

Problem of the Day

Assume that $n = 2^{k+1} - 1$.

Solve this recurrence using repeated substitution showing all your work:

$$T(n) = (n+1) + 2 T((n-1)/2), \quad T(1) = 1.$$

Hint: These are a lot easier to solve if you express the equations in terms of k before starting.

For more of a challenge, solve this one:

$$T(n) = n + 2 T((n-1)/2), \quad T(1) = 1.$$

Announcements

Assignment 2A Programming: due Thursday Oct. 12 at 11:55pm. **Feedback will be given soon for assignment 1A.**

Assignment 2B: Written: due Monday Oct. 16 at the beginning of class.

We have labs this week.

We have no labs next week.

There are no classes on Monday Oct. 9 because it is Thanksgiving.

Quicksort

A well-known sorting algorithm developed by C. A. R. Hoare in 1962 that, on average, makes $\Theta(n \log n)$ comparisons to sort n items. However, **in the worst case, it makes $\Theta(n^2)$ comparisons.**

Typically, quicksort is significantly faster in practice than other $\Theta(n \log n)$ algorithms, because its inner loop can be efficiently implemented on most architectures, and in most real-world data, it is possible to make design choices which minimize the probability of requiring quadratic time.

Quicksort for Linked Lists

1. Choose a key value x to be the pivot.
2. **[Divide]** Break the problem into three subproblems:
 - P1: Keys $< x$.
 - P2: Keys equal to x .
 - P3: Keys $> x$.
3. **[Conquer]** Solve the P1 and P3 recursively.
4. **[Marry]** The answer is that of P1 followed by P2 followed by P3.

Quicksort (Arrays)

1. Choose a key value x to be the pivot.
2. **[Divide]** Break the problem into three subproblems:
 - P1: Keys $< x$.
 - P2: The pivot x .
 - P3: Keys $\geq x$.
3. **[Conquer]** Solve the P1 and P3 recursively.
4. **[Marry]** The answer is that of P1 followed by P2 followed by P3.

```
public class QuickSort {
```

```
// Quicksort code (modified from Sedgewick)
```

```
public static void quicksort(int[] A)
```

```
{
```

```
    shuffle(A); // to guard against worst case
```

```
    quicksort(A, 0, A.length - 1);
```

```
}
```

<http://www.cs.princeton.edu/introcs/42sort/QuickSort.java.html>

```
// quicksort A[left] to A[right]
```

```
public static void quicksort(int[] A,  
                             int left, int right)  
{  
    if (right <= left) return;  
  
    int pivot_pos = partition(A, left, right);  
  
    quicksort(A, left, pivot_pos-1);  
  
    quicksort(A, pivot_pos+1, right);  
}
```

```

// partition A[left] to A[right]
private static int partition(int [] A,
                             int left, int right) {
int i = left; int j = right-1;

while (true)
{
    while (A[i] < A[right]) {i++;}
    while (j > left && A[right] < A[j]) {j--};

    if (i >= j) break;
    swap(A, i, j); i++; j--;
}
swap(A, i, right); // Put pivot element into place
return i;
}

```

Quicksort

