Subject of the lesson: Sorting and Binary Search
Relevant STL Functions

• Sort: sort a range of values
• Searching a sorted range of values:
  – Binary search: Return true/false if the value exists
  – `lower_bound(x)`: Return a pointer to the first element which is greater or equal to x.
  – `upper_bound(x)`: Return a pointer to the first element which is strictly greater than x.
• For example, when searching for 3 in a sorted array:
  – 1 2 3 3 3 3 4 6
  – 1 2 4 6
When do we need to use binary search?

- If
  - We look for a minimal (or maximal) value $k$ for which some property holds.
  - Given such $k$, it is relatively easy to check if the property holds.
  - The property is **monotone**. If the property holds for $k$, then it holds for all $j > k$.
  - Then we can use binary search to find the minimal $k$ for which the property holds.
A construction company considers the placement of gas stations along a given road system. Construction is very expensive, and therefore the company wants to construct the least amount of gas stations that will cover the road.

The only information we have is a “test function”:
- Given k, the function returns true iff we can cover the road system using k gas stations or less.

Clearly, if we can construct k gas stations to cover the road, then we can also do it for every j>k.

Therefore, we can use binary search to find the minimal possible value of k.

In the problem we will encounter, you should look for a relatively simple “test function”.

(Hypothetical) Example
low = minimum possible index
high = maximum possible index
while (low < high)
    mid = (low + high)/2
    if check(mid)
        high = mid
    else
        low = mid + 1
return low
Sorting

• Sorting is a useful tool for a wide range of problems.
• After sorting it is easier to:
  – Quickly search for elements
  – Go over the elements in order
  – Find identical elements
• Example from the first lesson - Lawn Mower problem (4954)
• In every problem where the input is not ordered, try to think if ordering the elements might help after some manipulations.
Sorting – Example

• In a given street there are \( n \) houses, and each house has a number \( n_i \).

• The post office wants to re-number the houses. They suggest the tenants to choose a number \( k \), so the new number of house \( i \) will be \( k \cdot i \).

• We need to find the \( k \) such which minimizes the amount of houses that need to change their number.
Solution

• Define \( m_i = \frac{n_i}{i} \)

• For a given \( k \), every house \( i \) for which \( k = m_i \) will not change its number (since \( k \cdot i = m_i \cdot i = n_i \) )

• Therefore, the most frequent value in the new array \((m_1, \ldots, m_n)\) will give us the \( k \) we are looking for.

• This value is easy to find after sorting the array.
Convex Functions

• Equivalent definitions
  – Line segment between any two point on the graph of the function lies above or on the graph
  – Epigraph (the set of points lying on or above its graph) is a convex set
  – Second derivative greater or equal to 0 for entire domain

• Examples:
  – Quadratic function $x^2$
  – Exponential function $e^x$

• If $f(x), g(x)$ are convex functions, then $h(x) = \max\{f(x), g(x)\}$ is also a convex function.
Ternary Search

- Search for a maximum (or minimum) of a convex function.
- Logarithmic time complexity
Ternary Search – Pseudocode

left = minimum possible value
right = maximum possible value
while (right - left > epsilon)
    mid_left = (2*left + right)/3
    mid_right = (left + 2*right)/3
    if check(mid_left) < check(mid_right)
        right = mid_right
    else
        left = mid_left
return left