[Cheesiest Pizza]

[Houdaifa Bouamine]

Problem Statement

Sefar is a famous restaurant known for its giant pizzas, and you are one of their loyal clients. Recently, the chef at Sefar made a huge pizza, and being a loyal customer, the chef offered you a special opportunity: you can choose any `i x j` rectangular part of the pizza. Since you absolutely love cheesy pizza, your goal is to find the `i x j` slice of the pizza with the highest amount of cheese. The pizza is represented as a rectangular grid of size `n x m`, where each cell is either a pizza surface (`@`) or cheese (`#`).

Your task is to extract the slice of size `i x j` that contains the most cheese.

Input

- The first line contains `T`, the number of test cases.
- For each test case:
  - The first line contains two integers `n` and `m`, representing the size of the full pizza.
  - The second line contains two integers `i` and `j`, representing the size of the pizza slice you need to extract.
  - The next `n` lines each contain `m` characters, representing the pizza grid.

Output

For each test case, you should find the slice of size `i x j` that contains the maximum amount of cheese.

compute the flag by concatenating the grid of the cheesiest pizza slice you find (concatenate the grid line by line without a separator), then generate a SHA256 hash of the concatenated string. This will be the flag for the test case.

Examples

Example 1:

Input:

```
1
5 5
2 3
@#@#@
#@#@#
@#@##
#@#@#
@#@#@
```

Output:

```
#@#
@##
```

Flag:

`3243e24479e503d4a0229b1db4b21325b30c85b2e86c3f940a4188b1f11d4f13`

Explanation:

- For the given pizza grid, you need to extract a `2 x 3` slice.
- The slice with the most cheese is the one starting at position `(1,2)`, which contains 4 of `#` characters (cheese).
- the concatenated string is “#@#@##” (“#@#” + “@##”).
- Compute the SHA256 hash of "#@#@##" to get the flag:

```
sha256("#@#@##") =>
3243e24479e503d4a0229b1db4b21325b30c85b2e86c3f940a4188b1f11d4f13
```

Example 2:

Input:

```
2
5 6
1 3
@#@#@@
@@@@@@
#@@@@@
@@@#@#
@@@@@@
15 18
2 2
@@@@@@#@@@@@@@@@@@@
##@@@@@@#@@@@@@@@@@
#@@@@@@@#@@@@@@@@@#
@@@@@@#@@@@@@@@@@@#
#####@@@#@@@@@@@@@@
@@@@@@#@@@#@#@@@@@#
@#@@@@@#@@@@#@@@#@@
@@#@@@@@@#@@@@@#@@#
#@@@@@@@@@@@@@@@@@#
@#@#@###@@@@@@@@@@@
@@@@@@@@@@@@#@@@@@@
@@#@#@@@@@#@@@@#@@#@
@@@@@@##@@@@@@@#@@@
@@#@##@@@@@@#@@@@@##
#@@@@@@@#@@@@@@@@@@
```

Output:

```
#@#  // testcase 1's output
##   // testcase 2's output
#@
```

Flag:

```
c975a8b700d7e6bbf0bfcd1dbc645faaea1ed8a84198fdb14315f900175d6805
```

Calculating the flag: Concatenate all the strings from all test cases, then pass it to sha256:

```
("#@#") + ("##" + "#@") = "#@####@"
sha256("#@####@") =>
c975a8b700d7e6bbf0bfcd1dbc645faaea1ed8a84198fdb14315f900175d6805
```