ORF 201

COMPUTER METHODS FOR PROBLEM SOLVING

Lecture 10

Sorting

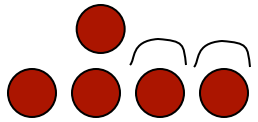
Bubble Sort

```
public void bubbleSort(Zip[] zips, int n)
{
    int j, k;

    for (k=n-1; k>=0; k--) {
        for (j=0; j<k; j++) {
            if (zips[j].zip > zips[j+1].zip) {
                swap(zips, j, j+1);
            }
        }
    }
}
```

Complexity:

- $n^2/2$ comparisons
- $n^2/4$ swaps on average
- $n^2/2$ swaps in worst case



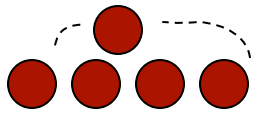
Insertion Sort

```
public void insertSort(Zip[] zips, int n)
{
    int i, j=-1, k, max;

    for (k=n; k>1; k--) {
        max = 0;
        for (i=0; i<k; i++) {
            if (zips[i].zip > max) {
                j = i;
                max = zips[i].zip;
            }
        }
        swap(zips, k-1, j);
    }
}
```

Complexity:

- $n^2/2$ comparisons
- n swaps on average
- n swaps in worst case

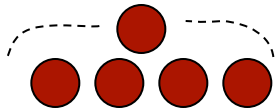


Quick Sort

```
public void quickSort(Zip[] zips, int left, int right)
{
    int i, last;

    if (left >= right) return;
    swap(zips, left, (left+right)/2);
    last = left;
    for (i=left+1; i<=right; i++) {
        if (zips[i].zip < zips[left].zip) {
            last++;
            swap(zips, last, i);
        }
    }
    swap(zips, left, last);
    quickSort(zips, left, last-1);
    quickSort(zips, last+1, right);
}
```

Recursive!



Complexity:

- (2/3)*n* log *n* comparisons
- (1/3)*n* log *n* swaps on average
- 2*n* log *n* comparisons in worst case
- *n* log *n* swaps in worst case

Swapping

```
void swap(Zip[] zips, int i, int j)
{
    Zip temp;

    if (animate) {
        GL.color(Color.black);
        GL.pnt2(i, zips[i].zip);
        GL.pnt2(j, zips[j].zip);
    }

    temp = zips[j];
    zips[j] = zips[i];
    zips[i] = temp;

    if (animate) {
        GL.color(Color.white);
        GL.pnt2(i, zips[i].zip);
        GL.pnt2(j, zips[j].zip);
    }
}
```

`zips[i]` is a pointer to **`Zip`**. Even if class **`Zip`** were huge, swap would be easy.

Would be nice if **`swap`** didn't have to know about **`Zip`**.

Complexity

log(n)	n	n^2	n^3	2^n
0.00	1	1	1	2
0.69	2	4	8	4
1.10	3	9	27	8
1.39	4	16	64	16
1.61	5	25	125	32
1.79	6	36	216	64
1.95	7	49	343	128
2.08	8	64	512	256
2.20	9	81	729	512
2.30	10	100	1000	1024
3.00	20	400	8000	1.0486e+06
3.40	30	900	27000	1.0737e+09
3.69	40	1600	64000	1.0995e+12
3.91	50	2500	125000	1.1259e+15
4.09	60	3600	216000	1.1529e+18
4.25	70	4900	343000	1.1806e+21
4.38	80	6400	512000	1.2089e+24
4.50	90	8100	729000	1.2379e+27
4.61	100	10000	1000000	1.2677e+30
5.30	200	40000	8000000	
5.70	300	90000	27000000	
5.99	400	160000	64000000	
6.21	500	250000	125000000	
6.40	600	360000	216000000	
6.55	700	490000	343000000	
6.68	800	640000	512000000	
6.80	900	810000	729000000	
6.91	1000	1000000	1000000000	

Complexity	n^3	2^n
$n = 10$	0.001 sec	0.001 sec
$n = 25$	0.016 sec	33.55 sec
$n = 50$	0.125 sec	35.70 yrs
$n = 75$	0.422 sec	11,979,621 cen