

# Problem 1618: Maximum Font to Fit a Sentence in a Screen

## Problem Information

**Difficulty:** Medium

**Acceptance Rate:** 61.65%

**Paid Only:** Yes

**Tags:** Array, String, Binary Search, Interactive

## Problem Description

You are given a string `text`. We want to display `text` on a screen of width `w` and height `h`. You can choose any font size from array `fonts`, which contains the available font sizes **in ascending order**.

You can use the `FontInfo` interface to get the width and height of any character at any available font size.

The `FontInfo` interface is defined as such:

```
interface FontInfo { // Returns the width of character ch on the screen using font size fontSize.
// O(1) per call public int getWidth(int fontSize, char ch); // Returns the height of any character
on the screen using font size fontSize. // O(1) per call public int getHeight(int fontSize); }
```

The calculated width of `text` for some `fontSize` is the **sum** of every `getWidth(fontSize, text[i])` call for each `0 ≤ i < text.length` (**0-indexed**). The calculated height of `text` for some `fontSize` is `getHeight(fontSize)`. Note that `text` is displayed on a **single line**.

It is guaranteed that `FontInfo` will return the same value if you call `getHeight` or `getWidth` with the same parameters.

It is also guaranteed that for any font size `fontSize` and any character `ch`:

```
* `getHeight(fontSize) ≤ getHeight(fontSize+1)` * `getWidth(fontSize, ch) ≤
getWidth(fontSize+1, ch)`
```

Return \_the maximum font size you can use to display\_`text` \_on the screen\_. If `text` cannot fit on the display with any font size, return `-1`.

**Example 1:**

**Input:** `text = "helloworld", w = 80, h = 20, fonts = [6,8,10,12,14,16,18,24,36]` **Output:** `6`

**Example 2:**

**Input:** `text = "leetcode", w = 1000, h = 50, fonts = [1,2,4]` **Output:** `4`

**Example 3:**

**Input:** `text = "easyquestion", w = 100, h = 100, fonts = [10,15,20,25]` **Output:** `-1`

**Constraints:**

`1 <= text.length <= 50000` \* `text` contains only lowercase English letters. `1 <= w <= 107`  
`1 <= h <= 104` \* `1 <= fonts.length <= 105` \* `1 <= fonts[i] <= 105` \* `fonts` is sorted in ascending order and does not contain duplicates.

## Code Snippets

**C++:**

```
/**
 * // This is the FontInfo's API interface.
 * // You should not implement it, or speculate about its implementation
 * class FontInfo {
 * public:
 * // Return the width of char ch when fontSize is used.
 * int getWidth(int fontSize, char ch);
 *
 * // Return Height of any char when fontSize is used.
 * int getHeight(int fontSize)
 * };
 */
class Solution {
public:
int maxFont(string text, int w, int h, vector<int>& fonts, FontInfo fontInfo)
```

```
{  
  
}  
};
```

## Java:

```
/**  
 * // This is the FontInfo's API interface.  
 * // You should not implement it, or speculate about its implementation  
 * interface FontInfo {  
 * // Return the width of char ch when fontSize is used.  
 * public int getWidth(int fontSize, char ch) {}  
 * // Return Height of any char when fontSize is used.  
 * public int getHeight(int fontSize)  
 * }  
 */  
  
class Solution {  
    public int maxFont(String text, int w, int h, int[] fonts, FontInfo fontInfo)  
    {  
  
    }  
}
```

## Python3:

```
# """  
# This is FontInfo's API interface.  
# You should not implement it, or speculate about its implementation  
# """  
#class FontInfo(object):  
#    # Return the width of char ch when fontSize is used.  
#    # def getWidth(self, fontSize, ch):  
#    # """  
#    # :type fontSize: int  
#    # :type ch: char  
#    # :rtype int  
#    # """  
#  
#    # def getHeight(self, fontSize):  
#    # """  
#    # :type fontSize: int  
#    # :rtype int
```

```
# ""  
class Solution:  
    def maxFont(self, text: str, w: int, h: int, fonts: List[int], fontInfo :  
        'FontInfo') -> int:
```