Segment Trees

CS 491 – Competitive Programming

Dr. Mattox Beckman

University of Illinois at Urbana-Champaign
Department of Computer Science

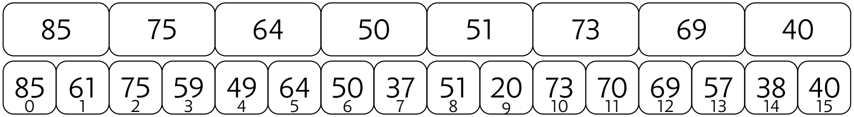
Fall 2023

Running Example

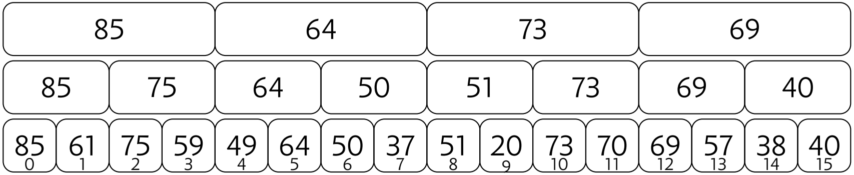
- Consider the following array:

85	61	75	59	49	64	50	37	51	20	73	70	69	57	38	40
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

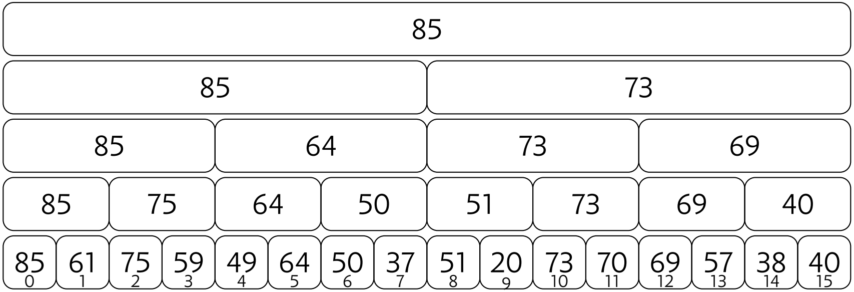
Segment Trees, Level 1



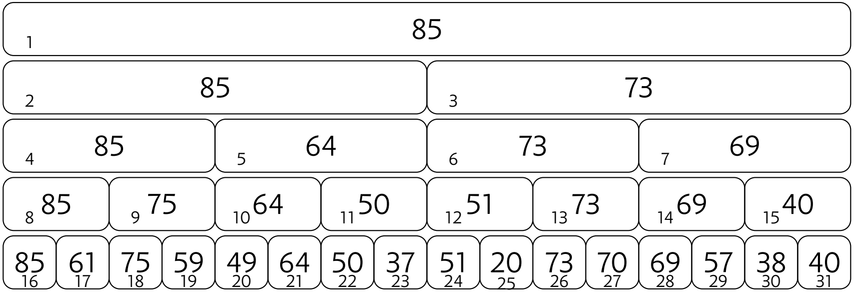
Segment Trees, Level 2



Segment Trees, Level 3 and 4



Segment Trees, Numbering the Elements



Build the Segement Tree

- L and R give you the bounds with respect to the orignal array.
- This code gives you a min range.

```
void build(int p, int L, int R) {
  if (L == R) // as L == R, either one is fine
     st[p] = data[L]; // store the data
  else {
    // recursively compute the values
    build(left(p) , L               , (L + R) / 2);
    build(right(p), (L + R) / 2 + 1, R          );
    int p1 = left(p), p2 = right(p);
    st[p] = min(st[p1],st[p2]);
  } }
```

Query the Tree

- L and R give you the bounds with respect to the orignal array.
- i and j give you the bounds for the query

```
int rmq(int p, int L, int R, int i, int j) {
  if (i > R || j < L) return -1; // current segment outsid
  if (L >= i && R <= j) return st[p];
  // compute the min position in the left and right part o
  int lm = rmq(left(p) , L         , (L+R)/2, i, j);
  int rm = rmq(right(p), (L+R)/2+1, R       , i, j);
  if (lm == -1) return rm;
  // if we try to access segment outside query
  if (rm == -1) return lm;
  return min(lm,rm);
}

```