## Administrivia

**Slide 1**

- Reminder: Homework 4 due 5pm today.

## Analysis of Algorithms — Review/Recap

**Slide 2**

- Often useful to be able to estimate algorithm's execution time as a function of "problem size".

  Customary to skip over housekeeping operations and count only "important stuff" — arithmetic operations, comparisons, etc.
- Useful in comparing efficiency of different algorithms for same problem. Also useful in determining feasibility of single algorithm. (E.g., something that requires evaluating $N!$ possibilities will not be practical for large $N$.)

Slide 3

## Analysis of Algorithms, Longer Example

- Look at several algorithms for computing $a^b$, for $b$ a positive integer. First version:

```
double exp(double a, int b) {
    double temp = a;
    for (int i = 1; i < b; ++i)
        temp *= a;
    return temp;
}
```

- How many multiplications needed?

Slide 4

## Analysis of Algorithms, Longer Example Continued

- We could also express this recursively:

```
double exp(double a, int b) {
    if (b == 1)
        return a;
    else
        return a * exp(a, b-1);
}
```

Does this work? (Yes. Why?)

- How to figure out how many multiplications? Define and solve a recurrence relation.

**Slide 5**

## Analysis of Algorithms, Longer Example Continued

- We could also express this recursively another way:

```
double exp(double a, int b) {
    if (b == 1)
        return a;
    else {
        double temp = exp(a, b/2);
        if (b % 2 == 0)     return temp * temp;
        else                return temp * temp * a;
    }
}
```

  Does this work? (Yes. Why?)

- How to figure out how many multiplications? Define and solve a recurrence relation.

**Slide 6**

## Analysis of Algorithms, Continued

- More complicated (but faster) $a^b$ algorithm — example of "divide and conquer" algorithms. General form:

```
if (base case)
    solve
else {
    split into 2 subproblems
    solve subproblem(s)
    merge subsolutions
}
```

- In general, recurrence relation for work involved has the form

$$S(n) = cS(n/2) + g(n), \quad \text{for } n = 2^m, n > 1$$

  for which we can derive a formula — equation (6) on p. 152.

Slide 7

## Analysis of Algorithms, Continued

- Example — recurrence relation for exponentiation algorithm:

$$\begin{aligned} M(1) &= 0 \\ M(n) &= 1 + M(n/2), \text{for } n = 2^m, n > 1 \end{aligned}$$

Slide 8

## Minute Essay

- How many comparisons are needed to sort an array of $N$ elements using bubble sort?:

```
for (int i = 0; i < N-1; ++i) {
    for (int j = 0; j < N-1-i; ++j) {
        if (a[j+1] < a[j])
            swap(a[j+1], a[j]);
    }
}
```

## Minute Essay Answer

- `N-1` + `N-2` + `N-3` + ... + `0`, i.e., `(N-1)*N/2`. (One comparison per trip through the inner loop, and the number of inner-loop trips for each trip through the outer loop depends on the value of `i`.)