8. Sorting I

Simple Sorting

8.1 Simple Sorting

Selection Sort, Insertion Sort, Bubblesort [Ottman/Widmayer, Kap. 2.1, Cormen et al, Kap. 2.1, 2.2, Exercise 2.2-2, Problem 2-2

Problem

Input: An array $A = (A[1], ..., A[n])$ with length n.
Output: a permutation A' of A, that is sorted: $A'[i] \leq A'[j]$ for all $1 \leq i \leq j \leq n$.

Algorithm: `IsSorted`(A)

Input: Array $A = (A[1], ..., A[n])$ with length n.
Output: Boolean decision "sorted" or "not sorted"

```
for i ← 1 to n − 1 do
    if A[i] > A[i + 1] then
        return "not sorted";
return "sorted";
```

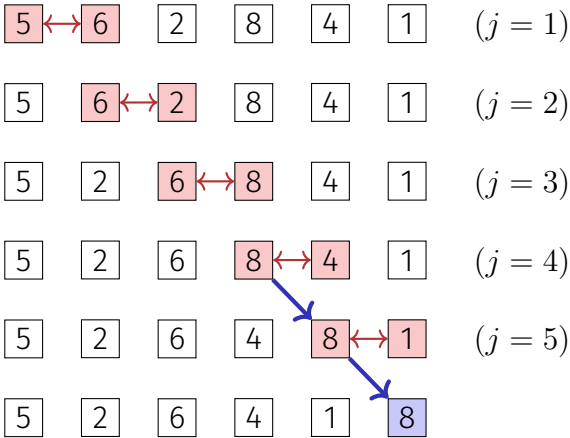
Observation

IsSorted(A):"not sorted", if $A[i] > A[i+1]$ for any i.

$\Rightarrow$ idea:

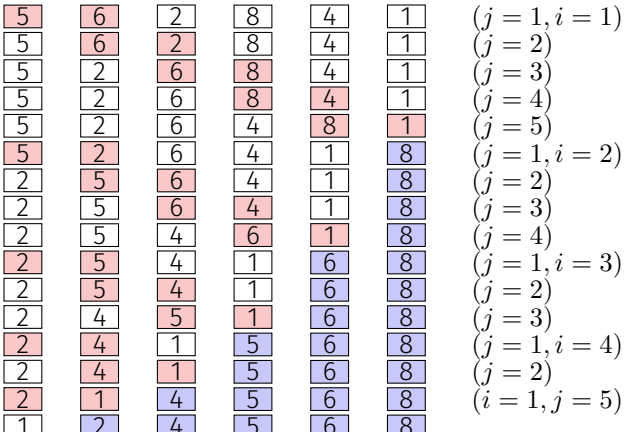
for $j \leftarrow 1$ **to** $n-1$ **do**
    if $A[j] > A[j+1]$ **then**
        swap($A[j], A[j+1]$);

Give it a try



- Not sorted! ☹.
- But the greatest element moves to the right
  $\Rightarrow$ new idea! ☺

Try it out



- Apply the procedure iteratively.
- For $A[1, \ldots, n]$, then $A[1, \ldots, n-1]$, then $A[1, \ldots, n-2]$, etc.

Algorithm: `Bubblesort`

Input: Array $A = (A[1], \ldots, A[n])$, $n \geq 0$.
Output: Sorted Array A
for $i \leftarrow 1$ **to** $n - 1$ **do**
    for $j \leftarrow 1$ **to** $n - i$ **do**
        if $A[j] > A[j + 1]$ **then**
            swap$(A[j], A[j + 1])$;

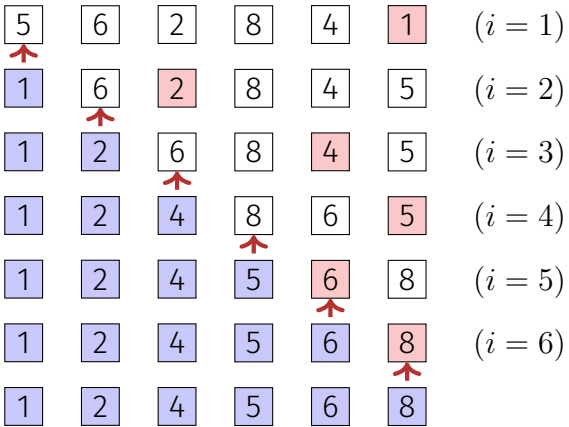
Analysis

Number key comparisons $\sum_{i=1}^{n-1}(n-i) = \frac{n(n-1)}{2} = \Theta(n^2)$.
Number swaps in the worst case: $\Theta(n^2)$

What is the worst case?

If A is sorted in decreasing order.

Selection Sort



- Selection of the smallest element by search in the unsorted part $A[i..n]$ of the array.
- Swap the smallest element with the first element of the unsorted part.
- Unsorted part decreases in size by one element ($i \to i + 1$). Repeat until all is sorted. ($i = n$)

Algorithm: Selection Sort

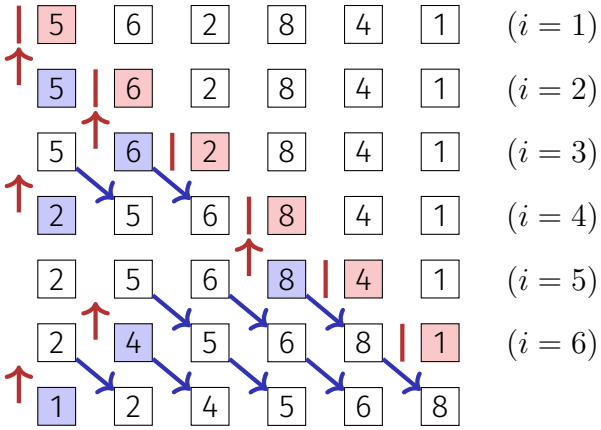
Input: Array $A = (A[1], \ldots, A[n])$, $n \geq 0$.
Output: Sorted Array A

```
for i ← 1 to n − 1 do
    p ← i
    for j ← i + 1 to n do
        if A[j] < A[p] then
            p ← j;
    swap(A[i], A[p])
```

Analysis

Number comparisons in worst case: $\Theta(n^2)$.
Number swaps in the worst case: $n - 1 = \Theta(n)$

Insertion Sort



- Iterative procedure: $i = 1...n$
- Determine insertion position for element i.
- Insert element i array block movement potentially required

Insertion Sort

What is the disadvantage of this algorithm compared to sorting by selection?

Many element movements in the worst case.

What is the advantage of this algorithm compared to selection sort?

The search domain (insertion interval) is already sorted. Consequently: binary search possible.

Algorithm: Insertion Sort

Input: Array $A = (A[1], \ldots, A[n])$, $n \geq 0$.
Output: Sorted Array A

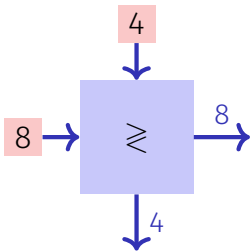
```
for i ← 2 to n do
    x ← A[i]
    p ← BinarySearch(A, 1, i − 1, x); // Smallest p ∈ [1, i] with A[p] ≥ x
    for j ← i − 1 downto p do
        A[j + 1] ← A[j]
    A[p] ← x
```

Analysis

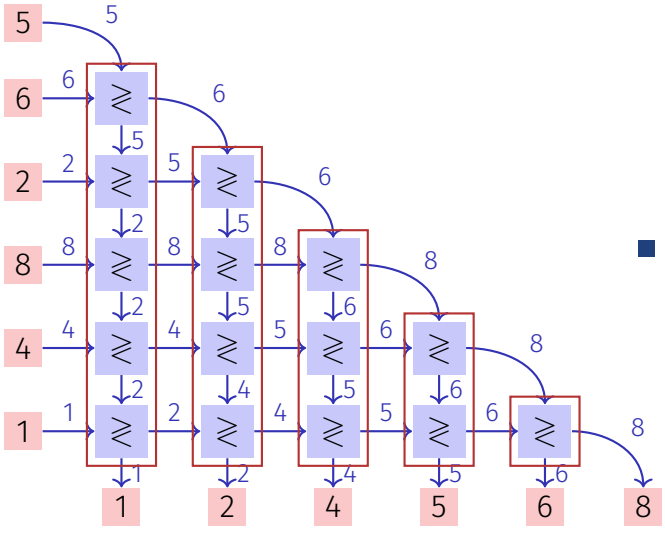
Number comparisons in the worst case:
$\sum_{k=1}^{n-1} a \cdot \log k = a \log((n-1)!) \in \Theta(n \log n)$.
Number swaps in the worst case $\sum_{k=2}^{n}(k-1) \in \Theta(n^2)$

Different point of view

Sorting node:

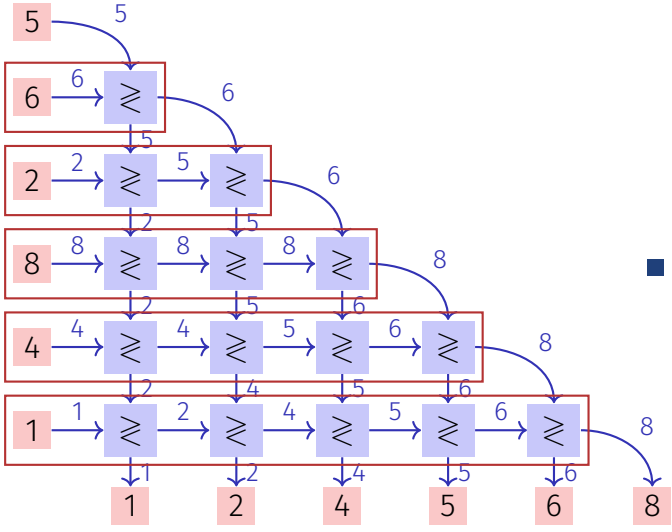


Different point of view



- Like selection sort [and like Bubblesort]

Different point of view



- Like insertion sort

Conclusion

In a certain sense, Selection Sort, Bubble Sort and Insertion Sort provide the same kind of sort strategy. Will be made more precise. [8]

[8] In the part about parallel sorting networks. For the sequential code of course the observations as described above still hold.

Shellsort (Donald Shell 1959)

Intuition: moving elements far apart takes many steps in the naive methods from abobe

Insertion sort on subsequences of the form $(A_{k \cdot i})$ $(i \in \mathbb{N})$ with decreasing distances k. Last considered distance must be $k = 1$.

Worst-case performance critically depends on the chosen subsequences

- Original concept with sequence $1, 2, 4, 8, ..., 2^k$. Running time: $\mathcal{O}(n^2)$
- Sequence $1, 3, 7, 15, ..., 2^k - 1$ (Hibbard 1963). $\mathcal{O}(n^{3/2})$
- Sequence $1, 2, 3, 4, 6, 8, ..., 2^p 3^q$ (Pratt 1971). $\mathcal{O}(n \log^2 n)$

Shellsort

9	8	7	6	5	4	3	2	1	0	
2	8	7	6	5	4	3	9	1	0	insertion sort, $k = 7$
2	1	7	6	5	4	3	9	8	0	
2	1	0	6	5	4	3	9	8	7	
2	1	0	3	5	4	6	9	8	7	insertion sort, $k = 3$
2	1	0	3	5	4	6	9	8	7	
2	1	0	3	5	4	6	9	8	7	
0	1	2	3	4	5	6	7	8	9	insertion sort, $k = 1$