Introduction to Algorithms

Science Honors Program (SHP)

Session 3

Josh Alman & Christian Lim

Saturday, March 2, 2024

A little bit about me

Josh Alman

Professor in the Columbia Computer Science Department

Teach and Research about Theoretical Computer Science

- Algorithm Design
- Complexity Theory

Used to do programming contests :)

- Competed at ICPC world finals
- Coached Stanford ICPC team for a couple years
- Help (a tiny bit) with the Columbia ICPC team now

Slide deck in github

- You may get to the link by:
  - https://github.com/yongwhan/
  - => yongwhan.github.io
  - => columbia
  - => shp
  - => session 3 slide

Overview

- **Custom data structures (con't)** (Segment tree; ordered set)
- Break #1 (5-minute)
- **Complete Search**
- Break #2 (5-minute)
- **Divide and Conquer**
- *Break #3 (5-minute)
- ***Interactive Session**

- ***: only if there is time at the end!**

Attendance

- Let's take a quick attendance before we begin!

CodeForces Columbia SHP Algorithms Group

- While I take the attendance, please join the following group: **https://codeforces.com/group/lfDmo9iEr5**

- We will be using them in the last portion of the session today!

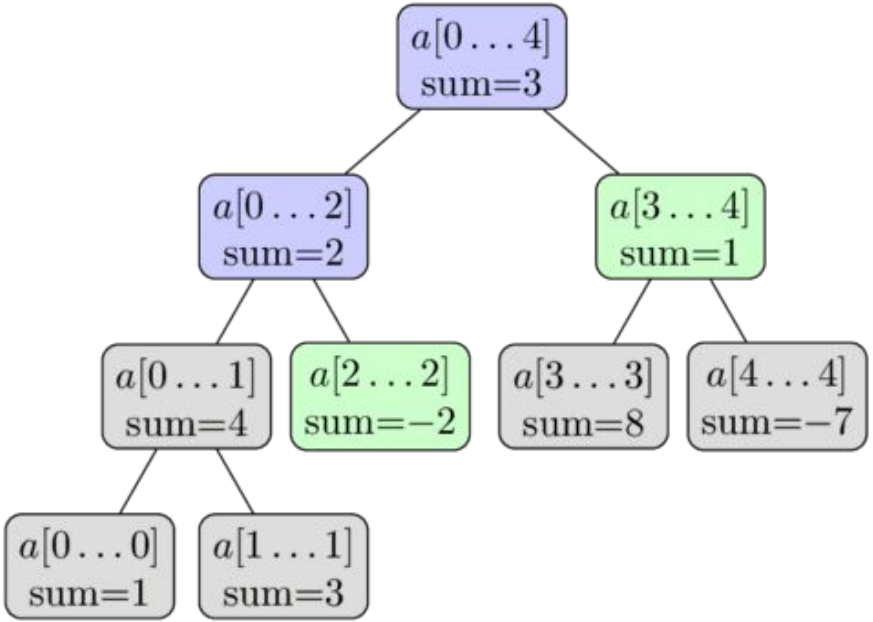
Custom Data Structures (con't)

- Segment tree
- ordered set

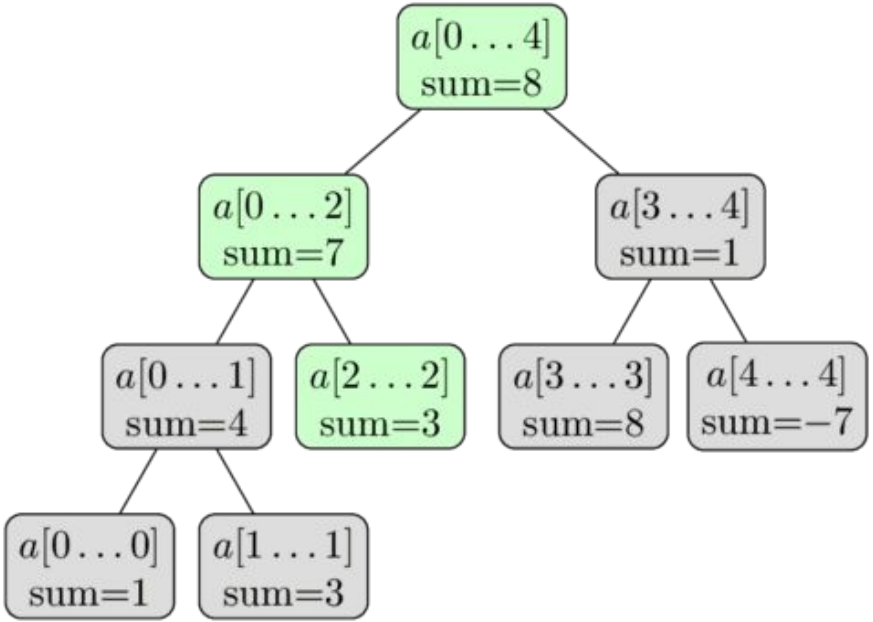
Segment Tree: Point Update & Range Query

- `update(i, x)`
- `sum(l, r)`

Segment Tree: Example: sum(2,4)



Segment Tree: Example: update(2,3)



Segment Tree: Implementation

```
int n, t[4*MAXN];
void build(int a[], int v, int tl, int tr) {
  if (tl == tr) {
    t[v] = a[tl];
  } else {
    int tm = (tl + tr) / 2;
    build(a, v*2, tl, tm);
    build(a, v*2+1, tm+1, tr);
    t[v] = t[v*2] + t[v*2+1];
  }
}
```

Segment Tree: Implementation

```
int sum(int v, int tl, int tr, int l, int r) {
  if (l > r)
    return 0;
  if (l == tl && r == tr) {
    return t[v];
  }
  int tm = (tl + tr) / 2;
  return sum(v*2, tl, tm, l, min(r, tm))
         + sum(v*2+1, tm+1, tr, max(l, tm+1), r);
}
```

Segment Tree: Implementation

```
void update(int v, int tl, int tr,
            int pos, int new_val) {
  if (tl == tr) t[v] = new_val;
  else {
    int tm = (tl + tr) / 2;
    if (pos <= tm) update(v*2, tl, tm, pos, new_val);
    else update(v*2+1, tm+1, tr, pos, new_val);
    t[v] = t[v*2] + t[v*2+1];
  }
}
```

Lazy Segment Tree: Range Update & Range Query

- `update(l, r, add)`
- `max(l, r)`

- **update, only when you need to!**

Lazy Segment Tree: Implementation

```
void push(int v) {
  t[v*2] += lazy[v];
  lazy[v*2] += lazy[v];
  t[v*2+1] += lazy[v];
  lazy[v*2+1] += lazy[v];
  lazy[v] = 0;
}
```

Lazy Segment Tree: Implementation

```
void update(int v, int tl, int tr,
            int l, int r, int add) {
  if (l > r) return;
  if (l == tl && tr == r) t[v] += add, lazy[v] += add;
  else {
    push(v);
    int tm = (tl + tr) / 2;
    update(v*2, tl, tm, l, min(r, tm), add);
    update(v*2+1, tm+1, tr, max(l, tm+1), r, add);
    t[v] = max(t[v*2], t[v*2+1]);
  }
}
```

Lazy Segment Tree: Implementation

```
int query(int v, int tl, int tr, int l, int r) {
  if (l > r)
    return -INF;
  if (l == tl && tr == r)
    return t[v];
  push(v);
  int tm = (tl + tr) / 2;
  return max(query(v*2, tl, tm, l, min(r, tm)),
             query(v*2+1, tm+1, tr, max(l, tm+1), r));
}
```

Persistent Segment Tree: Save History

- `update(l, r, new_val, k)`
- `sum(l, r, k)`

- **Use vertex struct!**

Ordered Set

- A policy based data structure in g++ that keeps the unique elements in sorted order.
- It performs all the operations as performed by the set data structure in STL in **`log(n)`** complexity.

- In addition, it performs two additional operations also in logarithmic complexity.
  - **`order_of_key(k)`**: Number of items strictly smaller than k.
  - **`find_by_order(k)`**: k^{th} element in a set (counting from zero).

Ordered Set: Implementation (header)

```
#include <ext/pb_ds/assoc_container.hpp>
using namespace __gnu_pbds;
template <class T>
using Tree = tree<T, null_type, less<T>, rb_tree_tag,
tree_order_statistics_node_update>;
// find_by_order
// order_of_key
```


BREAK #1

Complete Search

- Try them all!

Complete Search

- Try them all!

- Also known as **bruteforce**.

Palindromic Squares (USACO, Rob Kolstad)

- Palindromes are numbers that read the same forwards as backwards. The number 12321 is a typical palindrome.

- Given a number base B ($2 \leq B \leq 20$, in base 10), print all the integers N ($1 \leq N \leq 300$, in base 10) such that the square of N is palindromic when expressed in base B; also print the value of that palindromic square. Use the letters 'A', 'B', and so on to represent the digits 10, 11, and so on.

- Print both the number and its square in base B.

I/O Format

- **Input Format**: A single line with B, the base (specified in base 10).

- **Output Format**: Lines with two integers represented in base B. The first integer is the number whose square is palindromic; the second integer is the square itself. NOTE WELL THAT BOTH INTEGERS ARE IN BASE B!

Sample I/O

- **Input**
  - 10

- **Output**
  - 1 1
  - 2 4
  - 3 9
  - 11 121
  - ...

Discuss for few minutes!

Solution Idea?

- Try them all!

N Queens Problem - Number of Possible Placements

- What to do when N=8 (regular chessboard)?

Discuss for few minutes!

Solution Idea?

- Try them all!


BREAK #2

Divide and Conquer

- **Divide** the problem into multiple subproblems,

Divide and Conquer

- **Divide** the problem into multiple subproblems,

- **Conquer** each subproblem,

Divide and Conquer

- **Divide** the problem into multiple subproblems,
- **Conquer** each subproblem,
- **Combine** the result!

Binary Exponentiation

- Calculate $\mathbf{a^n}$ fast!

Binary Exponentiation: Main Idea

$$a^n = \begin{cases} 1 & \text{if } n == 0 \\ \left(a^{\frac{n}{2}}\right)^2 & \text{if } n > 0 \text{ and } n \text{ even} \\ \left(a^{\frac{n-1}{2}}\right)^2 \cdot a & \text{if } n > 0 \text{ and } n \text{ odd} \end{cases}$$

Binary Exponentiation: Implementation

```
long long binpow(long long a, long long b) {
    if (b == 0)
        return 1;
    long long res = binpow(a, b / 2);
    if (b % 2)
        return res * res * a;
    else
        return res * res;
}
```

Merge Sort

- Sort an array of integers in ascending order.

- Time: O(n log n)
- Space: O(n)

Merge Sort: Implementation

```
// vector<int> v, aux;
void merge_sort(int left, int right) {
  if (left==right) return;
  int middle=(left+right)/2;
  merge_sort(left,middle), merge_sort(middle+1,right);
  aux.clear();
  int i=left, j=middle+1;
  while (i<=middle || j<=right)
    if (j>right || (i<=middle && v[i]<v[j]))
      aux.push_back(v[i]), i++;
    else aux.push_back(v[j]), j++;
  for (int i=left; i<=right; i++)
    v[i]=aux[i-left];
}
```

Fast Fourier Transform (FFT) and Karatsuba Algorithm

- Multiply two polynomials in **n log n** time (instead of naive n^2)!
- **Karatsuba** is similar, but works well for multiplying numbers!
- We will cover **FFT** and **Karatsuba** when we jump to math section later!


BREAK #3

Again, CodeForces Columbia SHP Algorithms Group

- Please join the following group: **https://codeforces.com/group/lfDmo9iEr5**

Practice Problems

- **Segment Tree**
  - https://codeforces.com/problemset/problem/920/F
  - https://codeforces.com/problemset/problem/242/E
  - https://codeforces.com/contest/474/problem/F
  - https://codeforces.com/problemset/problem/52/C
  - https://codeforces.com/contest/438/problem/D

Practice Problems

- **Ordered Set**
  - https://codeforces.com/contest/1676/problem/H2
  - https://codeforces.com/contest/1915/problem/F
  - https://codeforces.com/contest/652/problem/D

Next Week!

- Next week, we will cover some algorithm paradigms!
  - **Greedy**
  - **Dynamic Programming**

Slide Deck

- You may **always** find the slide decks from:
  - https://github.com/yongwhan/yongwhan.github.io/blob/master/columbia/shp


THANK YOU