

Sqrt Decomposition

CS 491 – Competitive Programming

Dr. Mattox Beckman

UNIVERSITY OF ILLINOIS AT URBANA-CHAMPAIGN
DEPARTMENT OF COMPUTER SCIENCE

Fall 2023

Objectives

- ▶ Use sqrt decomposition to improve the time complexity of large problems.

Running Example

- ▶ Consider the following array:

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

- ▶ What is an algorithm, given i and j , of returning the sum between these numbers (inclusive)?
- ▶ What is an algorithm, given i and j , of returning the max between these numbers (inclusive)?

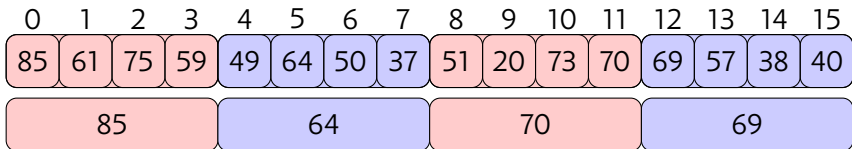
Code for Sum

```
1  vi run;
2  int a = 0;
3
4  run.push_back(0); // sentinel
5  for(int i: data) {
6      a += data;
7      run.push_back(a);
8  }
9
10 int sum(int i, int j) {
11     return run[j+1] - run[i];
12 }
```

- ▶ We can't do a "running max" though.

Solution

- ▶ We can create a separate array to handle each block of \sqrt{n} size.



- ▶ What is the max number between...
 - ▶ 0 and 2?
 - ▶ 4 and 11?
 - ▶ 3 and 8?
- ▶ What is the resulting time complexity?

Preprocessing Code

- ▶ `sq` contains the sqrt decompositions.
- ▶ `data` contains the raw data.

```
1 vi data, sq;
2 int n,a,d;
3
4 cin >> n;
5 int s = sqrt(n);
6 sq = vi(s+1);
7
8 for(int i=0,j=0; i < s && j < n; ++i ) {
9     cin >> d; data.push_back(d);
10    sq[i] = d;
11    for(k=1; k<s; ++j, ++k) {
12        cin >> d; data.push_back(d);
13        sq[i] = max(sq[i],d);
14    }
15 }
```

Search

```
1  int findMax(int i, int j, vi &data, vi &sq, int s) {
2      int a;
3      a = data[i];
4      while (i % s > 0) {
5          a = max(a,data[i]);
6          ++i;
7      }
8
9      while ( (j+1) % s > 0) {
10         a = max(a,data[j]);
11         --j;
12     }
13
14     for(k = i/s; k<=j/s; ++k)
15         a = max(a,sq[k]);
16 }
```