Answer Pages

Question 21 (pushAll) answer:

Question 22 (KStep) answer:



Question 23 (hasMore) answer:



Question 24 (step1) answer:

Question 25 (step1 running time) answer:

int vet = step(int k) {

int vet = step1();

for (int i = 0; i (k-1; itt) {

Treebble temp = st.pop()

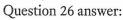
if (!has More()) {

veturn vet;

push All (temp -> right);

return vet;

}



Lower Bound	0(1)
Average	Och
Upper Bound Case	0(1)



```
Question 27 (buildPerfectTree) answer:
Qualitree Node * Quadtree: build Perfect Tree(int K, RGBAPixel p) {
        if((K-1) == 0) 11 (K-1)(0) }
             Quad-eree Node * temp = new Quad-tree Node ();
             temp- element = p;
            temp > nwChild = Nuzz;
             temp - ne Child = NULL;
            temp > swChild = NULL;
            temp -> Se Child = NULL;
            return temp,
       Quad tree Nocle * rt = new QuadtreeNode ();
        rt > nw Child = build Perfect Tree (K-1, p),
        re - ne Child = build Perfect True (16-1, p);
        re-swchild = build Perfect True (16-), p1;
        re - sechild = build Perfect Trace (16-1, p1;
       return rt
Question 28 (perfectify) answer:
Void Quadtree: perfectify (int levels) {
          "perfector (levels, voot);
  void Quaderee: perfector (in levels, Quaderer Node * ptr) }
      if Cptr == Nall) { return; }
      if (per > nwChild == NULL) { per + nwChild = build Perfect Tree (levels - 1, pers denent);}
      if (per = ne Child == Null) {per + ne Child = build Perfect Tree (levels -1, per > element); }
     if (ptr +se(hild == NULL) & ptr + se Child = build Perfect Tree (levels -1, ptr + element); }
     if (ptr = swChild == Nall) { ptr = swChild = build Perfect Tree (levels -1, per + element); }
     perfector (levels -1, per - nw Child);
perfector (levels -1, per -> ne Child);
     perfector ( levels -1, per > swChild);
     perfector [ levels -1, per -se Chila);
Question 29 (perfectify running time) answer:
```

a)

b)

c)

d)

e)

f)

Question 30 answer:

You may answer this question by filling in these blanks, or use the blank space for your own proof/disproof.



Preliminaries Let H(n) denote the maximum height of an n-node SAVL tree, and let N(h) denote the minimum number of nodes in an SAVL tree of height h. To prove (or disprove!) that $H(n) = \mathcal{O}(\log n)$, we attempt to argue that HCM < log2 h

$$H(n) \leq 3\log_2 n$$
, for all n

Rather than prove this directly, we'll show equivalently that

$$N(h) \geq \frac{1}{2}$$
, (1pt)

Proof For an arbitrary value of h, the following recurrence holds for all SAVL Trees:

We can simplify this expression to the following inequality, which is a function of N(h-3):

$$N(h) \ge 3 \times (h-3)$$
, (1pt)

By an inductive hypothesis, which states:

we now have

$$N(h) > \frac{\text{for every } j \ge 0, \text{ an save error of height } h}{N(h) > 0}$$
, (1pt)

will have $3(h-3)$ nooles

 $N(h) > 0$

= part (a) answer, (1pt)

which is what we wanted to show.

Given that $2^0=1, 2^{1/3}\approx 1.25,$ and $2^{2/3}\approx 1.58,$ what is your conclusion? Is an SAVL tree $O(\log n)$ or not? (Circle one): (2pt)



NO

Overflow Page

Use this space if you need more room for your answers.

