Finishing Mergesort and Quicksort, then ADTs based on queues

2/8/16
Announcements

• Quiz makeup: right at the end of class and all day tomorrow
• Next reading: Chapter 12 and Java Interlude 5
  – Delay this to next Monday
• Midterm grades entered for everyone: based just on original quizzes 1-3 and HW 1-5
  – Gave quizzes weight of 14% and HW a weight of 45%
  – Actual grade to be affected by later work, makeup quizzes, dropping lowest quiz, clicker participation, and final
Recall: Logarithms

\[ \log_b x = \text{"log base } b \text{ of } x\]

= power of \( b \) that gives \( x \)

= number of times you can divide \( x \) by \( b \) before getting 1

\[ \log_2 1 \text{ is 0} \]
\[ \log_2 2 \text{ is 1} \]
\[ \log_2 4 \text{ is 2} \]
\[ \log_2 32 \text{ is 5} \]
\[ \log_2 100 \text{ is between 6 and 7} \]
What is $\log_2 1000$?

A. Between 7 and 8
B. Between 8 and 9
C. Between 9 and 10
D. Between 10 and 11
E. None of the above
What is $\log_2 1000$?

A. Between 7 and 8
B. Between 8 and 9
C. Between 9 and 10
D. Between 10 and 11
E. None of the above
Properties of logarithms

• \( \log_b xy = \log_b x + \log_b y \)

• \( \log_b (x/y) = \log_b x - \log_b y \)

• \( \log_b x^y = y \log_b x \)

• \( \log_b a = \frac{\log_c a}{\log_c b} \) for any c
Recall: Running time of mergesort

Task of sorting $n$ numbers

- Task of sorting $n/2$ numbers
  - Task of sorting $n/4$ numbers
  - Task of sorting $n/4$ numbers
  - Task of sorting $n/4$ numbers

- Task of sorting $n/2$ numbers
  - Task of sorting $n/4$ numbers
  - Task of sorting $n/4$ numbers
  - Task of sorting $n/4$ numbers

... and so on
Quicksort

void quicksort(int[] a, int lo, int hi) {
    //sort a[lo..hi]
    if(hi <= lo) return;  //base case: empty array

    //split a into values <= pivot and >= pivot
    int j = partition(a, lo, hi);  //j = index of pivot

    quicksort(a, lo, j-1);        //sort left part
    quicksort(a, j+1, hi);        //sort right part
}

Partition

int partition(int[] a, int lo, int hi) {
    int i = lo, j = hi+1;  // left and right scan indices
    int v = a[lo];  // pivot value
    while(true) {
        while(a[++i] < v) if(i == hi) break;
        while(v < a[--j]) if(j == lo) break;
        if(i >= j) break;
        swap(a, i, j);  // swap a[i] & a[j]
    }
    swap(a, lo, j);
    return j;
}
int partition(int[] a, int lo, int hi) {
    int i = lo, j = hi+1;          //left and right scan indices
    int v = a[lo];                //pivot value
    while(true) {
        while(a[++i] < v) if(i == hi) break;
        while(v < a[--j]) if(j == lo) break;
        if(i >= j) break;
        swap(a, i, j);               //swap a[i] & a[j]
    }
    swap(a, lo, j);
    return j;
}
Partition

```c
int partition(int[] a, int lo, int hi) {
    int i = lo, j = hi+1;   //left and right scan indices
    int v = a[lo];         //pivot value
    while(true) {
        while(a[++i] < v) if(i == hi) break;
        while(v < a[--j]) if(j == lo) break;
        if(i >= j) break;
        swap(a, i, j);       //swap a[i] & a[j]
    }
    swap(a, lo, j);
    return j;
}
```

What is partition’s running time?
A) O(1)  D) O(n^2)
B) O(log n)  E) None of the
C) O(n)    above
Running time of Quicksort