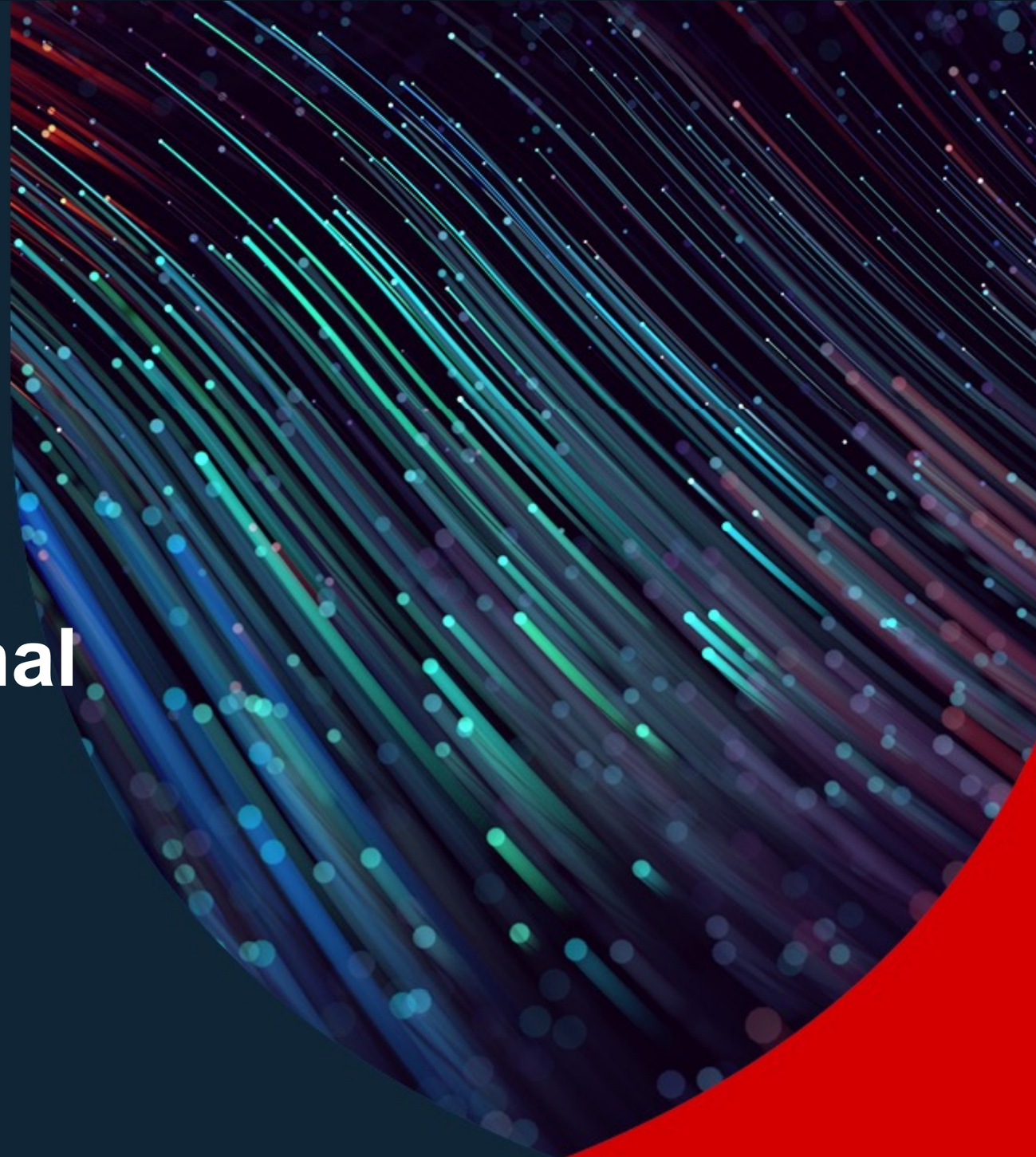
make history.

THE UNIVERSITY of ADELAIDE



Practical 4: Computational Complexity

Dr. Anna Kalenkova

Find prefixes of s2 in s1

s1: n elements

s2: m elements

Prefixes of s2:

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

THE UNIVERSITY of ADELAIDE

Find prefixes of s2 in s1

s1: n elements

s2: m elements

Prefixes of s2:

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

Output (indexes in s1):

0

THE UNIVERSITY of ADELAIDE

Find prefixes of s2 in s1

s1: n elements

s2: m elements

Prefixes of s2:

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

Output (indexes in s1):

0
3

THE UNIVERSITY of ADELAIDE

Find prefixes of s2 in s1

s1: n elements

s2: m elements

Prefixes of s2:

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

Output (indexes in s1):

0
3
6

THE UNIVERSITY of ADELAIDE

Find prefixes of s2 in s1

s1: n elements

s2: m elements

Prefixes of s2:

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

Output (indexes in s1):

0
3
6
-1
-1

THE UNIVERSITY of ADELAIDE

Algorithm

```
vector<int> result;

for(size_t i = 1; i <= s2.size(); i++) {
    size_t found = s1.find(s2.substr(0, i));

    if (found != string::npos) {
        result.push_back(found);
    } else {
        result.push_back(-1);
    }
}

return result;
```

O(m) iterations

O(i*n) the cost of searching prefix of length i in n

Overall complexity: $O(n+2n+3n+\ldots+ i*n+ \ldots+ m*n) = O(m^2n)$

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.