

Formation beOI Data structures

Benoît LEGAT

beOI

21 février 2015

Introduction

Introduction Pedagogy

Useful data structures

Conclusion

Noob Just tell them to listen.

Pro Don't answer question they didn't ask. If they don't ask question, give them problems.

Expert (WIP) Give them problems and don't answer their question.

Being competitive

- Know your classics!
- Master the STL.
- Master jostream AND stdio.
- Avoid common coding guidance.
- Don't allocate with malloc, new, do static allocation of the worst case (you need to pass it anyway).
- Don't optimize the easy cases
- Don't do useless optimisation
- Beware time waster problems! They might kill you while you are trying to kill them.
- Pick the solution that is fastest to code and works.
- If want to commit suicide when thinking about your solution. That means there exists a better one.

Fast IO in Java

```
public class Main {
   public static void main(String[] args) throws IOException {
   BufferedReader in = new BufferedReader(new InputStreamReader(System.
        in)):
   PrintWriter out = new PrintWriter(new BufferedWriter(new
        OutputStreamWriter(System.out))):
   StringTokenizer st = new StringTokenizer(in.readLine());
   int tests = Integer.parseInt(st.nextToken());
   for (int test = 1: test <= tests: test++) {</pre>
   st = new StringTokenizer(in.readLine());
   int n = Integer.parseInt(st.nextToken());
10
11
   in.close():
12
   out.close(); // don't forget me
13
14
15
```

GDB

```
$ gdb a.out
   NO WARRANTY this program won't work !
   >> run
3
   >> run < input
   # Once it runs
6 >> break 42
   >> break gridland # Break the function gridland
   >> break 42 if n == 2
   # Once it is stopped
   >> step # Goes inside functions
10
   >> next # Doesn't
11
   >> continue
   >> bt
13
   >> bt full
   >> frame 3
   >> info locals
16
   >> quit
17
```

Useful data structures

Introduction

Useful data structures

Basic data structures

STL

Hash Map

Binary Search Tree

Heap

Union find

Segment Trees

Fenwick Tree

Sparse table

Conclusion

- Keys : $\{0, \ldots, n-1\}$
- $\bullet \ \mathsf{Access} \ \mathsf{in} \ \mathcal{O}(1)$
- Modify in $\mathcal{O}(1)$

With dynamic array (ArrayList or std::vector), n is not fixed.

- Add in amortized $\mathcal{O}(1)$
- Remove in $\mathcal{O}(n)$ if not at the end

Linked list

- Access and modify in $\mathcal{O}(n)$
- ullet Modify and modify in $\mathcal{O}(1)$ at extremities
- ullet Add and remove in $\mathcal{O}(1)$

Stack and Queue

Specific Linked List (a bit faster and limited capability)

Stack LIFO, Last in First out.

```
Stack<Integer> s = new Stack<Integer>();
q.push(1);
int top = q.peek(); // just watch
top = q.pop();
```

Queue FIFO, First in First out.

```
Queue < Integer > q = new LinkedList < Integer > ();
q.add(1);
int first = q.peek(); // just watch
fitst = q.poll();
```

Classical Example I

Biggest rectangle (sometimes in the sense or perimeter, sometimes area)



medium NWERC 2010 Problem G hard COCI 2014/2015 Contest #5 Task Neo

Classical Example II

- With DP, $\mathcal{O}(n^3)$
- If we only check the size when the rectangles are "closed", $\mathcal{O}(n^2)$

```
class Rectangle {
1
       int start_row;
       int width;
3
4
       Stack < Rectangle > [] dp = new Stack [ncol]; // + init them
5
       int best = 0
       for row = 0... row -1
7
          int width = 0
          for col = ncol-1..0 // to process width in $\bigoh(1)$
            if free cell
10
11
              width++:
              int start row = row
12
              while dp[col].peek().width >= width
13
                Rectangle closed = dp[col].pop();
14
                best = Math.max(best. closed.size(row))
15
                start_row = Math.min(row, closed.start_row)
16
              dp[col].add(new Rectangle(start row, width))
            else
18
              while !dp[col].empty()
19
                best = Math.max(best, dp[col].pop().size(row))
20
              width = 0
21
       for col = ncol-1..0 // don't forget !!
22
          close all rectangles of dp[col] -> update best // Lines 19,20
23
```

Classical Example III

- If you want to do it in C, it will take you 1 h, you loose a precious time
- With the STL, you kill it in 5 min!

Input

Don't Panic

Mostly Harmless
42
Don't Panic
The Hitchhiker's Guide

Output

42 20.0000
Don't Panic 40.0000
Mostly Harmless 20.0000
The Hitchhiker's Guide 20.0000

- System.out.printf("%s %.4f\n", s, p)
- Exactly 4 decimals cout << setiosflags(ios::fixed)<< setprecision(4)
- Strings with spaces std::getline

Unique key

sorted unsorted

std::set std::unordered set

TreeSet HashSet

std::map std::unsorted map association

TreeMap HashMap

complexity $\mathcal{O}(\log(n))$ $\mathcal{O}(1)$ on average

Different elements can have the same key

existence

unsorted sorted

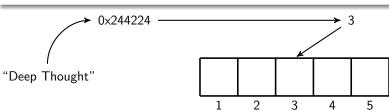
existence std::multiset std::unordered multiset association std::multimap std::unsorted multimap complexity $\mathcal{O}(\log(n))$ $\mathcal{O}(1)$ on average

Principle

There is a number N of buckets. The key is mapped to a bucket by hashing it to a number between 0 and N-1.

- 1. Transform the key to a number k of type $size_t$;
- 2. Get that number between 0 and N-1 (e.g. $k \pmod{N}$);
- 3. Access the bucket with that index.

For now, $\mathcal{O}(1)$!



Hash Map II

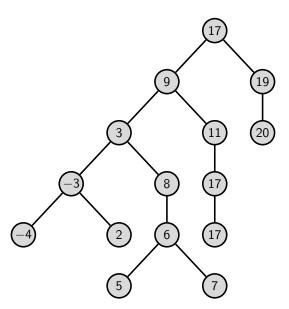
Collision

But there can be collision so it is $\mathcal{O}(1)$ on average! In case of collision, 2 solutions

- Store in each bucket all the collisions;
- Probe a empty spot (linear probing, quadratic probing, ...).

It is only $\mathcal{O}(1)$ on average if the load factor is good (e.g. $\ll 1$). When load factor get close to 1, increase N.

Binary Search Tree I



Binary Search Tree II

To get $\mathcal{O}(\log(n))$

- Balanced
- Comparison in $\mathcal{O}(1)$ (strings have $\mathcal{O}(m\log(n))$ where m is their length)
- Balancing operation in log(n)

Balanced trees are tricky to code! Use std::set, TreeSet and std::map, TreeMap!

When to use a BST or a hash map

- When you want an array to be indexed by another thing than an int
- When an array is too small and you don't need all
- (BST only) When you want a structure to be
 - ▶ an array to get $\mathcal{O}(\log(n))$ binary search access time
 - ▶ a linked list to get $\mathcal{O}(1)$ insertion time

In the STL

```
TreeMap < String, Integer > map = new TreeMap < String, Integer > ();
1
   map.put("Test", 42):
2
   map.get("Test"): // -> 42:
   map.containsKey("Test"); // -> true
   map.containsKey("Toto"); // -> false
5
   for (Map.Entry < String, Integer > entry : map.entrySet()) {
       String test = entry.getKey();
7
       int fortytwo = entry.getValue();
8
   }
9
10
   std::map<std::string, int> map();
11
   map["Test"] = 42;
12
   map["Test"]: // -> 42
13
   map.count("Test"); // -> 1
14
   map.count("Toto"); // -> 0 (only 0 or 1 since not multimap)
15
   for (std::map<std::string,int>::iterator it=map.begin(); it!=map.end
16
        (): ++it)
        std::cout << it->first << " => " << it->second << '\n':
17
```

Longest Increasing Subsequence: LIS, a classic!

- Find an $\mathcal{O}(n^2)$ algorithm.
- Improve it to $\mathcal{O}(n\log(n))$.
- Solve problem 481 in uva with it (even if $\mathcal{O}(n^2)$ is enough).

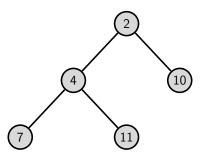
Let's do a problem, shall we?

Do problem 10954 from uva!

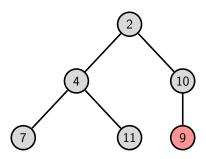
Heap

If we are only interested by the minimum, the BST is simplified and much faster and easier to code. Still $\mathcal{O}(\log(n))$ though.

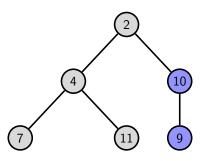
The tree can be always complete!



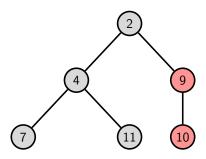
Push I



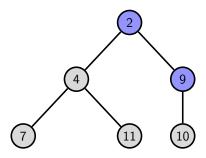
Push II



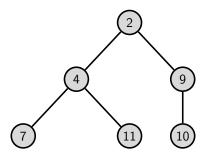
Push III



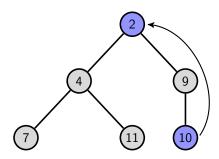
Push IV



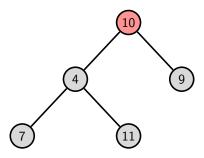
Push V



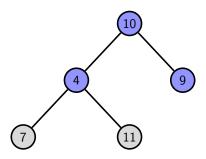
Pop I



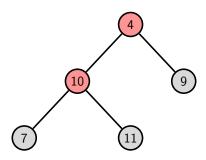
Pop II



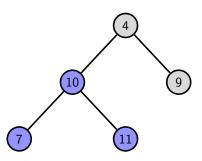
Pop III



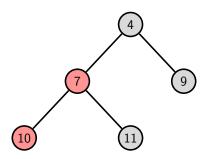
Pop IV



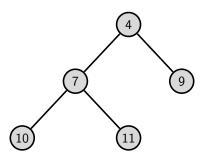
Pop V



Pop VI



Pop VII



Implémentation I

```
int hsize;
   int hid[MAX]; // give id from index
   int hval[MAX]; // give val from index
   int hlookup[MAX]; // give index from id
5
   void heap swap (int a, int b) { // a and b are the indexes
   int tmp = hval[a];
   hval[a] = hval[b]:
   hval[b] = tmp;
10
   hlookup[hid[a]] = b;
11
   hlookup[hid[b]] = a;
12
13
   tmp = hid[a];
14
   hid[a] = hid[b];
15
   hid[b] = tmp;
17
18
   void heap_up (int a) { // a is the index
19
     int up = (a-1)/2;
20
   if (0 < a && hval[a] < hval[up]) {</pre>
   heap_swap(a, up);
22
   heap up(up);
24
25
26
```

Implémentation II

```
void heap_down (int a) { // a is the index
27
   int left = 2*a+1, right = 2*a+2;
28
    if (left < hsize && (hsize <= right || hval[left] < hval[right]) &&
29
        hval[left] < hval[a]) {
    heap swap(a, left);
30
    heap_down(left);
31
32
    else if (right < hsize && hval[right] < hval[a]) {</pre>
33
    heap_swap(a, right);
34
    heap down(right);
36
37
```

A wild problem appear

n objects $0, \ldots, n-1$ are initially alone in a set. We have two types of query

- We merge 2 sets designated by 2 respecitive members
- We ask if 2 objects are in the same set

Union find solution I

```
class UnionFind {
      int rank[MAX N];
      int leader[MAX N];
3
      UnionFind(int n) {
        memset(rank, 0, n * sizeof(int));
5
        for(int i = 0: i < n: i++) leader[i] = i:</pre>
      int find(int a) {
8
        if(a != leader[a])
q
          leader[a] = find(leader[a]);
10
        return leader[a];
11
12
      void union(int a, int b) {
13
        int leaderA = find(a):
14
        int leaderB = find(b):
15
        if(leaderA == leaderB) return;
16
        if(rank[leaderA] > rank[leaderB]) {
17
          union(leaderB. leaderA): return:
18
19
        leader[leaderA] = leaderB:
20
        if (rank[leaderA] == rank[leaderB])
21
          rank[leaderB]++;
23
   };
24
```

Union find solution II

Complexity

time Amortized $\mathcal{O}(\alpha(n))$.

space $\mathcal{O}(n)$.

Example: gridland I

Grid $n \times m$ avec $1 < n, m < 1 \times 10^3$. Two squares are connected if they are activated and their is a path of activated square from one to the other. Initially, squares are only connected to themselves.

There is $1 < q < 1 \times 10^6$ queries

add a x y activate the square at (x, y)

connected? c xa ya xb yb see if (xa, ya) and (xb, yb) are connected.

Example: gridland II













Other examples

- uhunt.felix-halim.net 2.4.2.
- Connected componentents (overkill), Kruskal (impossible to get $\mathcal{O}(n\log(n))$ without it) (see next day!)

Segment Tree I

- Dynamic Range Minimum Query
- Dynamic Range Sum Query (Fenwick Tree is simpler)
- Dynamic Range Anyfunction Query

```
class SegmentTree {
1
      int[] st, A;
      int n:
3
      int left (int p) { return p << 1; }</pre>
      int right(int p) { return (p << 1) + 1; }</pre>
6
      void build(int p, int L, int R) {
        if (L == R)
8
          st[p] = L: // or R
        else {
10
          int mid = (L + R) / 2:
11
          build(left(p) , L      , mid);
12
          build(right(p), mid + 1, R);
13
          int p1 = st[left(p)], p2 = st[right(p)];
14
          st[p] = (A[p1] \le A[p2]) ? p1 : p2;
15
      } }
16
17
      int rmq(int p, int L, int R, int i, int j) { // O(log n)
18
        if (i > R || j < L) return -1; // outside query range
19
        if (i <= L && R <= j) return st[p]; // inside query range</pre>
20
```

Segment Tree II

```
int mid = (L + R) / 2;
21
       22
       int p2 = rmq(right(p), mid + 1, R, i, j);
23
24
       if (p1 == -1) return p2: // outside query range
       if (p2 == -1) return p1;
26
       return (A[p1] <= A[p2]) ? p1 : p2; }</pre>
27
28
     int update(int p, int L, int R, int i, int j, int v) {
29
       if (i > R || i < L)
                                      // outside update range
30
        return st[p]:
31
       //if (i <= L && R <= j) // could be lazy here !! Depends on
32
           application
       if (L == R) {
33
        A[i] = v;
34
        return st[p] = L: // or R
35
36
       int mid = (L + R) / 2:
37
       38
       int p2 = update(right(p), mid + 1, R , i, j, v);
39
       return st[p] = (A[p1] <= A[p2]) ? p1 : p2;</pre>
40
41
42
     public:
43
44
     SegmentTree(int[] _A) {
45
       A = _A; n = A.length;
46
```

Segment Tree III

```
st = new int[4 * n];
for (int i = 0; i < 4 * n; i++) st[i] = 0;
build(1, 0, n - 1);
}
int rmq(int i, int j) { return rmq(1, 0, n - 1, i, j); }
int update_point(int i, int v) {
   return update(1, 0, n - 1, i, i, v); }
int update_interval(int i, int v) {
   return update(1, 0, n-1, i, j, v); }
};</pre>
```

- IOI-2013 day 2 : "game"
- http://codeforces.com/problemset/problem/474/E

Fenwick Tree I

Dynamic Range Sum Query

What are the numbers smaller that 1011001000?

- 1011000???
- 1010??????
- 100???????
- 0????????

```
\begin{split} \mathsf{rsq}(1011001000) &= \mathsf{ft}(1011001000) + \mathsf{ft}(1011000000) \\ &\quad + \mathsf{ft}(1010000000) + \mathsf{ft}(1000000000) \end{split}
```

```
1     adjust(01011001000,1):
2         ft[01011010000]++
3         ft[01011100000]++
4         ft[01100000000]++
5         ft[10000000000]++
```

Fenwick Tree II

```
class FenwickTree {
1
2
      int *ft:
3
     int n;
      int LSOne(int S) { return (S & (-S)); }
4
     public:
5
     FenwickTree(int n) { // ignore index 0
6
      this -> n = n:
7
       ft = new int[n+1];
8
        for (int i = 0; i \le n; i++) ft[n] = 0;
9
10
     int rsq(int b) {  // returns RSQ(1, b)     PRE 1 <= b <= n</pre>
11
        int sum = 0; for (; b > 0; b -= LSOne(b)) sum += ft[b];
12
        return sum:
13
14
      int rsq(int a, int b) { // returns RSQ(a, b) PRE 1 <= a,b <= n
15
        return rsq(b) - (a == 1 ? 0 : rsq(a - 1));
16
17
     void adjust(int k, int v) { // n = ft.size() - 1 PRE 1 <= k <= n</pre>
18
        for (; k <= n; k += LSOne(k)) ft[k] += v;</pre>
19
20
   };
21
```

Fenwick Tree III

Exemples:

- NWERC 2011 Problem C
- http://codeforces.com/contest/504/problem/B
- http://codeforces.com/problemset/problem/459/D

Sparse Table

Static Range Minimum Query m[i][j] stores the smallest of $[i;i+2^j[.min([a;b[)=min(m[a][k],m[b-2^k][k])$ for k such that $2^{k-1} < b-a \le 2^k$

Get a tree flat with DFS then apply RSQ and RMQ.

- Least Common Ancestor : LCA
- http://codeforces.com/contest/383/problem/C

Conclusion

Introduction

Useful data structures

Conclusion

- Codeforces
- Codechef
- Usaco
- UVa online judge
- Competitive Programming 3: https://sites.google.com/site/stevenhalim/
- uhunt.felix-halim.net
- IOI syllabus : http://people.ksp.sk/~misof/ioi-syllabus/