Persistent Data Structure

Jonathan Irvin Gunawan

E

E E

E E E E

E E E E E E E E

E E E E E E E E E E E E E E E E

prerequisite

C++ pointer

linked list

segment tree

In computing, a **persistent data structure** is a data structure that always preserves the previous version of itself when it is modified.

let's use this task to illustrate

given a linked list, there are two types of operation:

1. update the first x values
2. print the linked list after the k-th update

bruteforce :
create a new linked list every update
store it to an array for each “version”

optimisation: if value of x is small, can use the previous linked list allocate only x new spaces, then the x-th element points to the (x+1)-th element of the previous linked list

i can see the confusion

example

initial = {1, 2, 3, 4, 5, 6}

update 1 : x = 3, {12, 13, 14}

update 2 : x = 2, {20, 21}

update 3 : x = 5, {1, 2, 3, 4, 5}

update 4 : x = 1, {100}

example

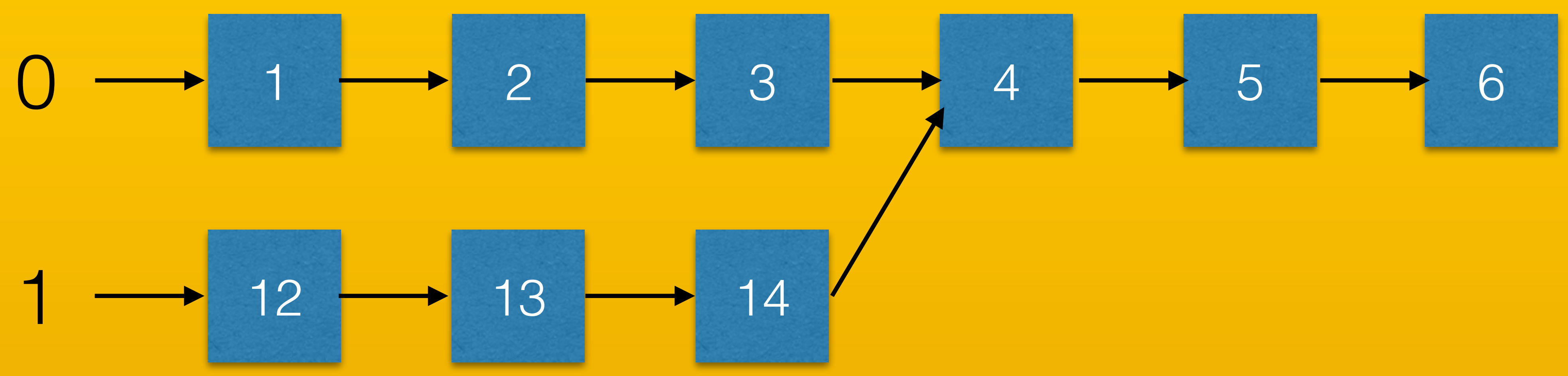
initial = {1, 2, 3, 4, 5, 6}

update 1 : x = 3, {12, 13, 14}

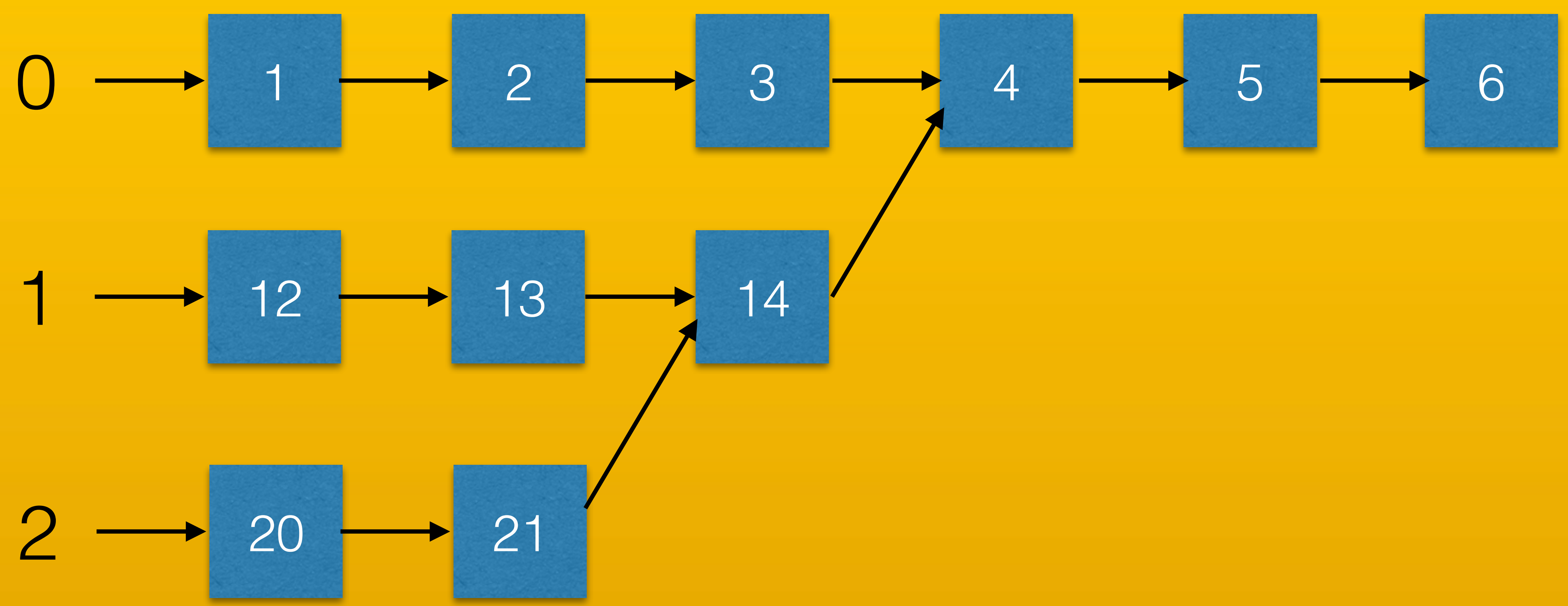
update 2 : x = 2, {20, 21}

update 3 : x = 5, {1, 2, 3, 4, 5}

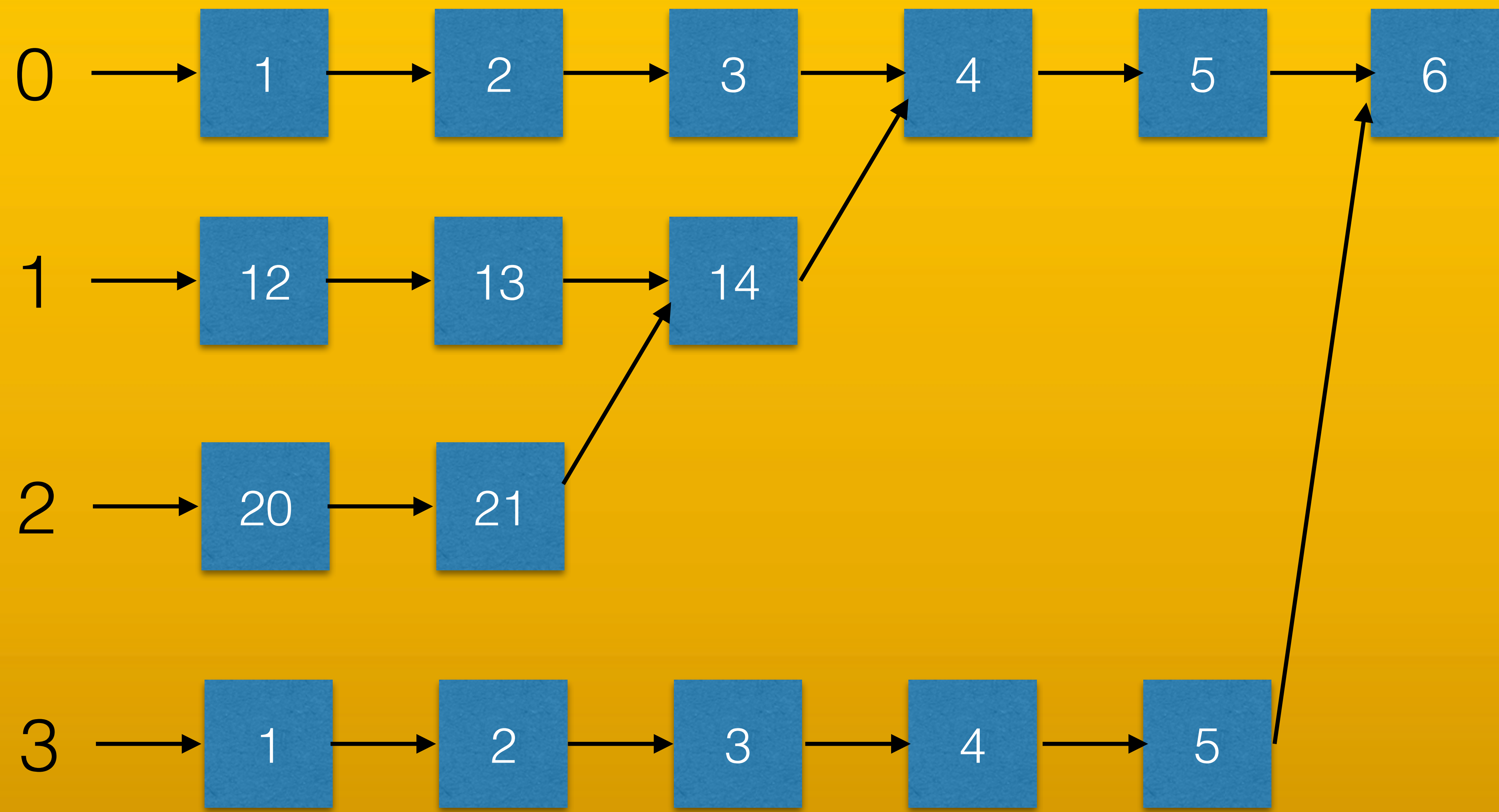
update 4 : x = 1, {100}



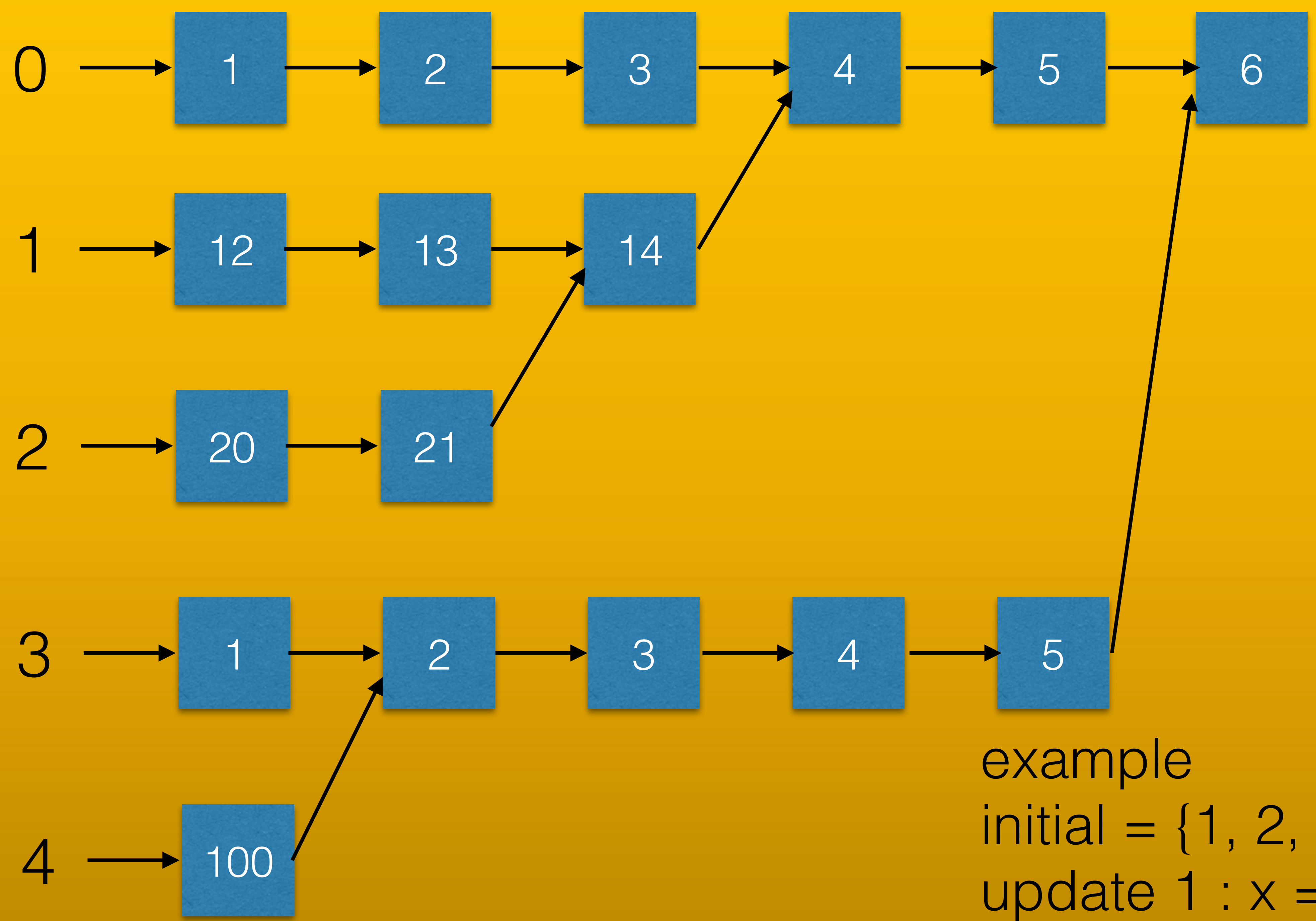
example
initial = {1, 2, 3, 4, 5, 6}
update 1 : x = 3, {12, 13, 14}
update 2 : x = 2, {20, 21}
update 3 : x = 5, {1, 2, 3, 4, 5}
update 4 : x = 1, {100}



example
initial = {1, 2, 3, 4, 5, 6}
update 1 : x = 3, {12, 13, 14}
update 2 : x = 2, {20, 21}
update 3 : x = 5, {1, 2, 3, 4, 5}
update 4 : x = 1, {100}



example
initial = {1, 2, 3, 4, 5, 6}
update 1 : x = 3, {12, 13, 14}
update 2 : x = 2, {20, 21}
update 3 : x = 5, {1, 2, 3, 4, 5}
update 4 : x = 1, {100}



example

initial = {1, 2, 3, 4, 5, 6}

update 1 : x = 3, {12, 13, 14}

update 2 : x = 2, {20, 21}

update 3 : x = 5, {1, 2, 3, 4, 5}

update 4 : x = 1, {100}

create the class first

```
class Node {
  int val;
  Node* next;

  Node(int val, Node* next): val(val), next(next) {}
}
```

```cpp
Node* last[N + 1];
Node* head[Q];

void init(vector<int> initial) {
  last[N - 1] = new Node(initial[N - 1], NULL);
  for (int i = N - 2; i >= 0; --i) {
    last[i] = new Node(initial[i], last[i + 1]);
  }
}

Node* insert(int index, const vector<int>& x) {
  if (index >= x.size()) {
    return last[index];
  }
  Node* now = new Node(x[index], insert(index + 1, x));
  return now;
}
```

if there is a vector x update (let's say k-th update), then just do

```cpp
head[k] = insert(0, x);
```

```
Node* last[N + 1];
Node* head[Q];

void init(vector<int> initial) {
  last[N - 1] = new Node(initial[N - 1], NULL);
  for (int i = N - 2; i >= 0; --i) {
    last[i] = new Node(initial[i], last[i + 1]);
  }
}

Node* insert(int index, const vector<int>& x) {
  if (index >= x.size()) {
    return last[index];
  }
  Node* now = new Node(x[index], insert(index + 1, x));
  return last[index] = now; // do not forget to update the last
}
```

if there is a vector x update (let's say k-th update), then just do

```
head[k] = insert(0, x);
```

```
Node* last[N + 1];
Node* head[Q];

void print(Node* now) {
  if (now == NULL) {
    return;
  }
  printf("%d ", now->val);
  print(now->next);
}
```

if there is a query to print the k-th linked list, just do

```
print(head[k]);
```

ok now
persistent segment tree

give motivation first

given an array A of N integers and Q queries

each query has three integers x,y,z. you must answer how many i satisfies $x \le i \le y$, $A[i] \le z$

$0 \le A[i] \le N$ so you won’t need any compression range trees?

O(lg^2 N) per query?

don’t want, I want O(lg N) per query

assume we have inifinite time and memory for precomputation before query we can create N^2 segment tree for each interval (i,j) we can create N^2 segment tree for each interval (i,j)

each segment tree stores the occurence of each number ONLY in that interval

```
int query(int ix, int L, int R, int z) {
  if (R == z) return tree[ix];
  int M = (L + R) >> 1;
  if (z <= M) return query(ix*2+1, L, M, z);
  else return tree[ix*2+1] + query(ix*2+2, M+1, R, z);
}
```

but the preprocessing becomes $O(N^3)$

so expensive

optimisation 1 :

instead of N^2 segment tree, we can optimise to only N segment tree k-th node of segment tree (i,j)

=

k-th node of segment tree (1,j)

-

k-th node of segment tree (1,i-1)

we still have N segment tree, which means O(N^2) preprocessing optimisation 2 :

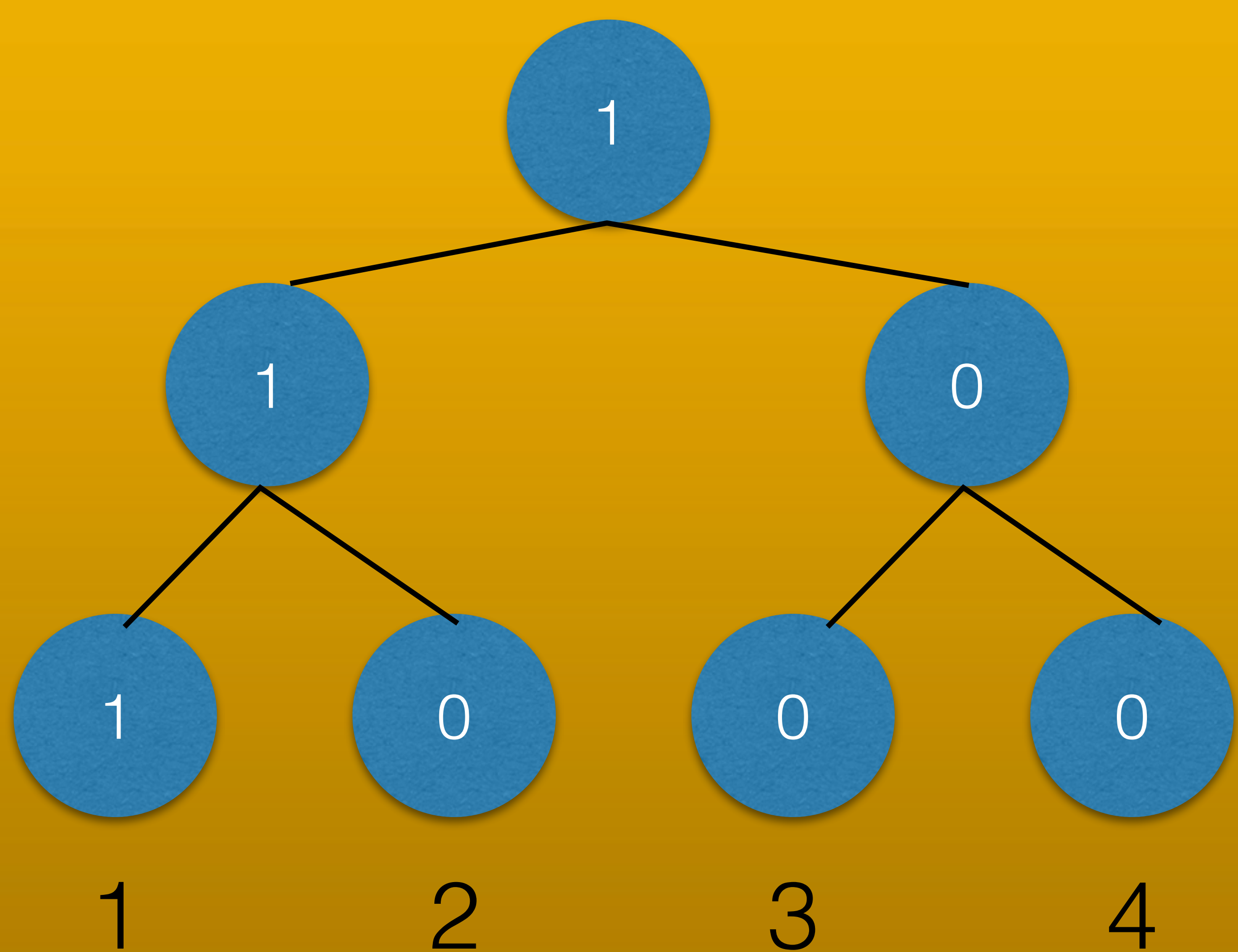
from segment tree (1,i) to segment tree (1,i+1), only log(N) node changes example

$$A = \{1,2,3,1\}$$

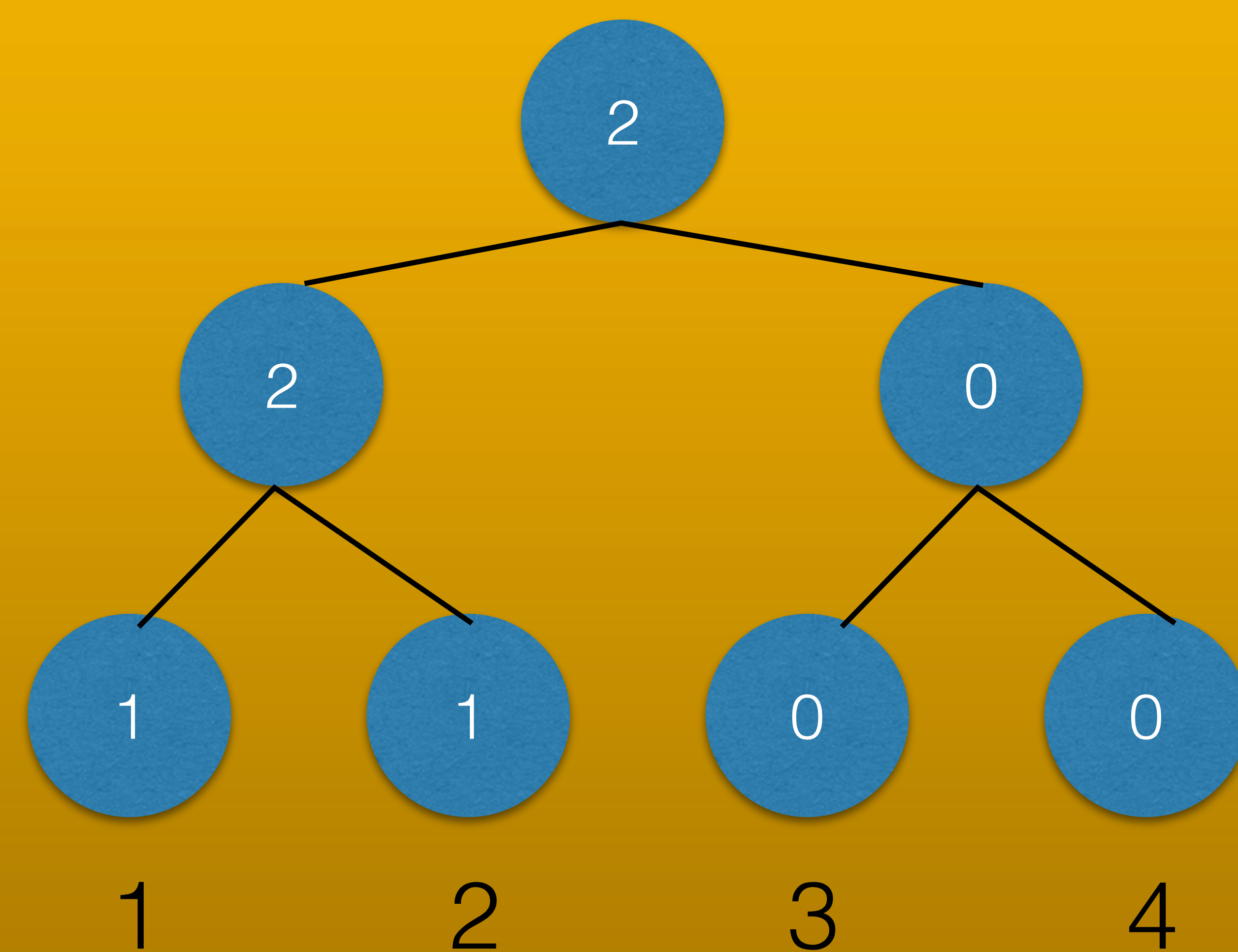
example

$$A = \{1,2,3,1\}$$

segtree (1,1)



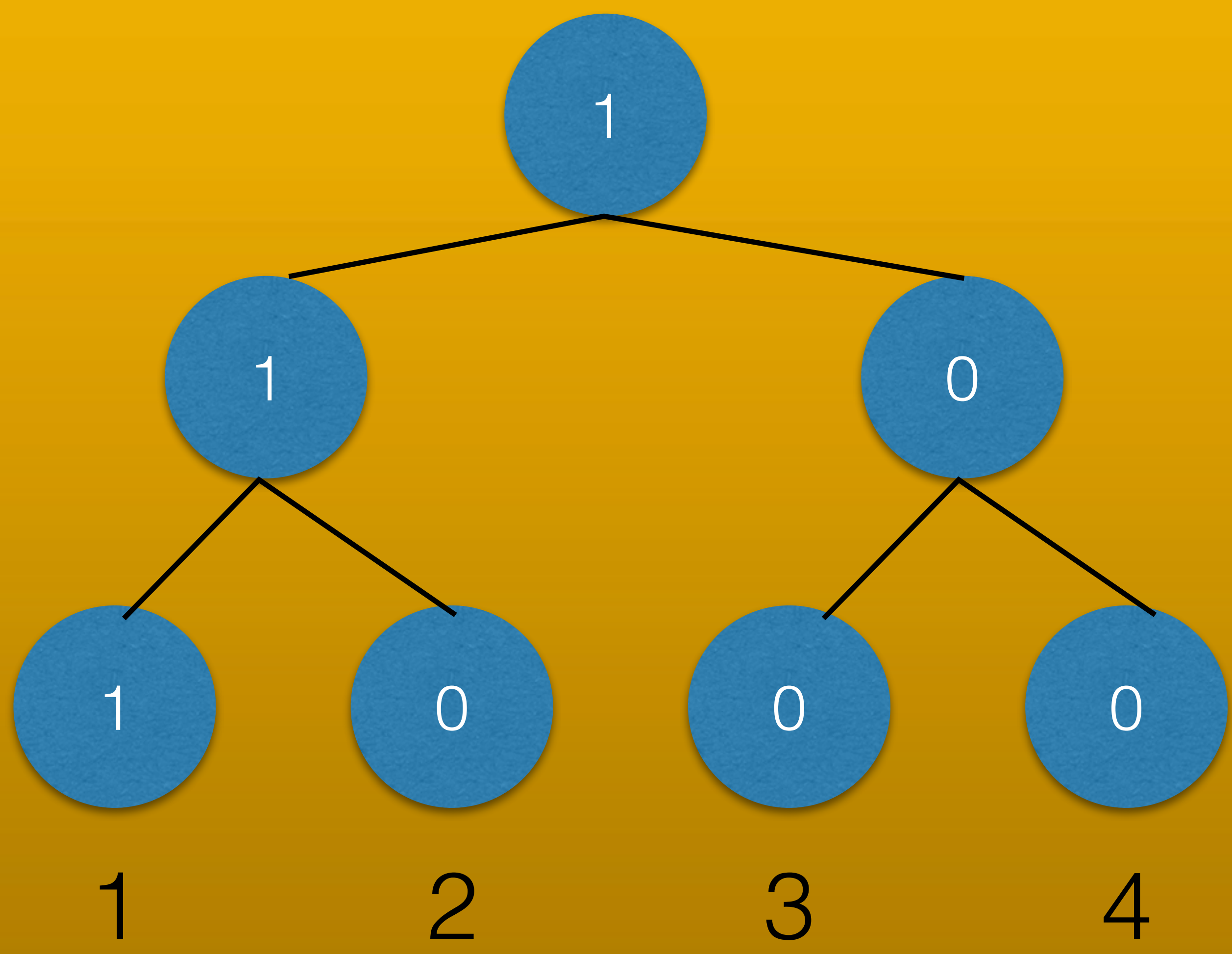
segtree (1,2)



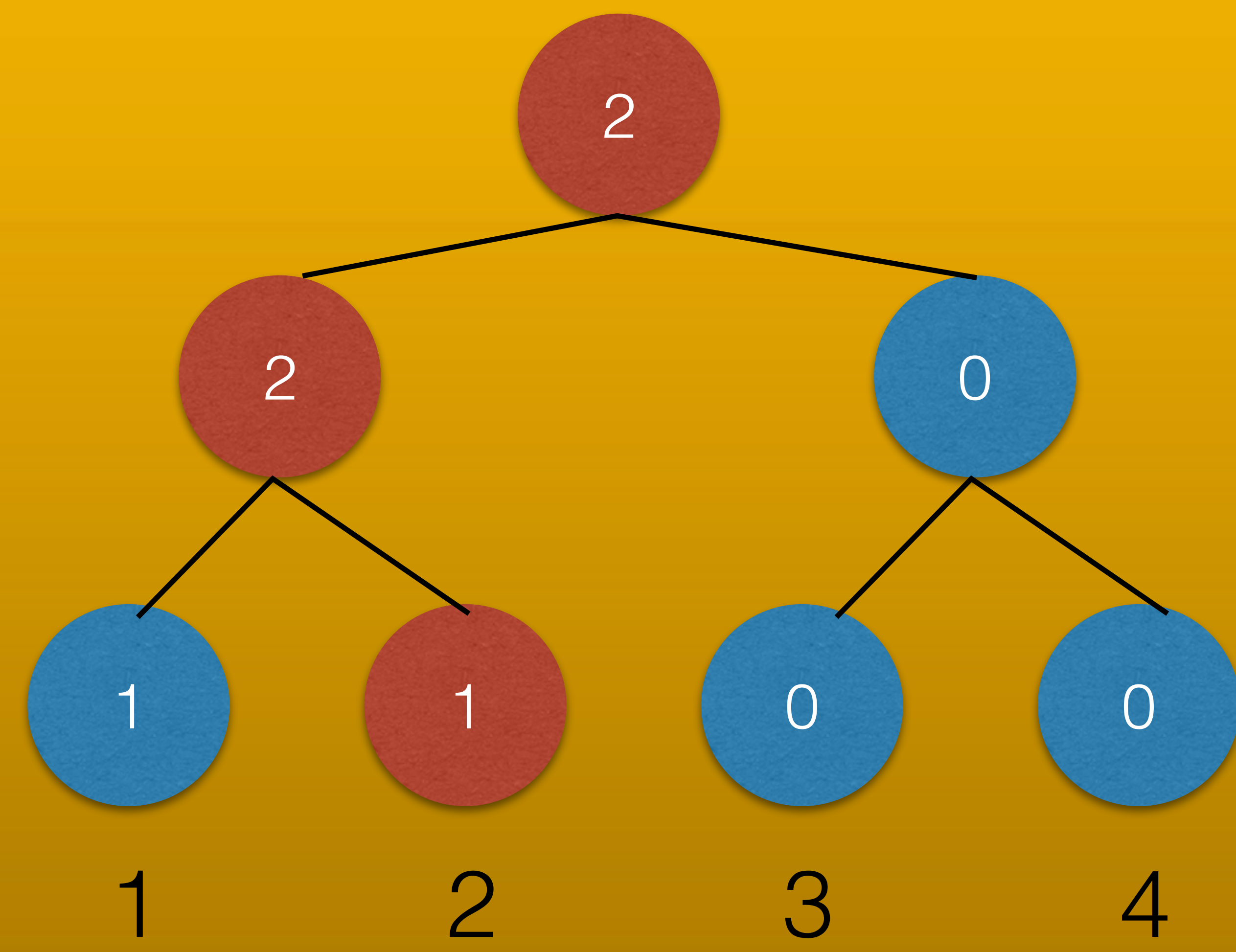
example

$A = \{1,2,3,1\}$

segtree (1,1)



segtree (1,2)



when computing segtree (1,2), if the range of a node does not cover 2, use the node from segtree (1, 1)

create the class first

```
class Node {
  int val;
  Node* left;
  Node* right;

  Node(int val, Node* left, Node* right):
    val(val), left(left), right(right) {}
}
```

```
Node* last[4 * N];
Node* root[N];

Node* insert(int ix, int L, int R, int z) {
  if (z < L || R < z) return last[ix];
  Node* now;
  if (L == R) {
    now = new Node(last[ix]->val + 1, NULL, NULL);
  } else {
    int M = (L + R) >> 1;
    now = new Node(last[ix]->val + 1,
        insert(ix*2+1, L, M, z), insert(ix*2+2, M+1, R, z));
  }
  return last[ix] = now;
}
```

call `root[k] = insert(0,0,N-1,A[k]);`

the query

```
Node* last[4 * N];
Node* root[N];

int query(Node* u, Node* v, int ix, int L, int R, int z) {
  if (L == R) return v->val - u->val;
  int M = (L + R) >> 1;
  if (z > M)
    return v->left->val - u->left->val +
         query(u->right, v->right, ix*2+2, M+1, R, z);
  return query(u->left, v->left, ix*2+1, L, M, z);
}


int x, y, z; // find how many integers < z in A[x..y]
int ans = query(root[x-1], root[y], 0, 0, N-1, z);
```

let’s practice

SPOJ MKTHNUM

I am pretty sure some of you have read it

given array N and Q queries.
each query has three integers x,y,k.
Find the k-th integer in {A[x], A[x+1], A[x+2], …, A[y]} if sorted

there is a solution using range trees + binary search, $O(N \lg^3 N)$

but now try to find the $O(N \lg N)$ solution

new slides
added after class

tasks that we discussed in the class but not in the slides

Codeforces Round #406 (Div 1) problem C

Given N people in a line, each having a color. For each $1 \leq k \leq N$, we want to partition the people so that each group is a contiguous interval and has at most k distinct colours. Determine the minimum number of groups

$$1 \leq N \leq 1e5$$

https://tlx.toki.id/problems/ngoding-seru-2019-final/C

(no English translation, use google translate to translate the I/O format :))

Given a weighted tree of N nodes and Q queries U, L, R
For each query, find a leaf in the subtree of U that has an index between L to R and is nearest to U

$1 \leq N,Q \leq 1e5$

https://tlx.toki.id/problems/toc-2017-oct/1C

(no English translation, use google translate to translate the I/O format :))

Given an array A, B of N integers and 6 types of operations

1, 2 : Set A[x] (or B[x]) = v

3: Set A[L..R] = B[L..R]

4: Set B[L..R] = A[L..R]

5: Calculate (A[L]*A[L+1]*…A[R]) % 1e9

6: Calculate (B[L]*B[L+1]*…B[R]) % 1e9

$1 \le N, \#op \le 1e5$

EOF

Q&A?