ITMO UNIVERSITY

# How to Win Coding Competitions: Secrets of Champions

## Week 3: Sorting and Search Algorithms
## Lecture 3: When to sort? Optimality of sorted sequences

Maxim Buzdalov
Saint Petersburg 2016

## Recall: Scalar product maximization problem

- Given two sequences $A = [A_1, \dots A_N]$ and $B = [B_1, \dots B_N]$
- Find permutations $P$ and $Q$ such that $\sum_{i=1}^{N} A_{P_i} \cdot B_{Q_i}$ is maximum possible

| | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 7 | 1 | 4 | 6 | 8 | 2 | 9 | 3 | 1 | 4 | 3 | 5 | 9 | 8 | 2 |
| 3 | 1 | 2 | 8 | 6 | 5 | 4 | 7 | 4 | 9 | 4 | 5 | 1 | 3 | 8 |

Recall: Scalar product maximization problem

- Given two sequences $A = [A_1, \ldots A_N]$ and $B = [B_1, \ldots B_N]$
- Find permutations $P$ and $Q$ such that $\sum_{i=1}^{N} A_{P_i} \cdot B_{Q_i}$ is maximum possible
- Solution: sort both sequences

| | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 1 | 2 | 2 | 3 | 3 | 4 | 4 | 5 | 6 | 7 | 8 | 8 | 9 | 9 |
| 1 | 1 | 2 | 3 | 3 | 4 | 4 | 4 | 5 | 5 | 6 | 7 | 8 | 8 | 9 |

Recall: Scalar product maximization problem

- Given two sequences $A = [A_1, \dots A_N]$ and $B = [B_1, \dots B_N]$
- Find permutations $P$ and $Q$ such that $\sum_{i=1}^{N} A_{P_i} \cdot B_{Q_i}$ is maximum possible
- Solution: sort both sequences

| | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 1 | 2 | 2 | 3 | 3 | 4 | 4 | 5 | 6 | 7 | 8 | 8 | 9 | 9 |
| 1 | 1 | 2 | 3 | 3 | 4 | 4 | 4 | 5 | 5 | 6 | 7 | 8 | 8 | 9 |

How to prove things like this one?

Recall: Insertion sort

**procedure** $\textsc{InsertionSort}(A, \leq)$
  **for** $i$ **from** $1$ **to** $|A|$ **by** $1$ **do**
    $k \leftarrow i$
    **while** $(k > 1)$ **and not** $(A[k-1] \leq A[k])$ **do**
      $A[k-1] \Leftrightarrow A[k]$
      $k \leftarrow k - 1$
    **end while**
  **end for**
**end procedure**

Recall: Insertion sort

```
procedure InsertionSort(A, ≤)
    for i from 1 to |A| by 1 do
        k ← i
        while (k > 1) and not (A[k − 1] ≤ A[k]) do
            A[k − 1] ⇔ A[k]
            k ← k − 1
        end while
    end for
end procedure
```

The only used way to move the elements is swapping neighbors!

Recall: Insertion sort

**procedure** $\textsc{InsertionSort}(A, \leq)$
    **for** $i$ **from** $1$ **to** $|A|$ **by** $1$ **do**
        $k \leftarrow i$
        **while** $(k > 1)$ **and not** $(A[k-1] \leq A[k])$ **do**
            $A[k-1] \Leftrightarrow A[k]$
            $k \leftarrow k - 1$
        **end while**
    **end for**
**end procedure**

The only used way to move the elements is swapping neighbors!

We will use this feature to simplify the optimality proof

- Assume we have proven that, for any array $A$ and some relation $\leq$, whenever we don't have $A_i \leq A_{i+1}$ for some index $i$, we do not make our solution worse if we swap $A_i$ and $A_{i+1}$

- Assume we have proven that, for any array $A$ and some relation $\leq$, whenever we don't have $A_i \leq A_{i+1}$ for some index $i$, we do not make our solution worse if we swap $A_i$ and $A_{i+1}$
- Then an array, sorted by the $\leq$ relation, yields the best solution, because:

- Assume we have proven that, for any array $A$ and some relation $\leq$, whenever we don't have $A_i \leq A_{i+1}$ for some index $i$, we do not make our solution worse if we swap $A_i$ and $A_{i+1}$
- Then an array, sorted by the $\leq$ relation, yields the best solution, because:
  - Every array can be sorted only by swapping the neighbors which are out of order (this is how insertion sort works)

- Assume we have proven that, for any array $A$ and some relation $\leq$, whenever we don't have $A_i \leq A_{i+1}$ for some index $i$, we do not make our solution worse if we swap $A_i$ and $A_{i+1}$
- Then an array, sorted by the $\leq$ relation, yields the best solution, because:
  - Every array can be sorted only by swapping the neighbors which are out of order (this is how insertion sort works)
  - Every such swap does not make the solution worse (this is our assumption)

- Assume we have proven that, for any array $A$ and some relation $\leq$, whenever we don't have $A_i \leq A_{i+1}$ for some index $i$, we do not make our solution worse if we swap $A_i$ and $A_{i+1}$
- Then an array, sorted by the $\leq$ relation, yields the best solution, because:
  - Every array can be sorted only by swapping the neighbors which are out of order (this is how insertion sort works)
  - Every such swap does not make the solution worse (this is our assumption)
  - Thus, a sorted array is never worse than any other array

- Assume we have proven that, for any array $A$ and some relation $\leq$, whenever we don't have $A_i \leq A_{i+1}$ for some index $i$, we do not make our solution worse if we swap $A_i$ and $A_{i+1}$
- Then an array, sorted by the $\leq$ relation, yields the best solution, because:
  - Every array can be sorted only by swapping the neighbors which are out of order (this is how insertion sort works)
  - Every such swap does not make the solution worse (this is our assumption)
  - Thus, a sorted array is never worse than any other array
- Note that we don't have to use insertion sort
  - Any sorting algorithm, which uses the $\leq$ relation, will do!

- Assume we have proven that, for any array $A$ and some relation $\leq$, whenever we don't have $A_i \leq A_{i+1}$ for some index $i$, we do not make our solution worse if we swap $A_i$ and $A_{i+1}$
- Then an array, sorted by the $\leq$ relation, yields the best solution, because:
  - Every array can be sorted only by swapping the neighbors which are out of order (this is how insertion sort works)
  - Every such swap does not make the solution worse (this is our assumption)
  - Thus, a sorted array is never worse than any other array
- Note that we don't have to use insertion sort
  - Any sorting algorithm, which uses the $\leq$ relation, will do!
  - We may use a more efficient algorithm

Problem: Scalar product maximization

- Given two sequences $A = [A_1, \ldots A_N]$ and $B = [B_1, \ldots B_N]$
- Find permutations $P$ and $Q$ such that $\sum_{i=1}^{N} A_{P_i} \cdot B_{Q_i}$ is maximum possible
- Solution: sort both sequences

Problem: Scalar product maximization

- Given two sequences $A = [A_1, \dots A_N]$ and $B = [B_1, \dots B_N]$
- Find permutations $P$ and $Q$ such that $\sum_{i=1}^{N} A_{P_i} \cdot B_{Q_i}$ is maximum possible
- Solution: sort both sequences

Proof:

- We can do whatever we want with one of the sequences!
  - Let's sort $A$ – we still free to do everything using permutations of $B$

Problem: Scalar product maximization

- Given two sequences $A = [A_1, \dots A_N]$ and $B = [B_1, \dots B_N]$
- Find permutations $P$ and $Q$ such that $\sum_{i=1}^{N} A_{P_i} \cdot B_{Q_i}$ is maximum possible
- Solution: sort both sequences

Proof:

- We can do whatever we want with one of the sequences!
  - Let's sort $A$ – we still free to do everything using permutations of $B$
- Why should $B$ be sorted?
  - Assume we have $B_i > B_{i+1}$. In the same time, $A_i \leq A_{i+1}$

Problem: Scalar product maximization

- Given two sequences $A = [A_1, \ldots A_N]$ and $B = [B_1, \ldots B_N]$
- Find permutations $P$ and $Q$ such that $\sum_{i=1}^{N} A_{P_i} \cdot B_{Q_i}$ is maximum possible
- Solution: sort both sequences

Proof:

- We can do whatever we want with one of the sequences!
  - Let's sort $A$ – we still free to do everything using permutations of $B$
- Why should $B$ be sorted?
  - Assume we have $B_i > B_{i+1}$. In the same time, $A_i \leq A_{i+1}$
  - If we swap $B_i$ and $B_{i+1}$, nothing will change, except for two addends:
    - $A_i B_i + A_{i+1} B_{i+1} \Rightarrow A_i B_{i+1} + A_{i+1} B_i$

Problem: Scalar product maximization

- Given two sequences $A = [A_1, \dots A_N]$ and $B = [B_1, \dots B_N]$
- Find permutations $P$ and $Q$ such that $\sum_{i=1}^{N} A_{P_i} \cdot B_{Q_i}$ is maximum possible
- Solution: sort both sequences

Proof:

- We can do whatever we want with one of the sequences!
  - Let's sort $A$ – we still free to do everything using permutations of $B$
- Why should $B$ be sorted?
  - Assume we have $B_i > B_{i+1}$. In the same time, $A_i \leq A_{i+1}$
  - If we swap $B_i$ and $B_{i+1}$, nothing will change, except for two addends:
    - $A_i B_i + A_{i+1} B_{i+1} \Rightarrow A_i B_{i+1} + A_{i+1} B_i$
    - $A_i B_{i+1} + A_{i+1} B_i - A_i B_i - A_{i+1} B_{i+1} = (A_{i+1} - A_i)(B_i - B_{i+1}) \geq 0$

Problem: Scalar product maximization

- Given two sequences $A = [A_1, \ldots A_N]$ and $B = [B_1, \ldots B_N]$
- Find permutations $P$ and $Q$ such that $\sum_{i=1}^{N} A_{P_i} \cdot B_{Q_i}$ is maximum possible
- Solution: sort both sequences

Proof:

- We can do whatever we want with one of the sequences!
  - Let's sort $A$ – we still free to do everything using permutations of $B$
- Why should $B$ be sorted?
  - Assume we have $B_i > B_{i+1}$. In the same time, $A_i \leq A_{i+1}$
  - If we swap $B_i$ and $B_{i+1}$, nothing will change, except for two addends:
    - $A_i B_i + A_{i+1} B_{i+1} \Rightarrow A_i B_{i+1} + A_{i+1} B_i$
    - $A_i B_{i+1} + A_{i+1} B_i - A_i B_i - A_{i+1} B_{i+1} = (A_{i+1} - A_i)(B_i - B_{i+1}) \geq 0$
  - If we swap $B_i$ and $B_{i+1}$, the solution will not get worse! □