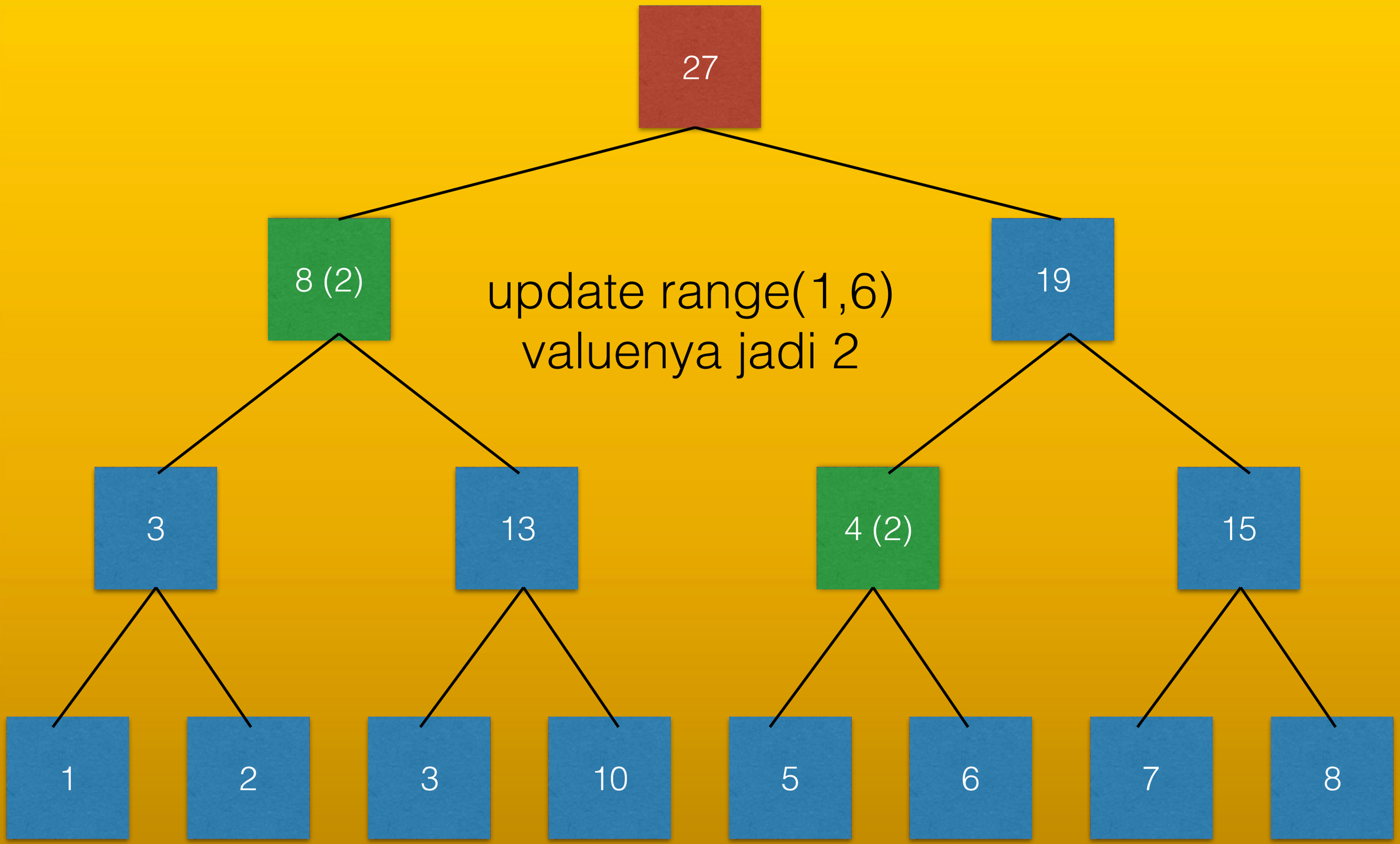
5

6

7

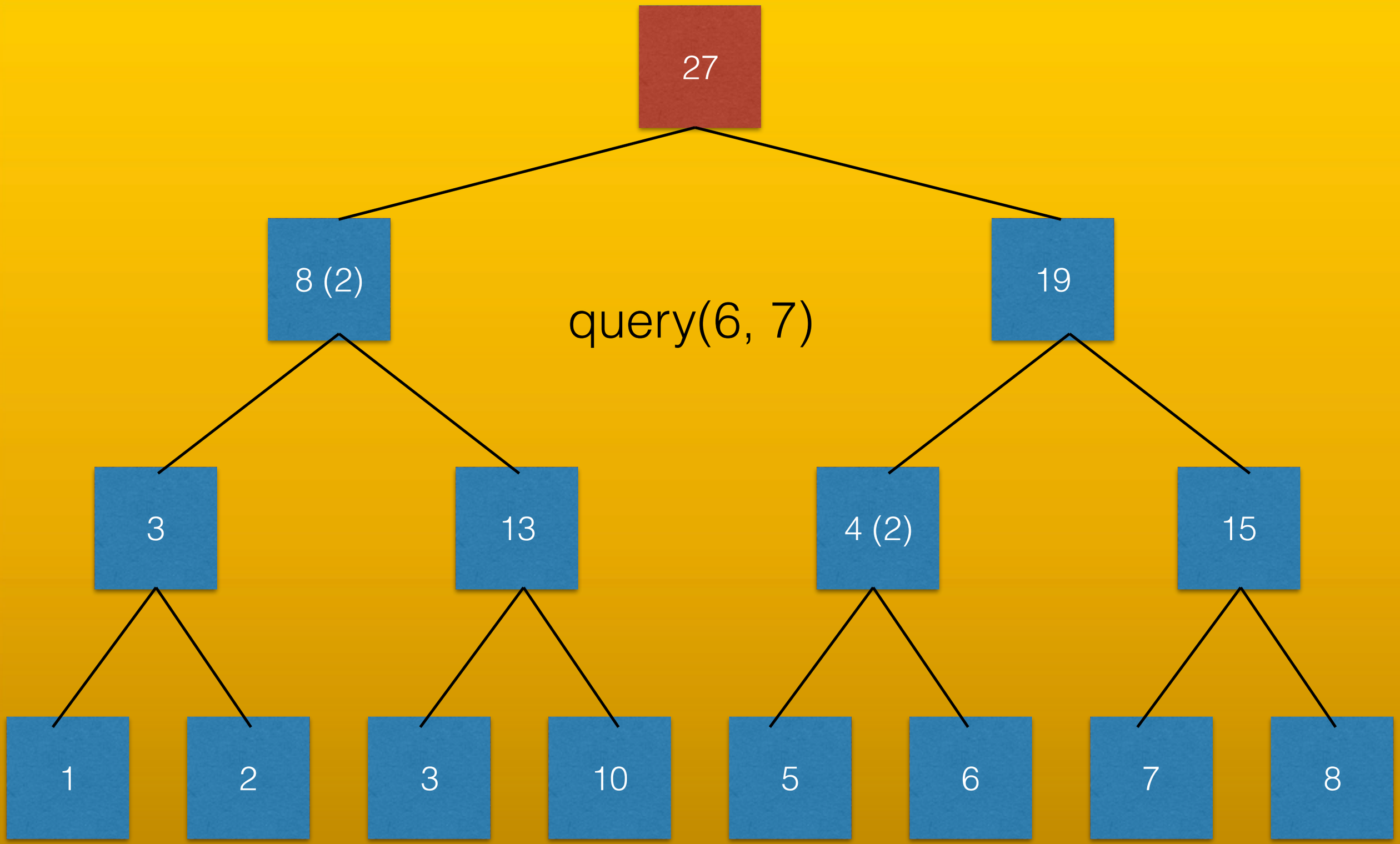
8

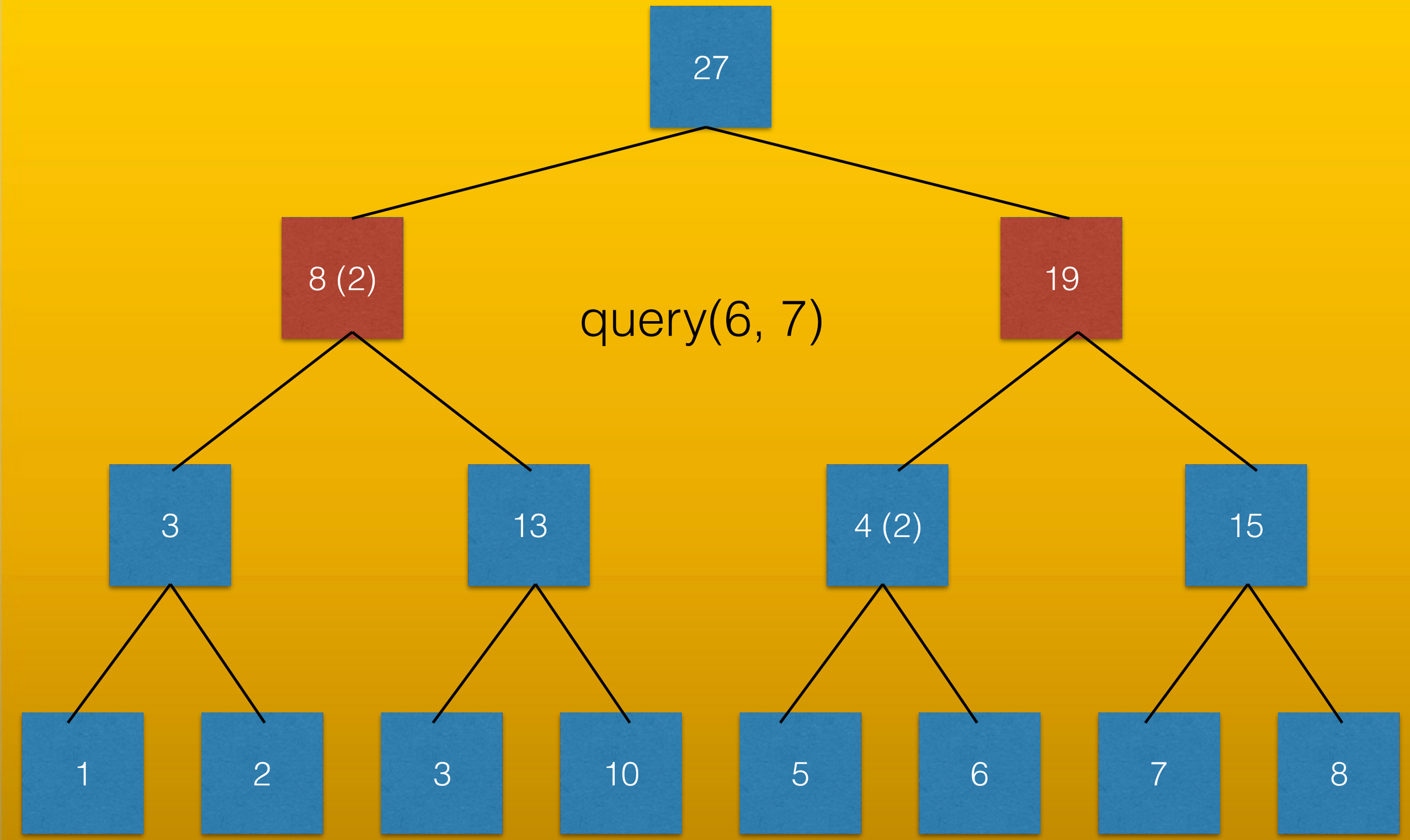

42
8 (2)
19
update range(1,6)
valuenya jadi 2
3
13
4 (2)
15
1
2
3
10
5
6
7
8

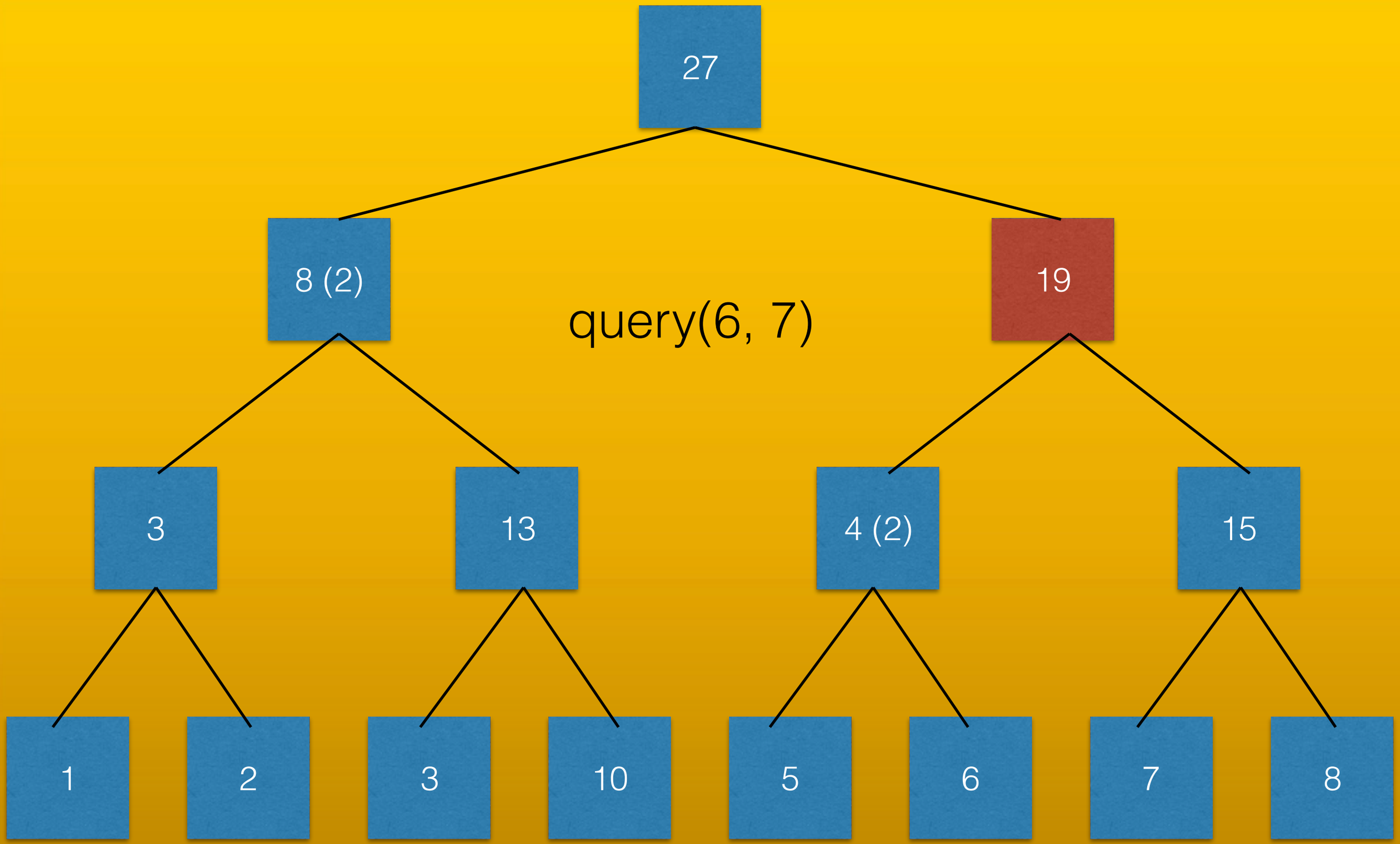

27
8 (2)
19
update range(1,6)
valuenya jadi 2
3
13
4 (2)
15
1
2
3
10
5
6
7
8

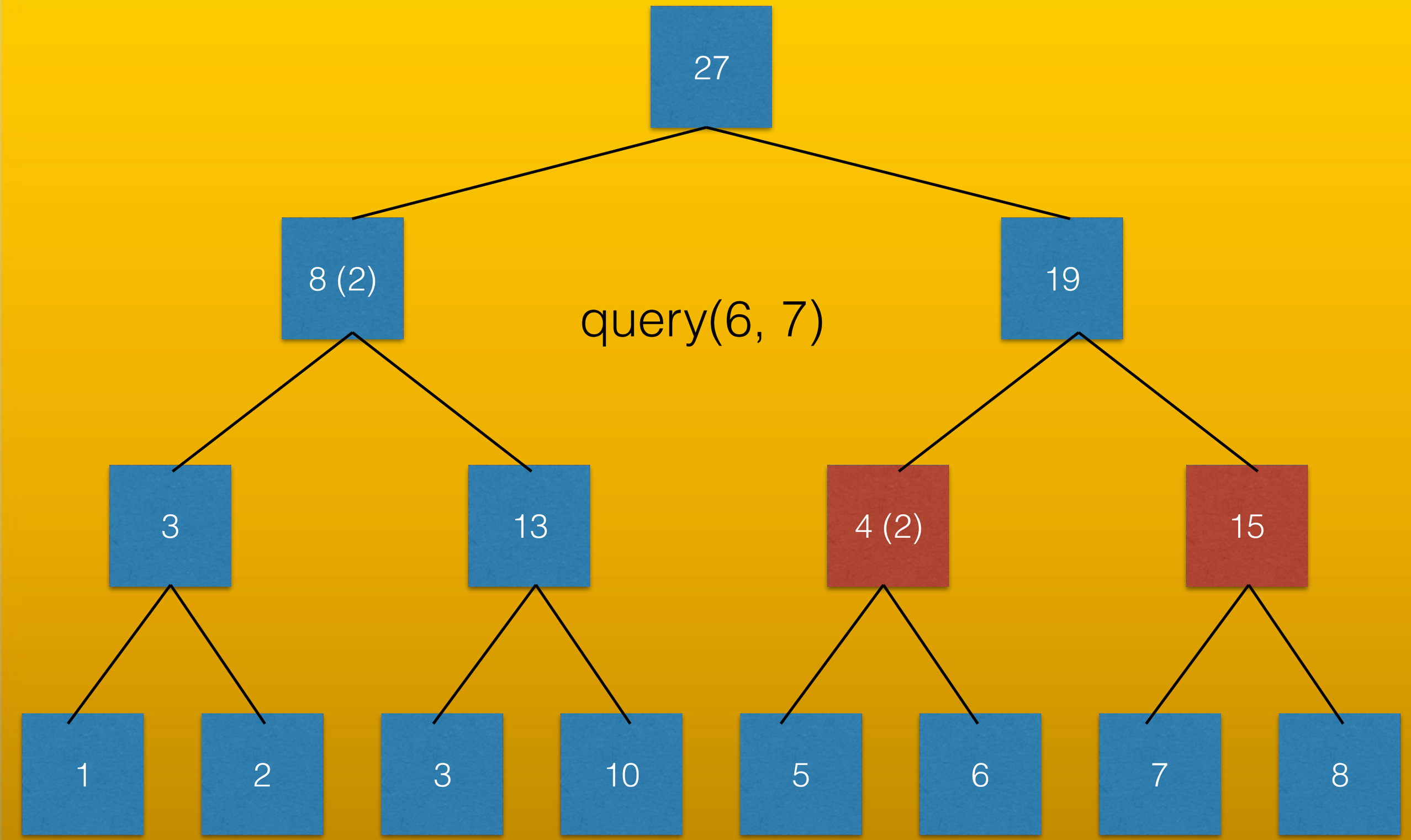
value yang disimpen ini bakal di “pushdown” kalo lu mau melewati node ini

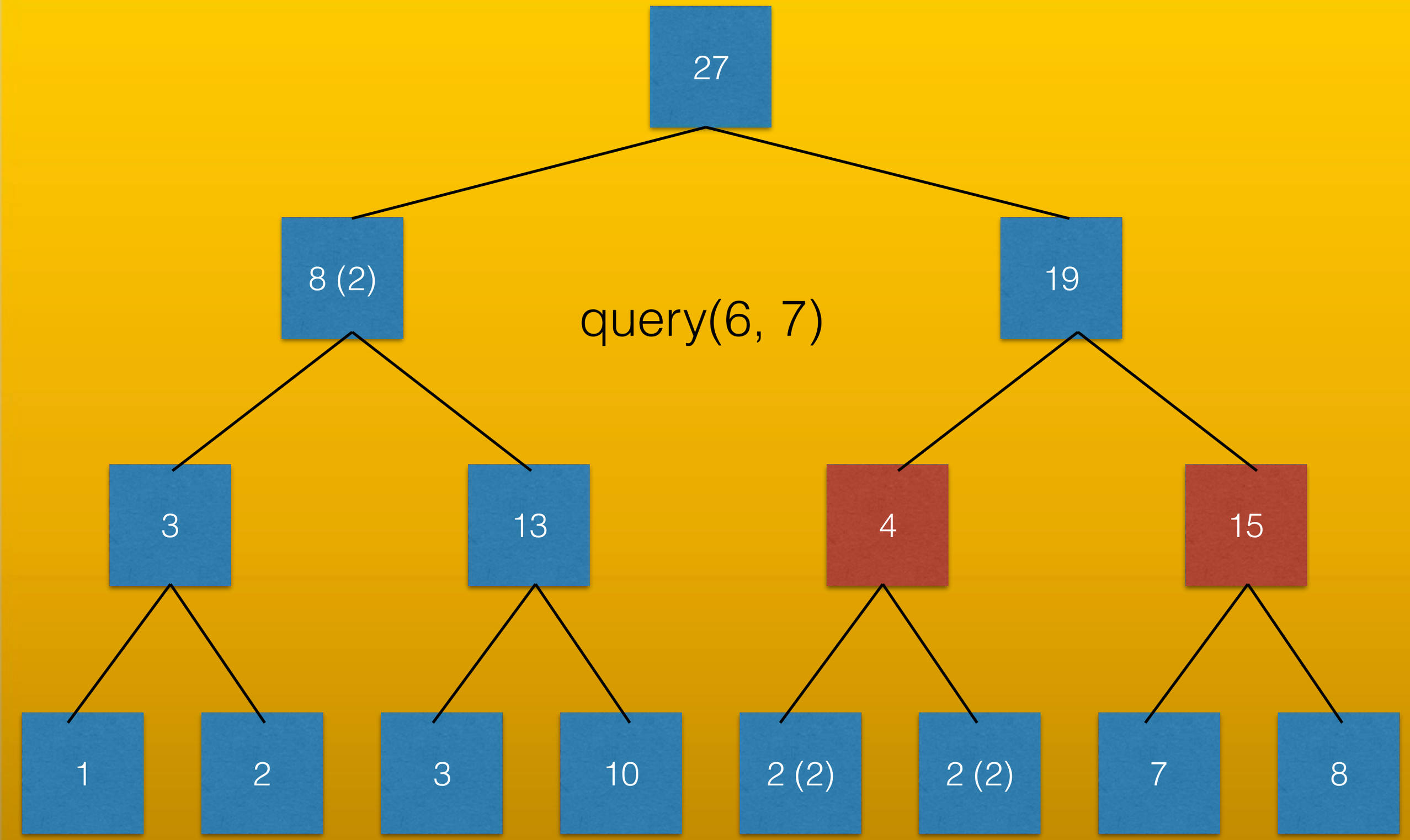
pasti ga ngerti

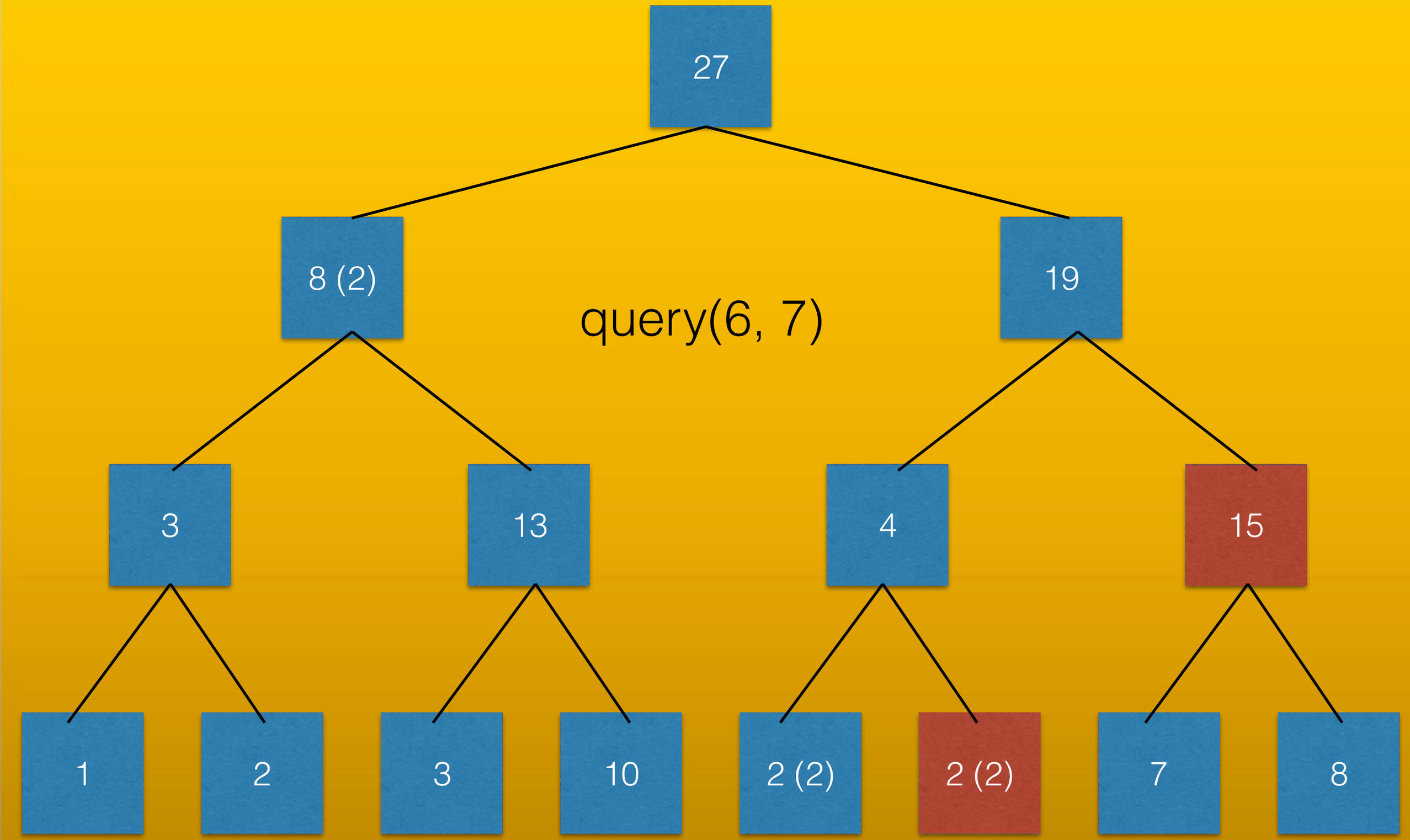

27
8 (2)
19
query(6, 7)
3
13
4 (2)
15
1
2
3
10
5
6
7
8

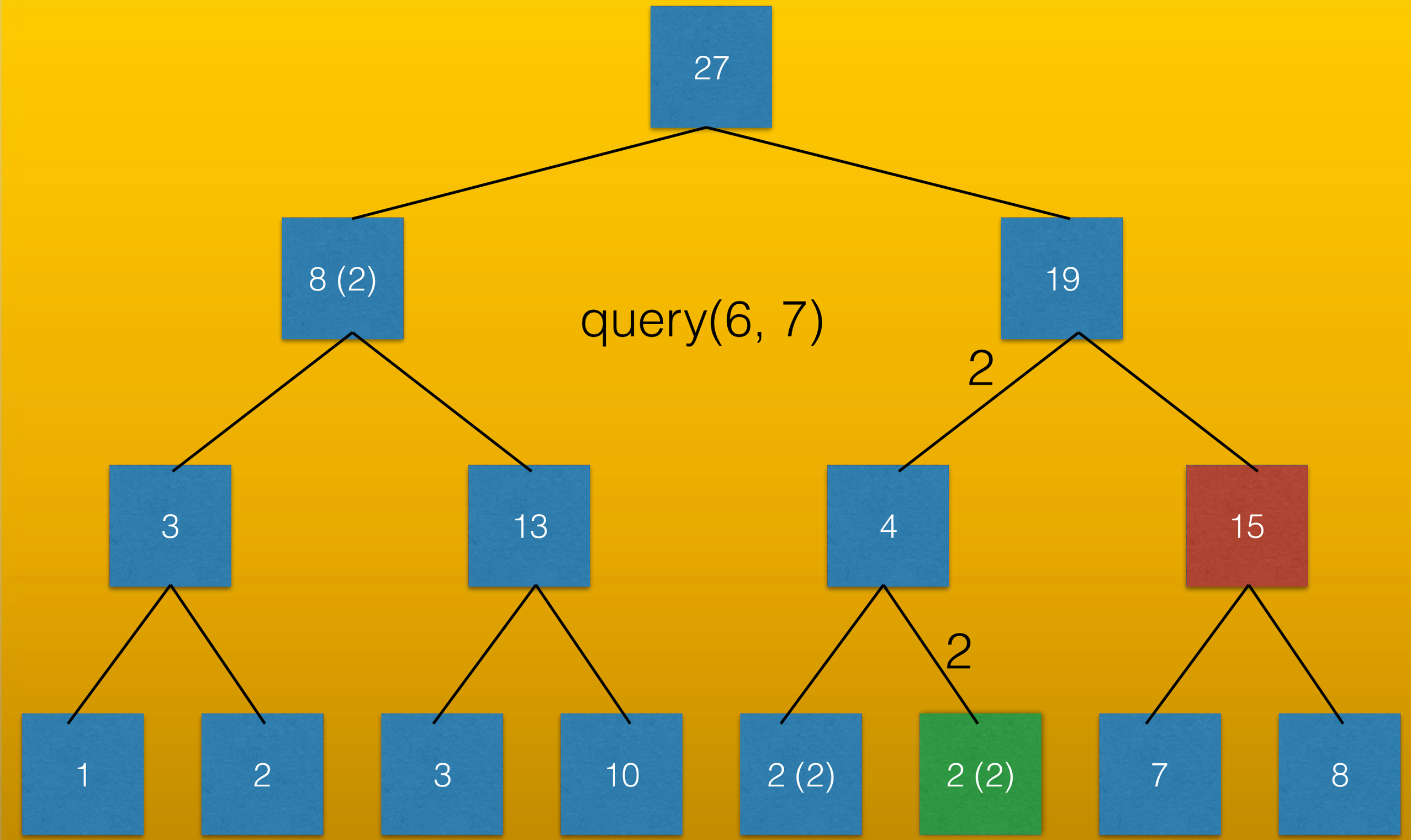

27
8 (2)
19
query(6, 7)
3
13
4 (2)
15
1
2
3
10
5
6
7
8

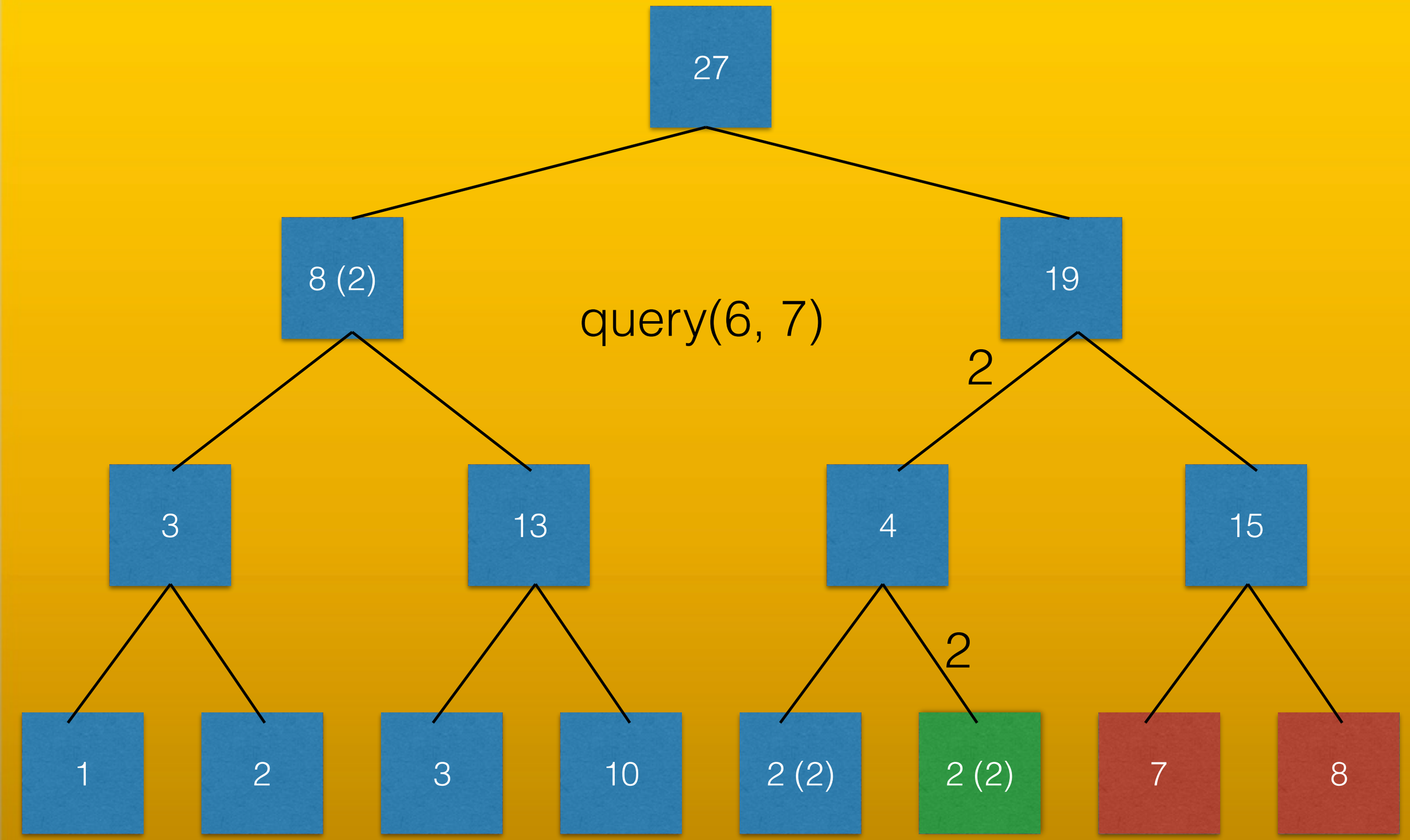

27
8 (2)
19
query(6, 7)
3
13
4 (2)
15
1
2
3
10
5
6
7
8

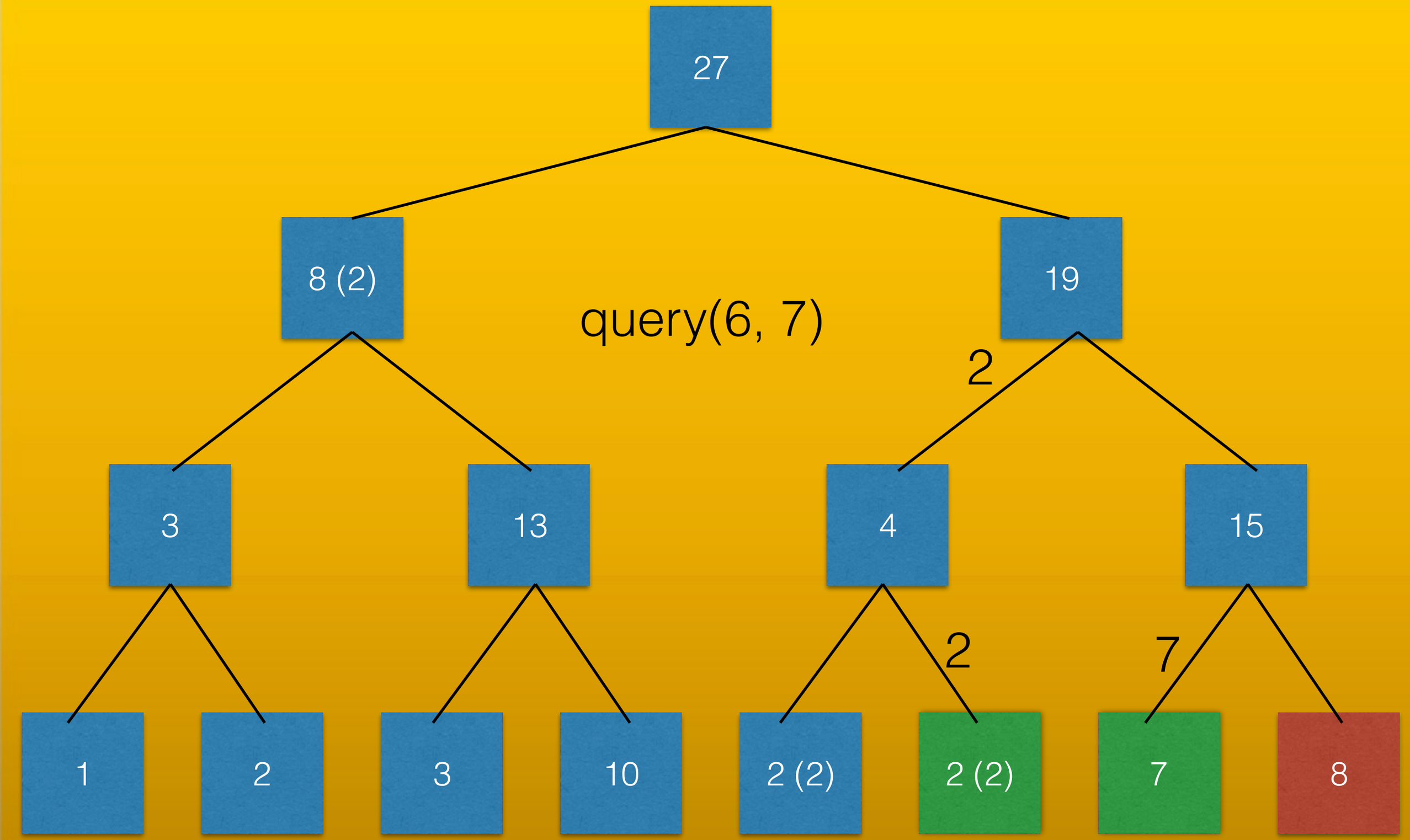

27
8 (2)
19
query(6, 7)
3
13
4 (2)
15
1
2
3
10
5
6
7
8

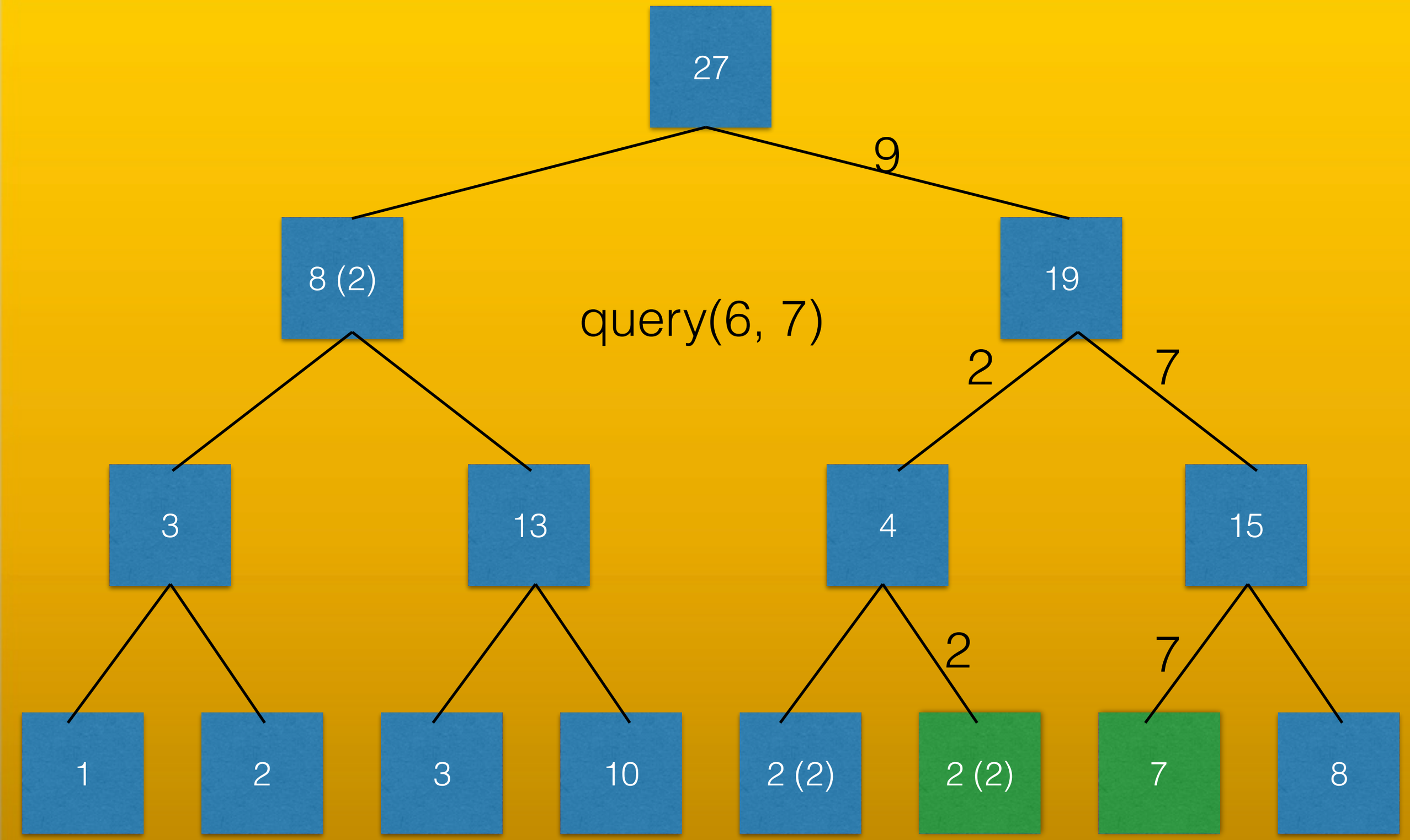

27
8 (2)
19
query(6, 7)
3
13
4
15
1
2
3
10
2 (2)
2 (2)
7
8


27
8 (2)
19
query(6, 7)
3
13
4
15
1
2
3
10
2 (2)
2 (2)
7
8


27
8 (2)
19
query(6, 7)
2
3
13
4
15
2
1
2
3
10
2 (2)
2 (2)
7
8

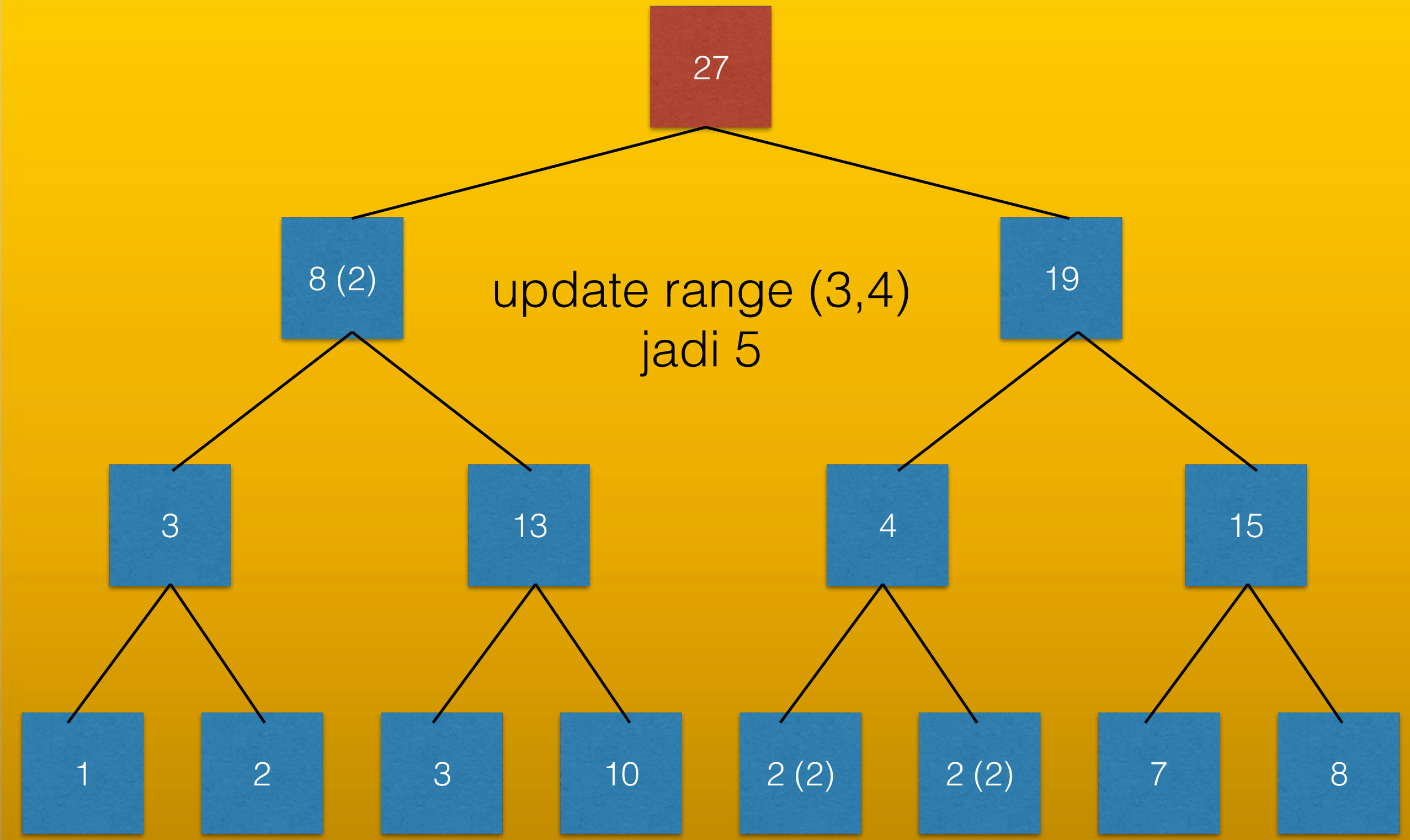

27
8 (2)
19
query(6, 7)
2
3
13
4
15
2
1
2
3
10
2 (2)
2 (2)
7
8


27
8 (2)
19
query(6, 7)
2
3
13
4
15
1
2
3
10
2 (2)
2 (2)
7
8
2
7


27
9
8 (2)
19
query(6, 7)
2
7
3
13
4
15
2
7
1
2
3
10
2 (2)
2 (2)
7
8

value yang disimpen bisa di”timpa” (atau dimerge) oleh value lain

pasti ga ngerti juga


27
8 (2)
19
update range (3,4)
jadi 5
3
13
4
15
1
2
3
10
2 (2)
2 (2)
7
8

27

8 (2)

19

update range (3,4)
jadi 5

3

13

4

15

1

2

3

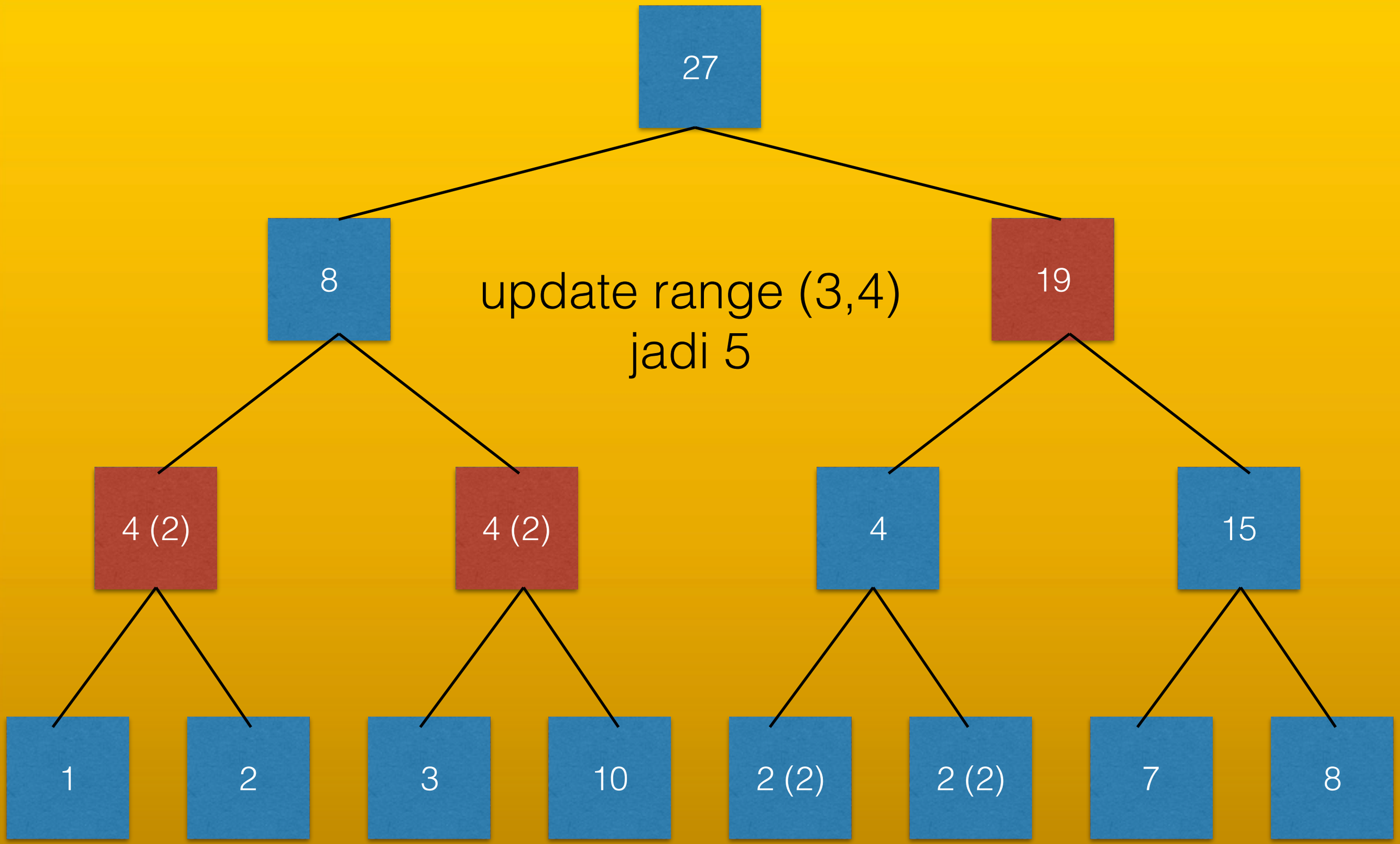
10

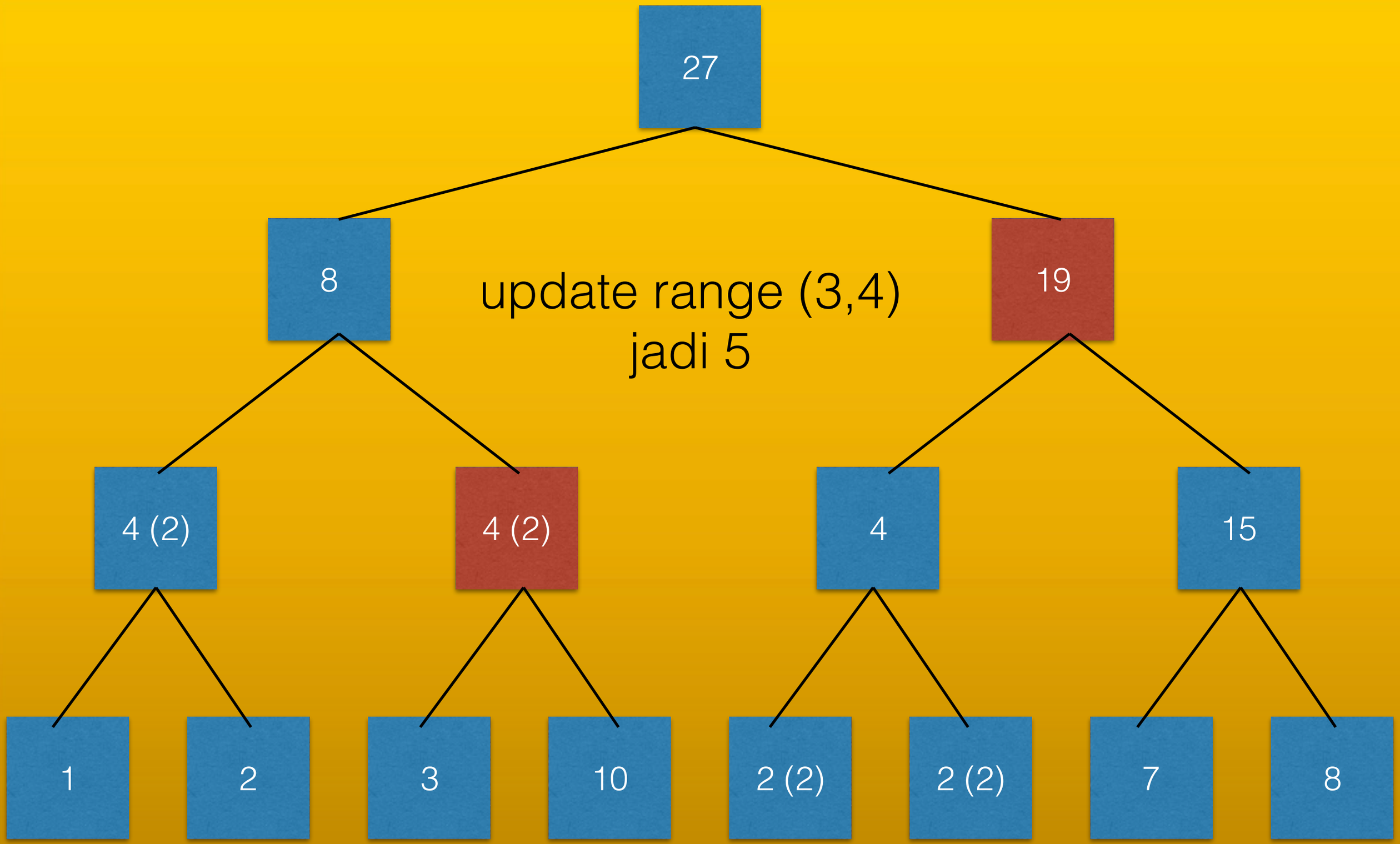
2 (2)

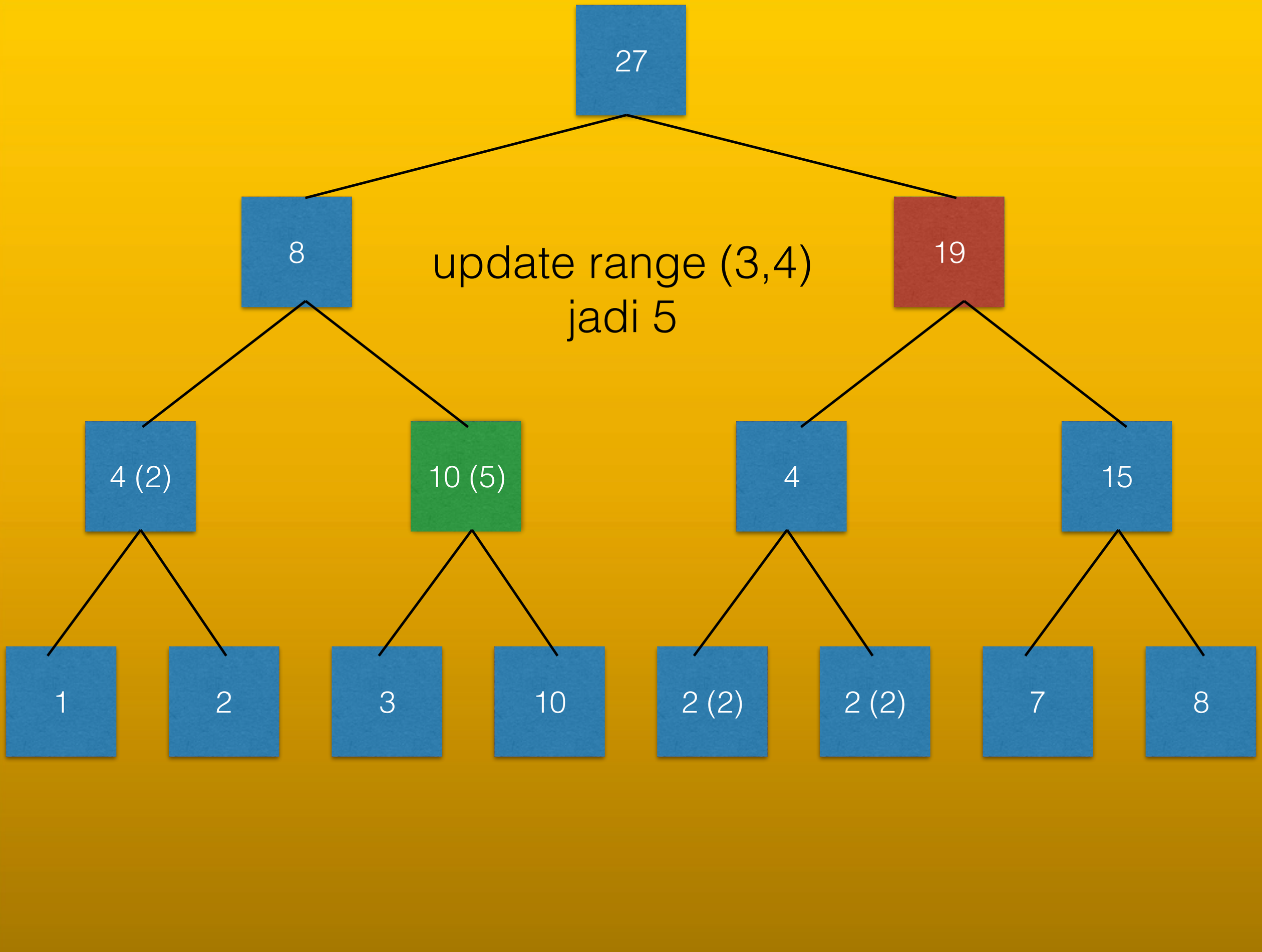
2 (2)

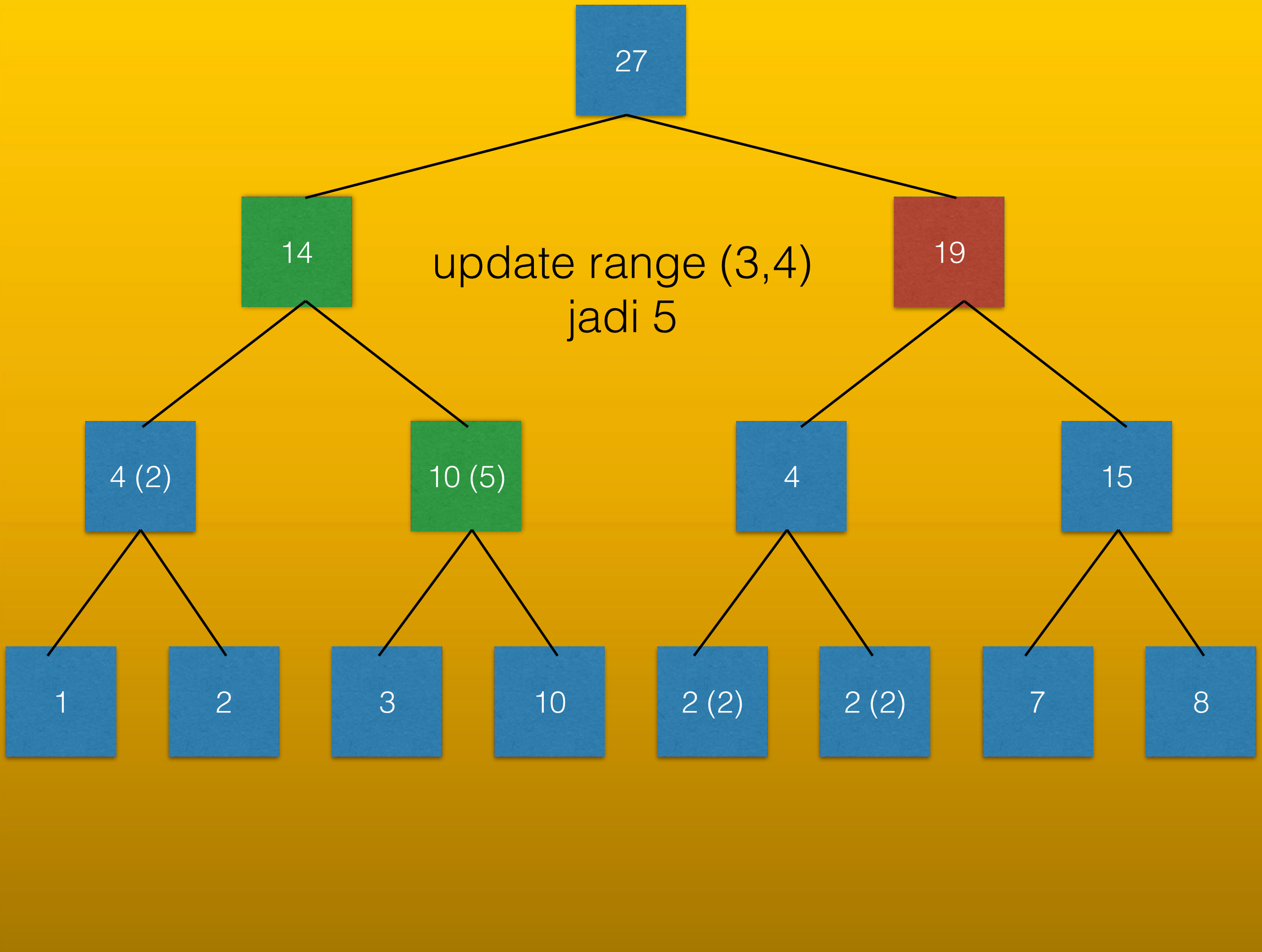
7

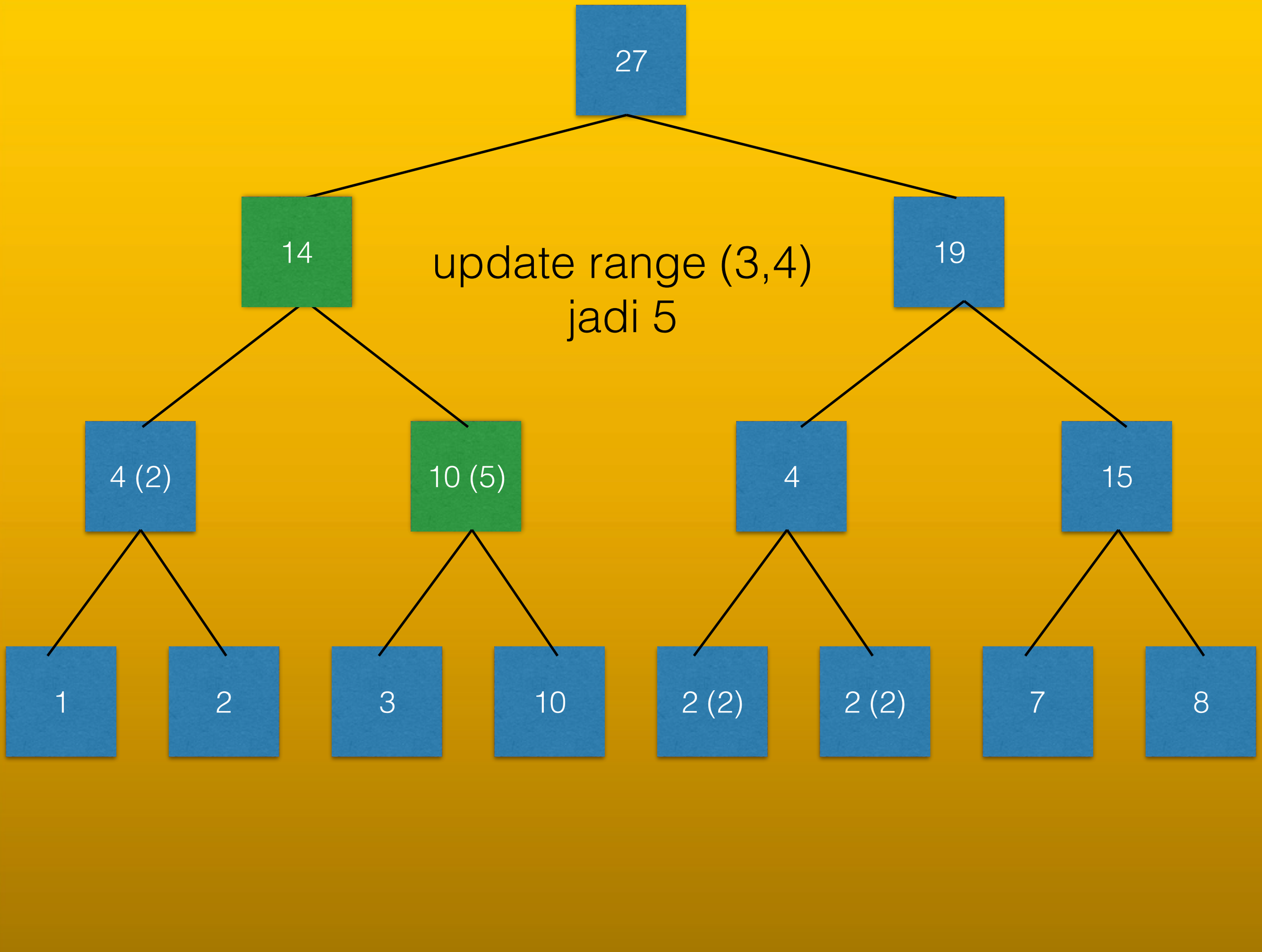
8

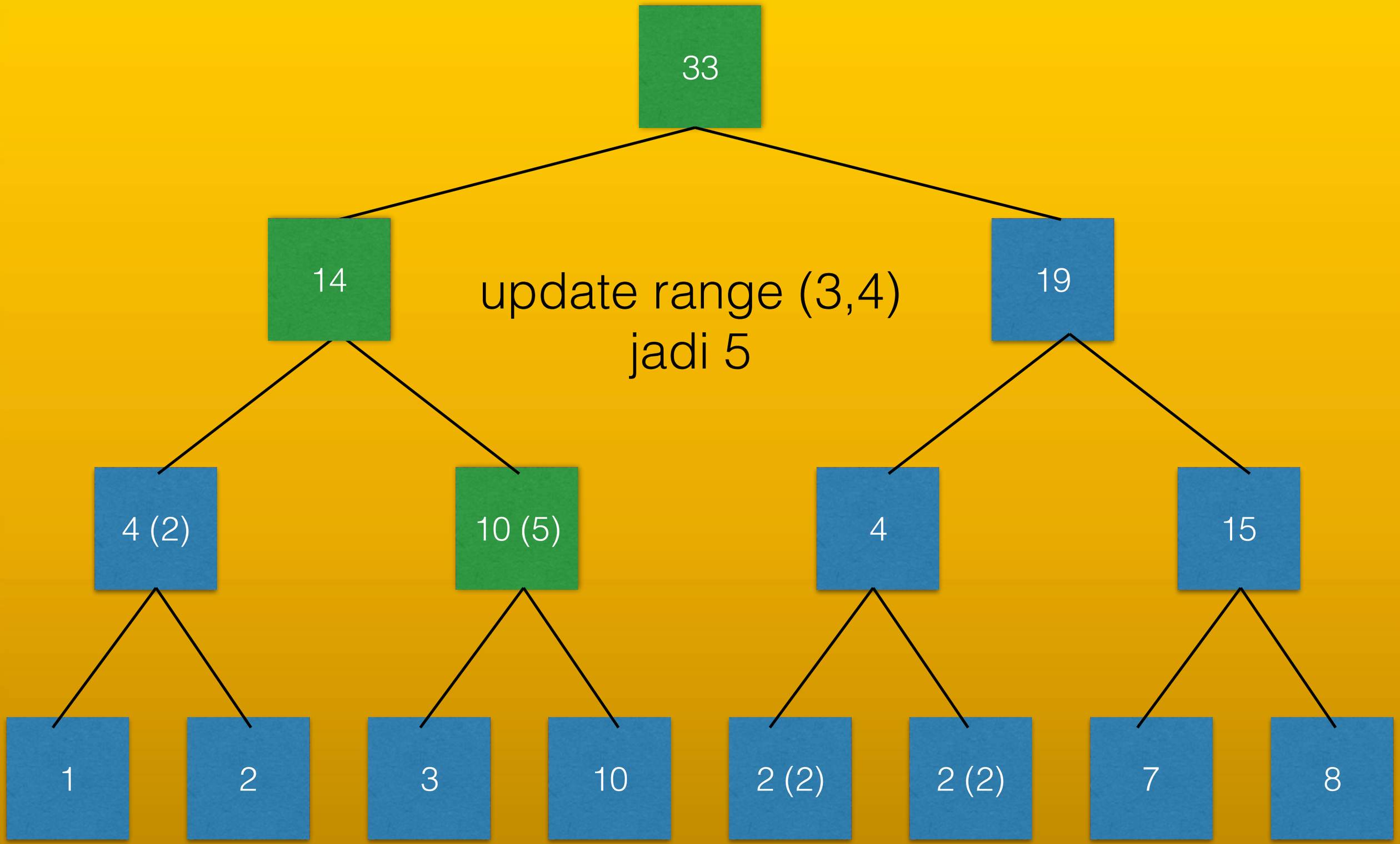

27
8
19
update range (3,4)
jadi 5
4 (2)
4 (2)
4
15
1
2
3
10
2 (2)
2 (2)
7
8


27
8
19
update range (3,4)
jadi 5
4 (2)
4 (2)
4
15
1
2
3
10
2 (2)
2 (2)
7
8


27
8
19
update range (3,4)
jadi 5
4 (2)
10 (5)
4
15
1
2
3
10
2 (2)
2 (2)
7
8


27
14
update range (3,4)
jadi 5
19
4 (2)
10 (5)
4
15
1
2
3
10
2 (2)
2 (2)
7
8


27
14
19
update range (3,4)
jadi 5
4 (2)
10 (5)
4
15
1
2
3
10
2 (2)
2 (2)
7
8


33
14
update range (3,4)
jadi 5
19
4 (2)
10 (5)
4
15
1
2
3
10
2 (2)
2 (2)
7
8

EOF

Q&A?