



Practical 4: Computational Complexity



Dr. Anna Kalenkova

s1: n elements

4	6	3	4	5	5	4	5	6	7
-			-))	-)		-

s2: m elements



Prefixes of s2:

4

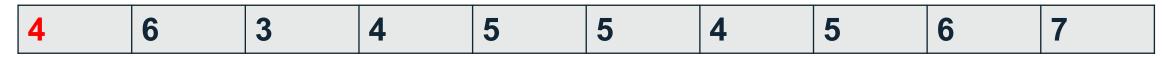
4 5

456

4568



s1: n elements



s2: m elements



Prefixes of s2:

4

4 5

456

4568

45689

Output (indexes in s1):



n elements s1:



s2: m elements



Prefixes of s2:

4 5

456

4568

45689

Output (indexes in s1):



s1: n elements



s2: m elements

4 5	6	8	9	
-----	---	---	---	--

Prefixes of s2: Output (indexes in s1):

4 5

4 5 6 4 5 6 8



s1: n elements

4	6	3	4	5	5	4	5	6	7
•			•			•			•

s2: m elements

```
4 5 6 8 9
```

Prefixes of s2: Output (indexes in s1):

4 4 5 4 5 6

4 5 6 8



Algorithm

```
vector<int> result;
                               O(m) iterations
for(size_t i = 1; i <= s2.size(); i++) {</pre>
                           O(i*n) the cost of searching prefix of length i in n
   size_t found = s1.find(s2.substr(0, i));
   if (found != string::npos) {
       result.push back(found);
   } else {
       result.push_back(-1);
                   Overall complexity: O(n+2n+3n+...+i*n+...+m*n) = O(m^2n)
return result;
```

Algorithm

Can the algorithm be optimized?

- 1. Find operations that are performed several times.
- 2. Try to avoid them. Hint: s1.find(s2, index) can find substrings starting from index.
- 3. If prefix was not found (-1 as an output), the larger prefix will not be found either.
- 4. What will be the time complexity of the optimized algorithm?



Practical 4

- 1. Update function :
 vector<int> Finder::findSubstrings(string s1, string s2)
- 2. Submit Finder.h and Finder.cpp to Practical 4 in Gradescope, (available till Apr 02 at 11:59PM)

