~~CSC363~~ CSC369 Tutorial 6

hope tomorrow's paul is feeling better! (paul, having a headache and sick, on feb 23, while preparing slides)

Paul "sushi_enjoyer" Zhang

University of Amogus

February 23, 2021

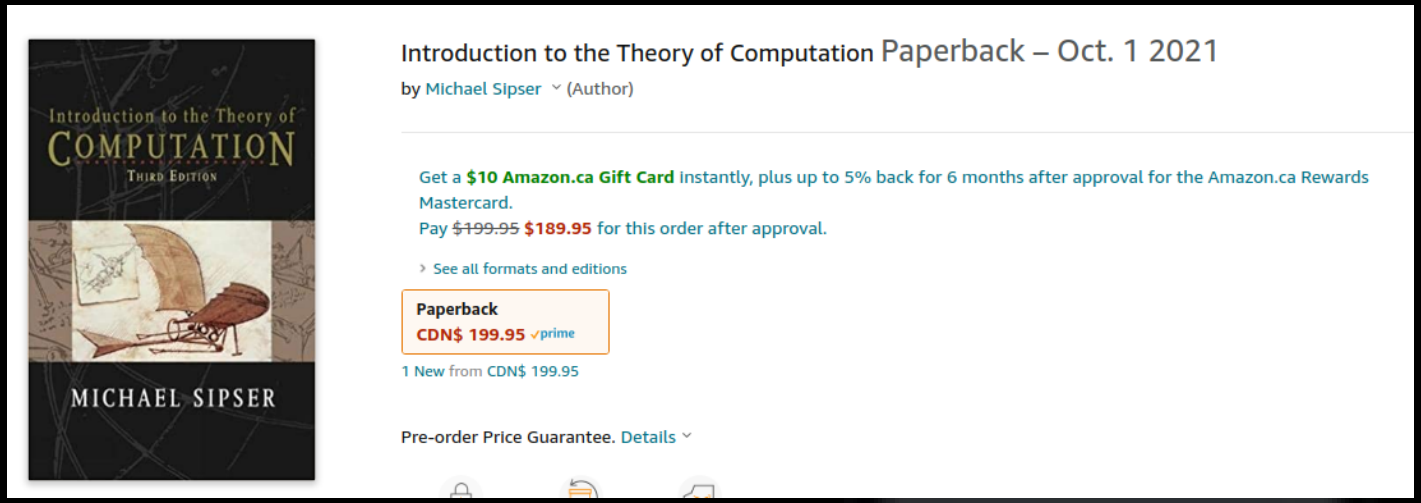
Learning objectives this tutorial

By the end of this tutorial, you should...

- Once again have CSIS (or whatever law enforcement for copyright violations in your jurisdiction) come to your house due to [REDACTED].
- Have recalled the formal definition of a Turing machine, and have built one in Minecraft (or any other Turing-complete game)
- Understand the formal definition of a multi-tape Turing machine, and have built one again in Minecraft (or any other Turing-complete game).
- Understand the formal definition of a nondeterministic Turing machine.
- Be infuriated by any further mentioning of computability, because we are completely done with that chapter of our lives.[1]

[1] If you hate computability, don't open assignment 3! (too bad you're forced to ;-;)

More unaffordable textbooks! yay :/



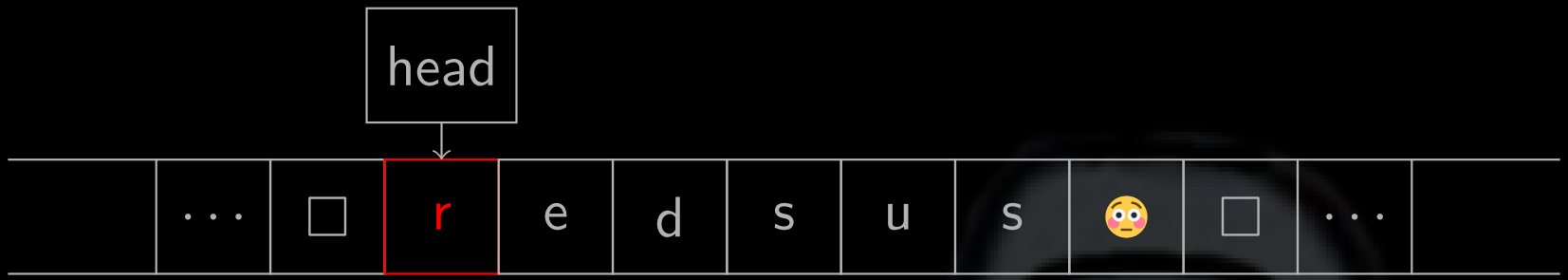
i'm broke from spending money on sushi. i can't afford this textbook.

Again, it's worth a read! you'll go over cool stuff like

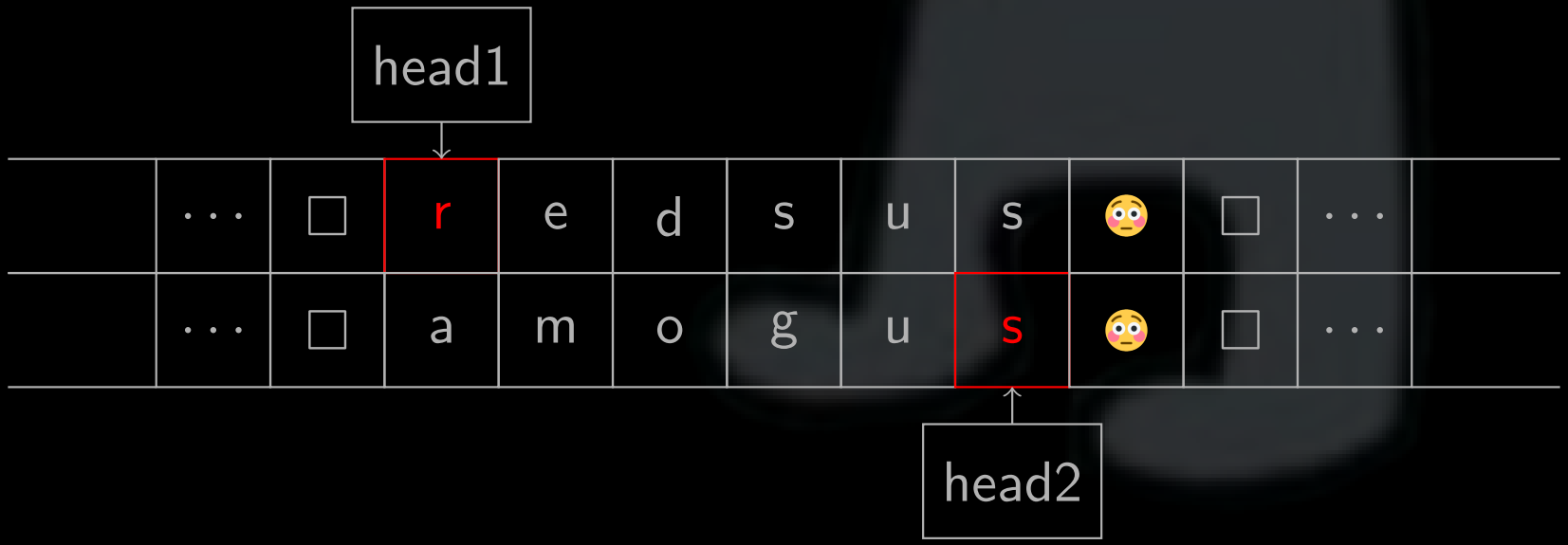
- What really is a Turing machine? (3.1)
- What are multi-tape and nondeterministic Turing machines? Why are they equivalent to Turing machines? (3.2)
- What does $f(n) = O(g(n))$ mean again? What does P mean? (7.1, 7.2)
- Why didn't we use this textbook earlier? (69.420)

Turing machines are back! :D

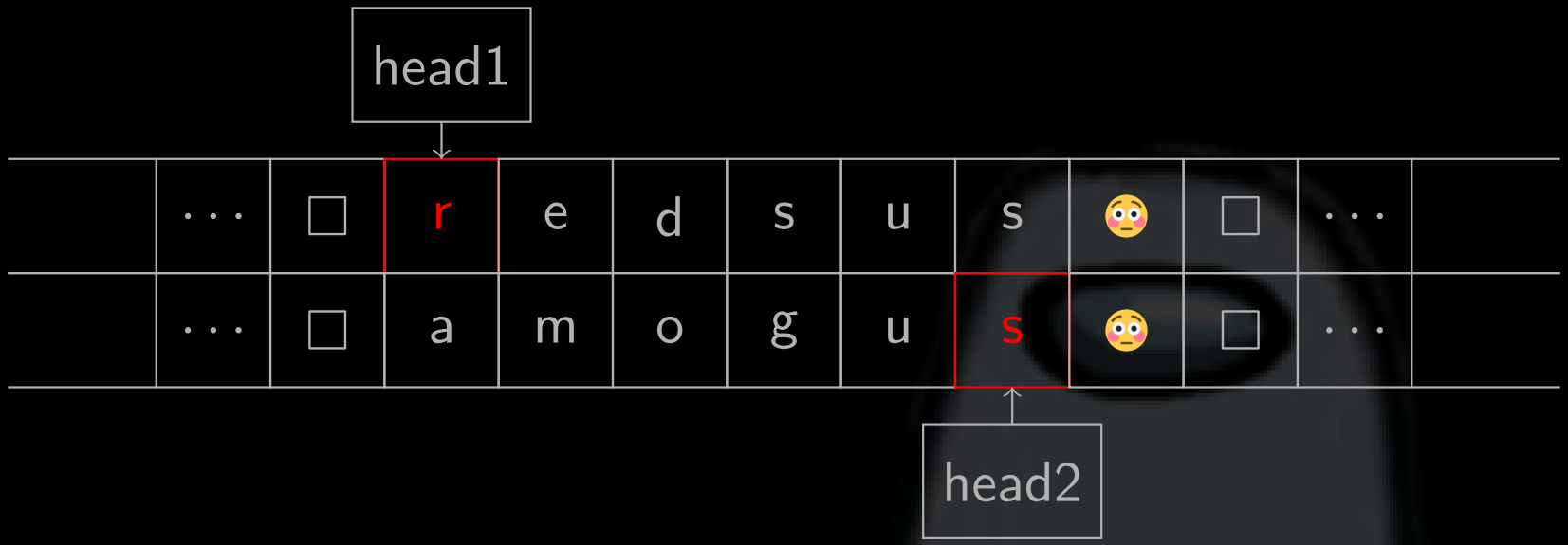
Hope you remember what a Turing machine is!



This Turing machine is sus 😳 it only has one tape. what if we had multiple tapes?



Turing machines are back! :D



This 2-tape Turing machine will read in 2 symbols at once (as a 2-tuple), consult the current state, then output the following:

1. Next state to transition towards;
2. Symbol to write back via head1, and direction to move head1;
3. Symbol to write back via head2, and direction to move head2.

Greek letters are back! D:

Do you like formal definitions? If you don't, too bad :(
Task: Recall that a Turing machine (the mathematical object) is an 8-tuple $(Q, \Sigma, \Gamma, \delta, q_0, q_{\text{accept}}, q_{\text{reject}}, \square)$ where

- Q is the (finite) set of states;
- Σ is the input alphabet not containing $\square$;
- Γ is the tape alphabet, and $\square \in \Gamma$ and $\Sigma \subseteq \Gamma$;
- $\delta : Q \times \Gamma \to Q \times \Gamma \times \{L, R\}$ is the transition function;
- $q_0 \in Q$ is the starting state;
- q_{accept} is the accept state;
- q_{reject} is the reject state (and $q_{\text{reject}} \neq q_{\text{accept}}$);
- $\square$ is the blank symbol.

Familiarize yourself with this definition. Ask any questions you have!

Greek letters are back! D:

Do you like formal definitions? If you don't, too bad :(
Task: The definition of a k-tape Turing machine (the mathematical object) is an 8-tuple $(Q, \Sigma, \Gamma, \delta, q_0, q_{\text{accept}}, q_{\text{reject}}, \square)$ where

- Q is the (finite) set of states;
- Σ is the input alphabet not containing $\square$;
- Γ is the tape alphabet, and $\square \in \Gamma$ and $\Sigma \subseteq \Gamma$;
- $\delta : Q \times \Gamma^k \to Q \times \Gamma^k \times \{L, R, -\}^k$ is the transition function;[2]
- $q_0 \in Q$ is the starting state;
- q_{accept} is the accept state;
- q_{reject} is the reject state (and $q_{\text{reject}} \neq q_{\text{accept}}$);
- $\square$ is the blank symbol.

Compare this with the previous definition. Ask any questions you have!

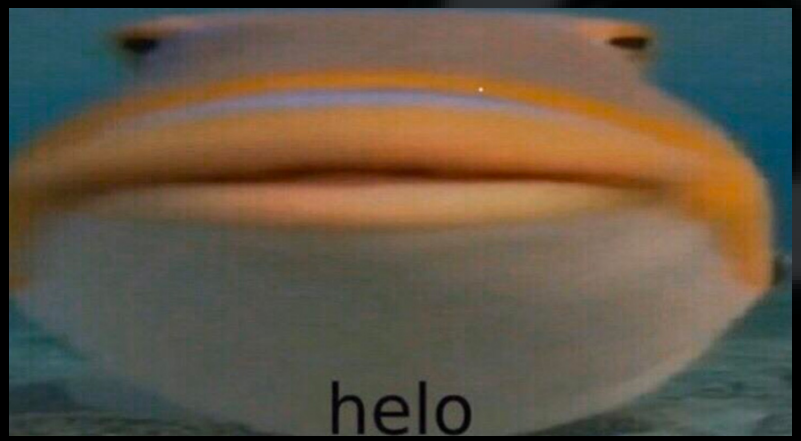
[2] $-$ is used to not move the head at all. This additional feature is not in the textbook definition of a k-tape Turing machine, but I added it as a user story in my spare time. Don't worry, the computational power is still equivalent.

Greek letters are back! D:

But how does the k-tape Turing machine execute?

Here are the differences between a k-tape Turing machine and a regular Turing machine:

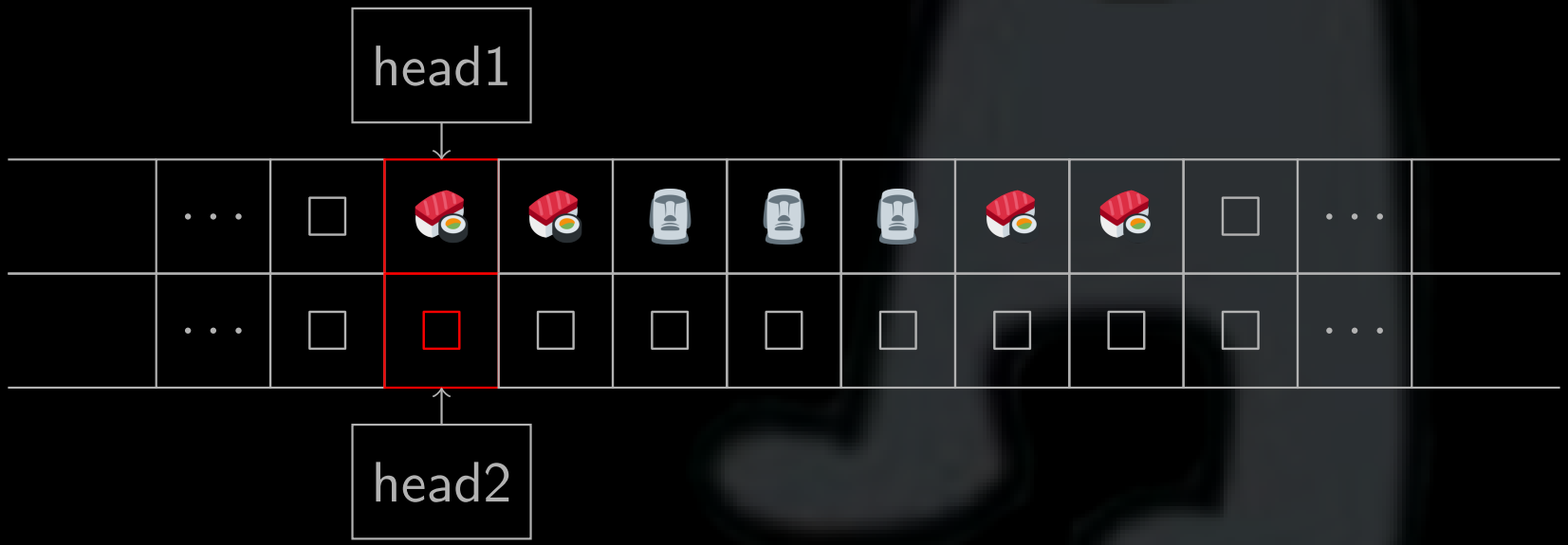
1. In a regular Turing machine, the string input is written on the (only) tape at the start of execution. In a k-tape Turing machine, the string input is always written on the first tape.
2. In a k-tape Turing machine, we read in k symbols at once, and we output k symbols and k directions from $\{L, R, -\}$, to specify the symbol to write and direction to move for each of the k tapes (where $-$ doesn't move it at all).
3. `helo_fish.jpg` likes multi-tape Turing machines, unlike regular Turing machines.



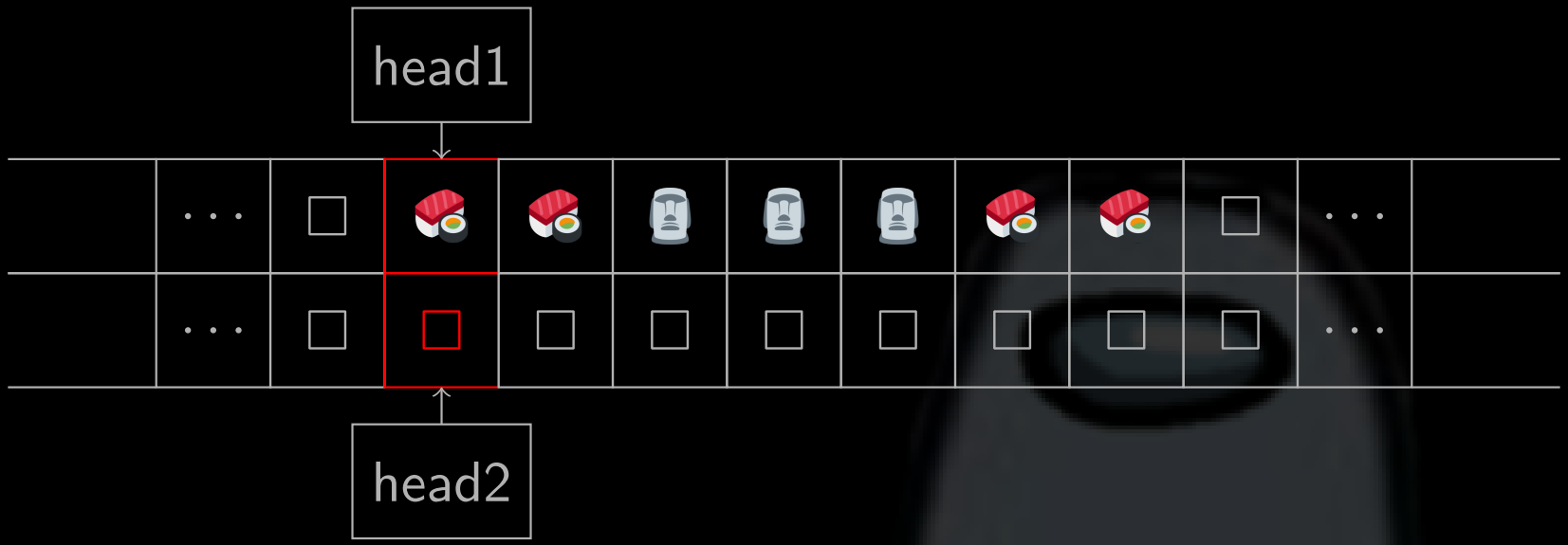
Let's try an example!

I want to construct a 2-tape Turing machine will check if a given binary string is a palindrome. For example, 🍣🗿🍣🍣🍣🍣🗿🍣 is accepted, while 🍣🗿🗿🗿🍣🗿🍣🗿 is not.

Say we are given input 🍣🍣🗿🗿🗿🍣🍣. The Turing machine would start out like so:

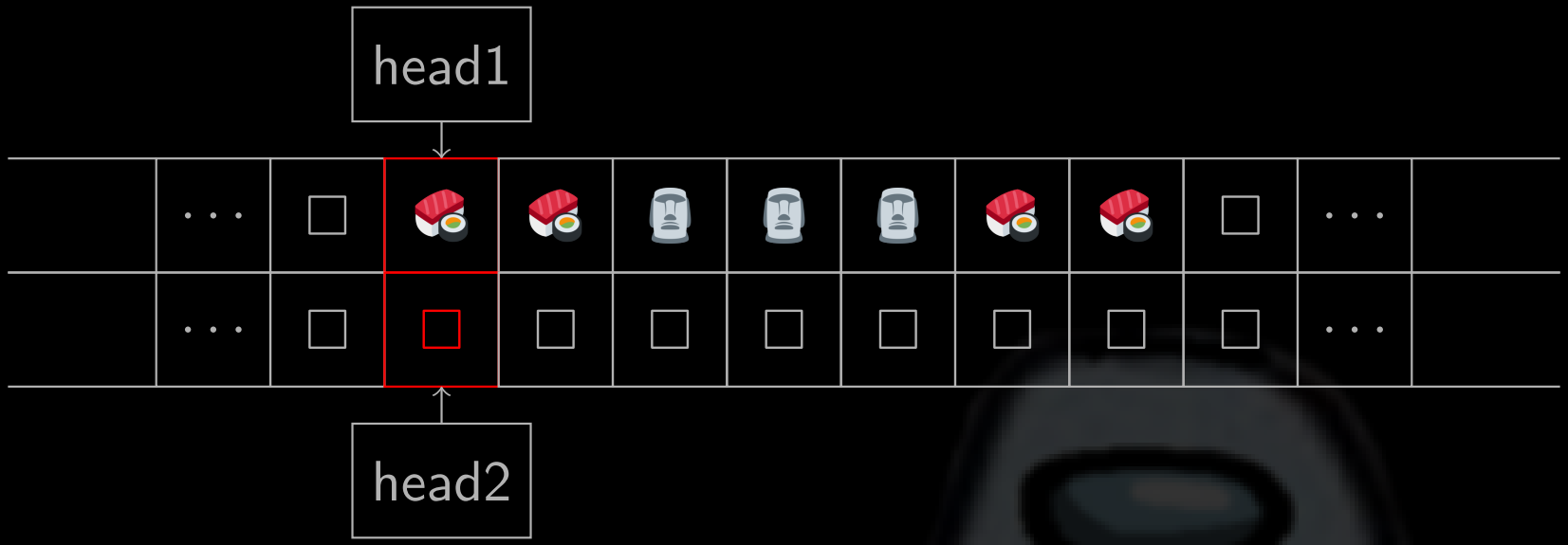


Let's try an example!

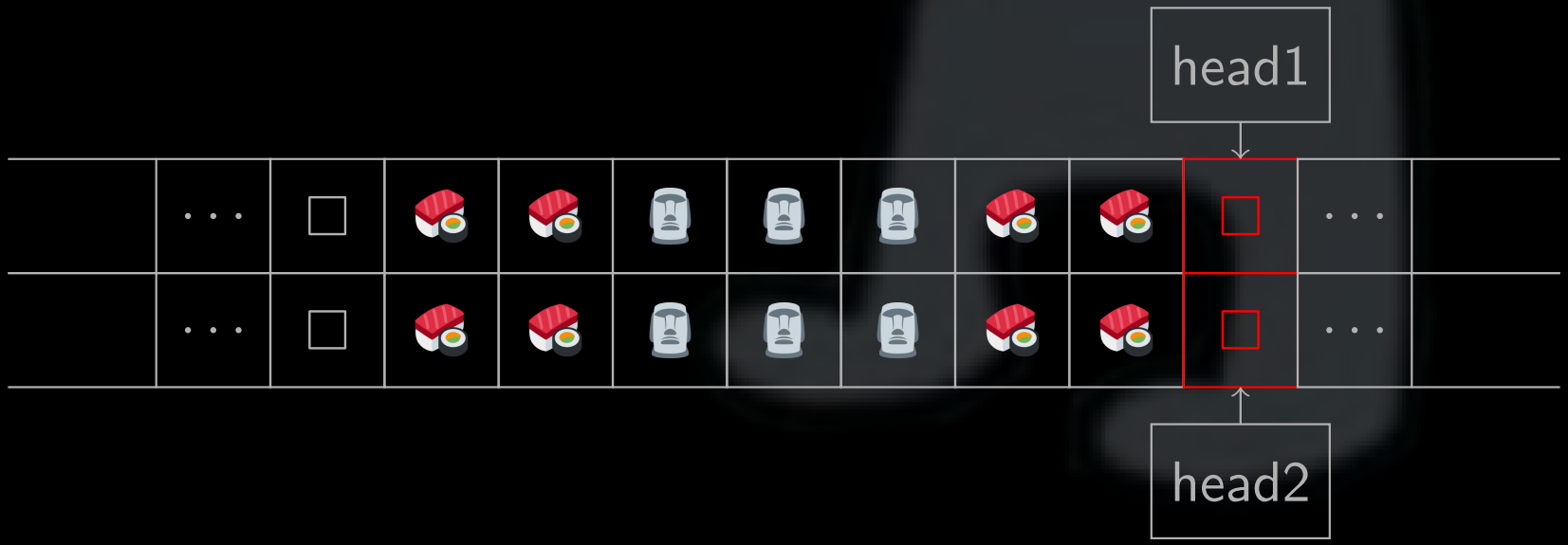


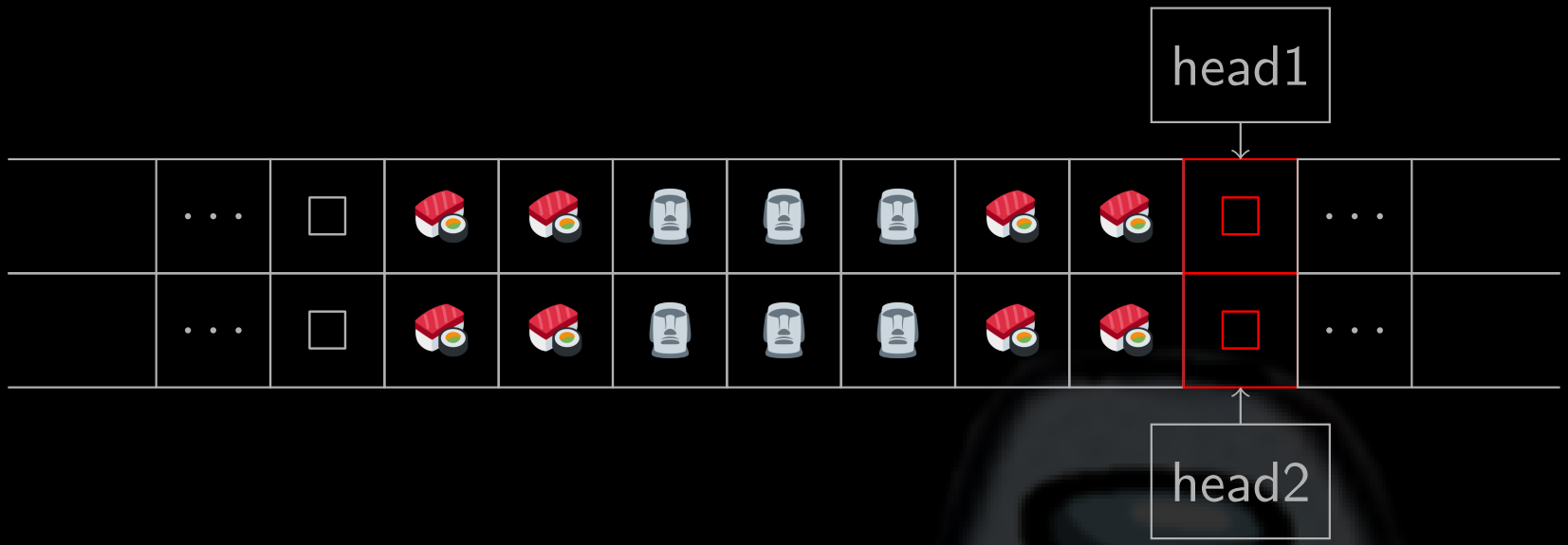
We'll program the Turing machine to do the following, at a high level:

1. Copy the string on the first tape to the second tape.
2. Move head1 to the beginning of the first tape, and head2 to the last character on the second tape.
3. Compare the symbols at head1 and head2. If they are not equal, reject. Else move head1 to the right and head2 to the left.

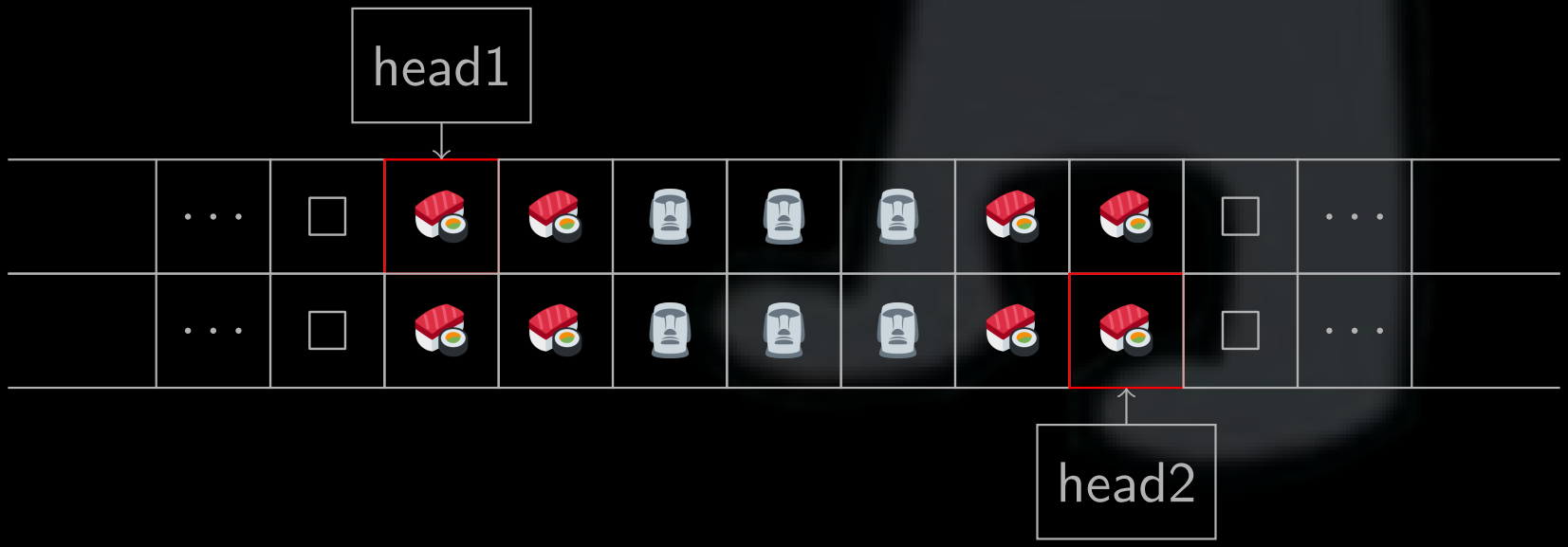


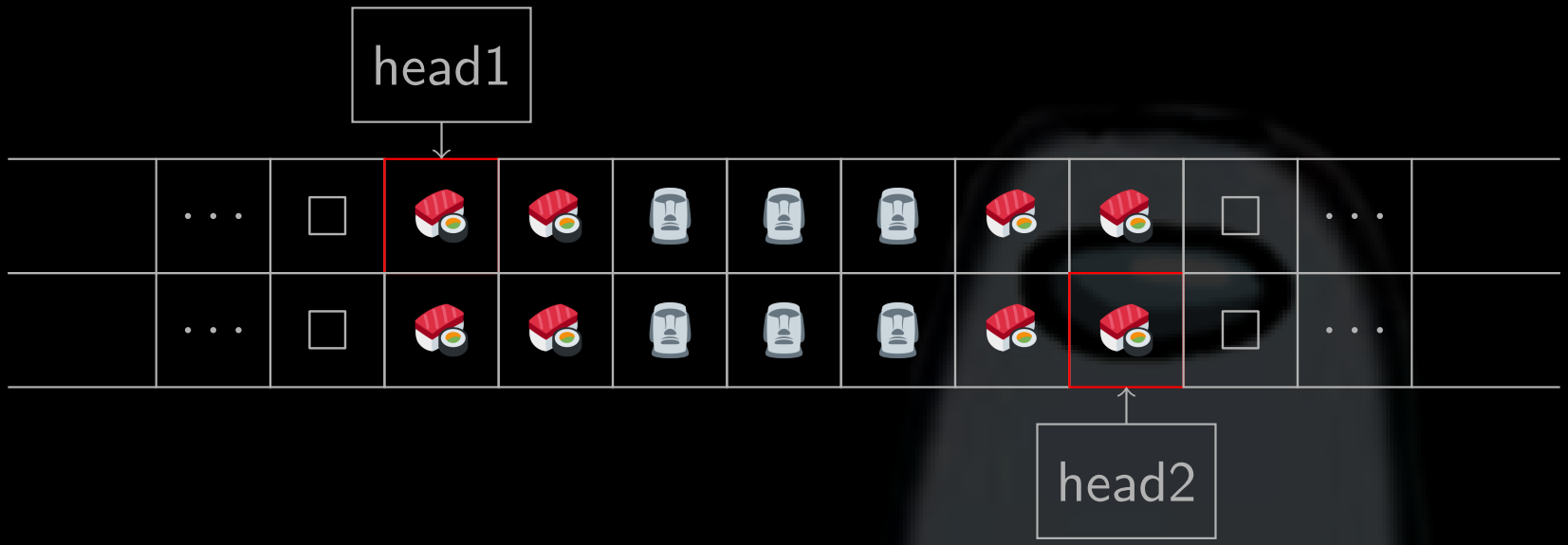
1. Copy the string on the first tape to the second tape.





2. Move head1 to the beginning of the first tape, and head2 to the last character on the second tape.





Compare the symbols at head1 and head2. If they are not equal, reject. Else move head1 to the right and head2 to the left.

I want to construct a 2-tape Turing machine will check if a given binary string is a palindrome. For example,  is accepted, while  is not.

We'll program the Turing machine to do the following, at a high level:

1. Copy the string on the first tape to the second tape.
2. Move head1 to the beginning of the first tape, and head2 to the last character on the second tape.
3. Compare the symbols at head1 and head2. If they are not equal, reject. Else move head1 to the right and head2 to the left.

Task: Formally write this Turing machine down. Remember, a k-tape Turing machine (the mathematical object) is an 8-tuple $(Q, \Sigma, \Gamma, \delta, q_0, q_{\text{accept}}, q_{\text{reject}}, \square)$ where

- $\delta : Q \times \Gamma^k \to Q \times \Gamma^k \times \{L, R, -\}^k$ is the transition function;[3]

[3] $-$ is used to not move the head at all. This additional feature is not in the textbook definition of a k-tape Turing machine, but I added it as a user story in my spare time. Don't worry, the computational power is still equivalent.

They're actually the same!

Not really. But they're equivalent.

They're actually the same!

If you're done, convince yourself this problem is much harder using a regular Turing machine.