Insertion Sort

Analysis and Correctness

Session 3 - 12th Jan, 2022

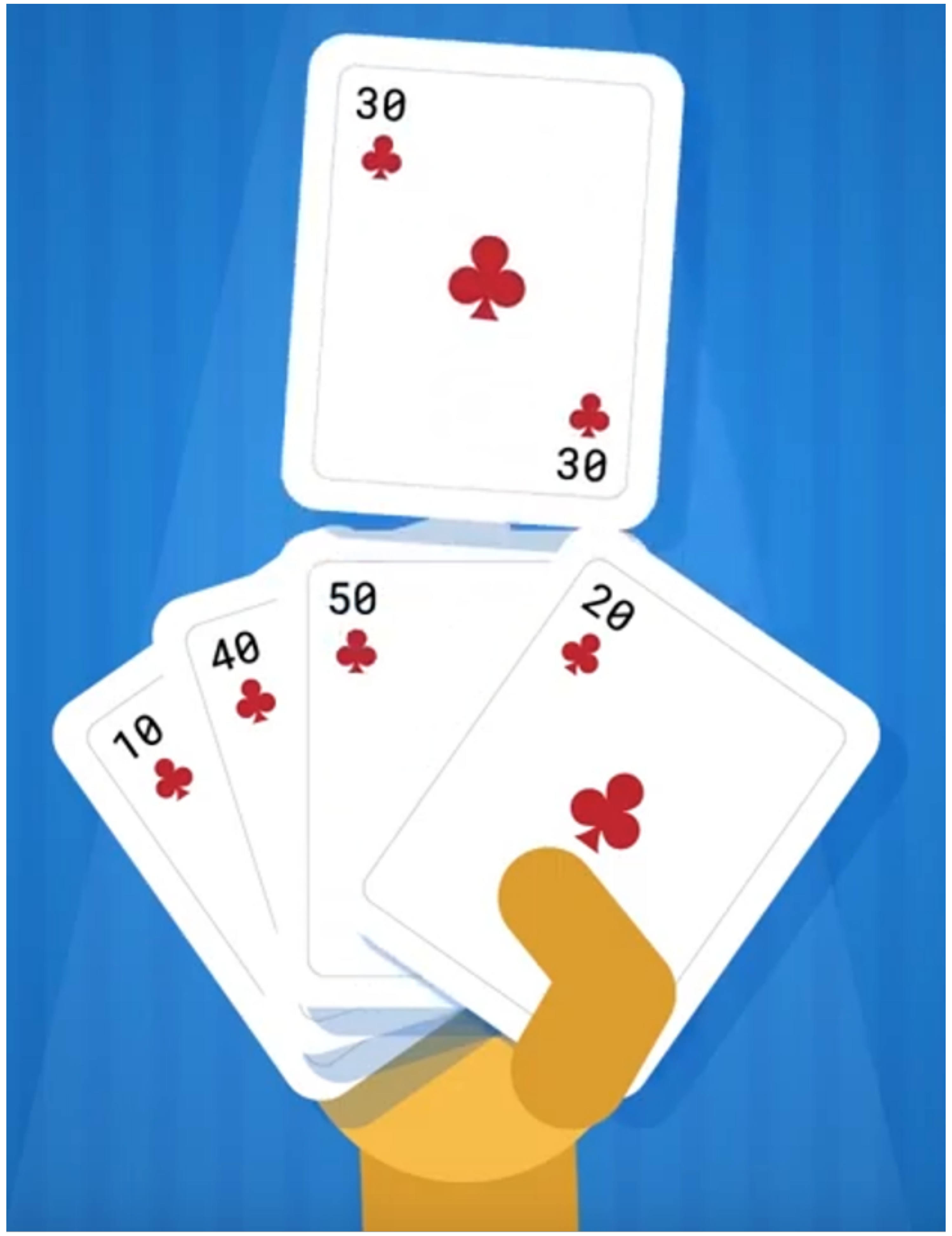
Insertion Sort

General Info

- Comparison Based Sort
- ***Principle:*** Remove an element from an un- sorted input list and insert in the correct position in an already-sorted

Insertion Sort

How it works?



Insertion Sort

Algorithm

Algorithm 1 InsertionSort(A)

```
for j ← 2 to N do
    key ← A[j]
    i ← j − 1
    while i > 0 AND A[i] > key do
        A[i + 1] ← A[i]
        i ← i − 1
    end while
    A[i + 1] ← key
end for
```

- Working of the Algorithm
- Input: Set of numbers $\{n_1, n_2, \ldots n_n\}$
- Output $\{n_1', n_2', \ldots n_n'\}$ in the sorted Sequence.
- Make a sublist, insert one element to the list in the sorted order

Insertion Sort

Proof of correctness

- We can prove using Loop- Invarients, Proof by induction and Proof by Contradiction.
- ***Loop Invarient Method:*** A loop invariant is a statement that is true across multiple iterations of a loop.
- Consider A [i…to …. j] is a sublist that starts from *'i'* and extends till *'j'*

Theorem 1: *Insertion Sort correctly sorts input list A*

- Proof: The first invariant, Inv1, that we will use is that at the start of each for loop iteration A[1..j – 1] is a sorted permutation of the original A[1..j – 1].
- Inv1 holds at the start because A[1..1] is sorted obviously.
- Inv1 holds for each iteration, we must reason about the execution within the loop, till the last statement
- Introduce another Inv2 concerning the While loop, which states that at the start of the while loop body A[i..j] are each greater than or equal to key.

Insertion Sort

Loop Invariant Method - Contnd

- Inv2 is true upon initialization (at the first iteration of the inner loop) because $i = j - 1$ and $A[i] > key$ by explicit testing and $A[j] = key$.

- The inner loop maintains the invariant because the statement $A[i + 1] \leftarrow A[i]$ moves a value in A that is known to be greater than key into A[i+1] which also held a value that was at least equal to key.

- The inner loop (the while loop) does not destroy data in A because the first iteration copies A[j] into key.

- As long as key is restored to A we maintain the invariant that A[1..j] contains the first j elements of the original list.

Insertion Sort

Loop Invariant Method - Contnd

- When the inner loop terminates, we know that
  - A[1..i] is sorted and is at most equal to key (which is true by default if i = 0 and true because A[1..i] is sorted and A[i] ≤ key if i > 0);
  - A[i+1..j] is sorted and at least equal to key because the loop invariant held before i was decremented and that invariant was A[i..j] ≥ key;
  - A[i+1] = A[i+2] if the loop is executed at least once and A[i+1] = key if the loop did not execute at all.
- With these observations, we know that A[i + 1] ← k ey does not destroy any data and gives us A[1..j], which is a sorted permutation of the original j elements of A.
- Inv1 is maintained after a loop iteration, also when terminates.
- At termination, j = N + 1 and so A[1..N] is sorted.

Source: http://courses.ece.ubc.ca/320/notes/InsertionSort.pdf

Insertion Sort

Proof by Induction

- General Information of the proof by induction:
- ***Initialization:*** The loop invariant is satisfied at the beginning of the for loop.
- ***Maintenance:*** If the loop invariant is true before the ith iteration, then the loop invariant will be true before the i + 1st iteration.
- ***Termination:*** When the loop terminates, the invariant gives us a useful property that helps show that the algorithm is correct.

Insertion Sort

Proof by Induction

- Lema: At the start of each iteration of the for loop, the subarray A[1..j – 1] consists of the elements originally in A[1..j – 1], but in sorted order.
- ***Initialization:*** Before the first iteration (which is when j = 2), the subarray [1..j – 1] is just the first element of the array, A[1]. This subarray is sorted, and consists of the elements that were originally in A[1..1].
- ***Maintenance:*** Suppose A[1..j – 1] is sorted. Informally, the body of the for loop works by moving A[j – 1], A[j – 2], A[j – 3] and so on by one position to the right until it finds the proper position for A[j] (lines 4-7), at which point it inserts the value of A[j] (line 8). The subarray A[1..j] then consists of the elements originally in A[1..j], but in sorted order. Incrementing j for the next iteration of the for loop then preserves the loop invariant.
- ***Termination:*** The condition causing the for loop to terminate is that j > n. Because each loop iteration increases j by 1, we must have j = n + 1 at that time. By the initialization and maintenance steps, we have shown that the subarray A[1..n + 1 – 1] = A[1..n] consists of the elements originally in A[1..n], but in sorted order.

Source: https://web.stanford.edu/class/archive/cs/cs161/cs161.1168/lecture1.pdf

Extra Reading

Learn at your convenience

- Source Code: Click here
- Data Visualization for Insertion Sort
- Insertion Sort - Loop Invariant method proof
- Insertion Sort - Induction method proof
- Sample Bubble Sort Proof