N Squared Sorts

Wednesday, March 2, 2022 5:32 AM

8, 2, 7, 1, 3, 12, 4, 5

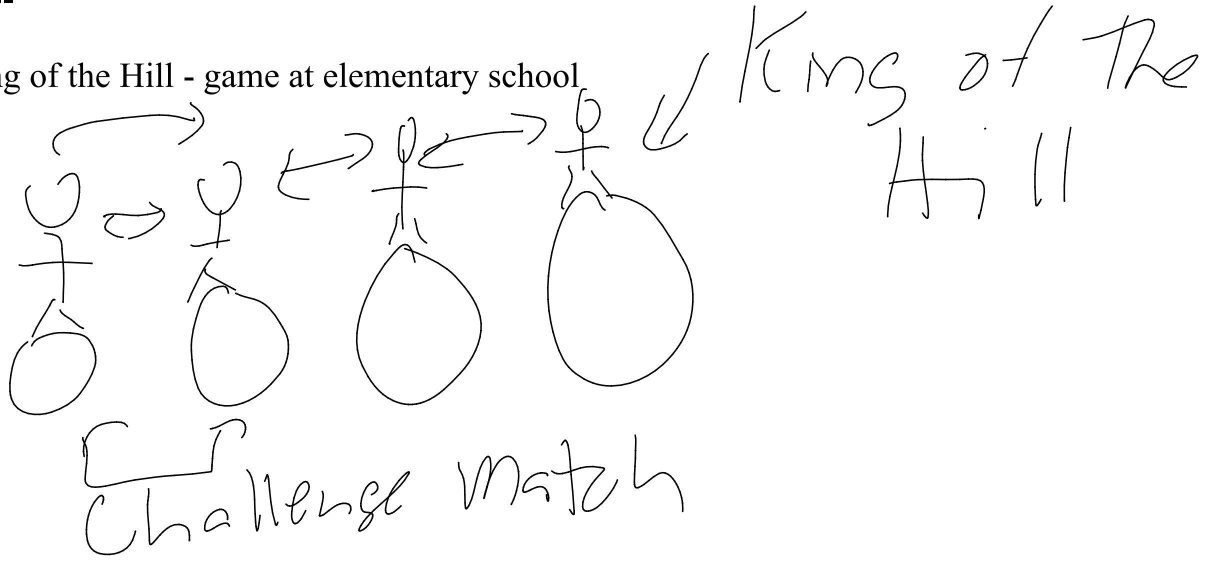
Reason that SOOO many sorting algorithms were covered in courses in the past and that so much attention was placed on it before, is that it's an EXCELLENT testbed to understand many concepts that apply to generating and analyzing algorithms.

Why do I put less emphasis on it now?
Most APIs have a sort function and generally speaking these work quite well…Also, there's more new knowledge in CS than there used to be when my father was a student!

1. Bubble Sort
2. Insertion Sort
3. Selection Sort

Bubble Sort

It's like King of the Hill - game at elementary school



Basic idea: run challenge matches from left to right, swapping when necessary, and at the end of this, the maximum value will end up at the end.

8, 2, 7, 1, 3, 12, 4, 5

8	2	7	1	3	12	4	5
2	8	7	1	3	12	4	5
2	7	8	1	3	12	4	5
2	7	1	8	3	12	4	5

2	7	1	3	8	12	4	5
2	7	1	3	8	12	4	5
2	7	1	3	8	4	12	5
2	7	1	3	8	4	5	12

This is one iteration of Bubble Sort. Now just blue is left to sort.

8	2	7	1	3	12	4	5	
2	7	1	3	8	4	5	12	Iter #1
2	1	3	7	4	5	8	12	Iter #2
1	2	3	4	5	7	8	12	Iter #3
1	2	3	4	5	7	8	12	
1	2	3	4	5	7	8	12	
1	2	3	4	5	7	8	12	
1	2	3	4	5	7	8	12	

For code, first we want a loop going to n, then n-1, n-2, etc. --> we are counting down!

So, outer loop will look like

```
for (int i=n-1; i>0; i--)
```

where n is the length of the array.

Insertion Sort

This is how most people sort cards in their hand…

8, 2, 7, 1, 3, 12, 4, 5, yellow is sorted list of size 1. Now, insert 2 into it.

2, 8, 7, 1, 3, 12, 4, 5, now insert 7 into sorted yellow list

2, 7, 8, 1, 3, 12, 4, 5, now insert 1 into sorted yellow list.

When we insert an item, we compare it to its direct left, and if out of order, we swap. If not, we stop that iteration.

8	2	7	1	3	12	4	5
2	8	7	1	3	12	4	5
2	7	8	1	3	12	4	5
1	2	7	8	3	12	4	5

1	2	3	7	8	12	4	5
1	2	3	7	8	12	4	5
1	2	3	4	7	8	12	5
1	2	3	4	5	7	8	12

We start with index 1 and insert that. Then index 2, etc.

outer loop for array length n.

for (int i=1; i<n; i++)

Selection Sort

Look through all the cards, keeping track of the maximum and swap that to the end. Repeat until sorted!

8	2	7	1	3	12	4	5

maxIndex = 0,5
j = 0,1,2,3,4,5

maxIndex $\boxed{\cancel{0}}$ 5

j $\boxed{\cancel{0}}$ ~~1~~ ~~2~~ ~~3~~ ~~4~~ ~~5~~ ~~6~~ 7

largest value in index 5, swap this with index 7…

8	2	7	1	3	12	4	5
8	2	7	1	3	5	4	12
4	2	7	1	3	5	8	12
4	2	5	1	3	7	8	12
4	2	3	1	5	7	8	12
1	2	3	4	5	7	8	12
1	2	3	4	5	7	8	12
1	2	3	4	5	7	8	12

Bubble Sort Analysis

Loop runs ~~n~~ n-1 times, n-1, n-2, …, 1

Run time is

$$\sum^{n-1} i = \frac{n(n-1)}{2} = O(n^2)$$

Run time is $$\sum_{i=1} i = \frac{}{2} = O(n)$$

The way I wrote it…the best case, worst case and average case are n

If you go to Wikipedia, then cut out of the loops if the array is sorted…so for their version best case is O(n)…DON'T USE THIS VERSION FOR OUR CLASS!!!

Insertion Sort

Worst case, we insert every time, we do 1 insertion, 2 insertions, 3 insertions, …, n-1 insertions…same exact analysis as bubble sort.

Best case: 0 insertions, 0 insertion, etc. O(n)

Average case: On average, insertion #i will swap i/2 steps, so our new sum for average case is

$$\sum_{i=1}^{n-1} \frac{i}{2} = \frac{(n-1)n}{4} = O(n^2)$$

Selection Sort

First loop runs in n steps, then n-1 steps, then n-2 steps, etc.
Best, worst and average case run time for selection sort is $O(n^2)$.

Algorithm	Best	Average	Worst
Bubble	n^2	n^2	n^2
Insertion	n	n^2	n^2
Selection	n^2	n^2	n^2