Example: $A = \{0^k1^k \mid k \geq 0\}$

- Let’s count the number of steps in the algorithm discussed before
- Check if the input is $0...01...1$

This takes $O(n)$

- Move back: $O(n)$
- Cross off each 0 and 1: $O(n)$

How many such crosses: $n/2$

$n/2 \times O(n) = O(n^2)$
Example: $A = \left\{0^k1^k \mid k \geq 0\right\}$

- Accept or not?
  - $O(n)$ to go through from beginning to end
- Total:
  $$O(n) + O(n^2) + O(n) = O(n^2)$$
Def: Time complexity class I

Definition:

\[ \text{TIME}(t(n)) \equiv \{ L \mid L \text{ a language decided by an } O(t(n)) \text{ TM} \} \]

We have

\[ \{0^k1^k \mid k \geq 0\} \in \text{TIME}(n^2) \]

Can we make it faster?
New Algorithm for $A = \{0^k1^k \mid k \geq 0\}$

- The procedure: cross off every other 0 and 1
  
  0000011111
  0011
  01
  $\epsilon$

  key: length of the string left must be always even

- A failed algorithm
  
  000011
  001

- Algorithm
New Algorithm for $A = \{0^k1^k \mid k \geq 0\}$ II

1. check 0...0 1...1
2. repeat if not empty
   total # 0 & 1: odd $\Rightarrow$ reject
   cross off every other 0 and 1
3. no 0 & 1 remain, accept

- If 13 "0" $\Rightarrow$ 6 "0" $\Rightarrow$ 3 "0" $\Rightarrow$ 1 "0"
  $1 + \log_2 n$ iterations
- Each iter: $O(n)$ operations
- Total cost: $O(n \log n)$
New Algorithm for $A = \{0^k1^k \mid k \geq 0\}$ III

- Therefore
  
  $$\{0^k1^k \mid k \geq 0\} \in \text{TIME}(n \log n)$$

- Can we do better? no

- Any language decided in $o(n \log n)$ on a single-tape TM $\Rightarrow$ regular (not proved here)

- But we know that
  
  $$\{0^k1^k \mid k \geq 0\}$$

  is not regular
Using two-tape TM for \( \{0^k1^k \mid k \geq 0\} \)

- We can have an \( O(n) \) procedure
  1. check 0...0 1...1
  2. copy 0 to 2nd tape
     find the first 1
  3. sequentially cut 1 and 0
     if no “0” reject
  4. if “1” left, reject
     otherwise, accept

- Each step \( O(n) \)
In Chapter 3 we showed that various TMs are equivalent.

For example, single-tape and multi-tape TMs are equivalent.

However, their “time complexity” are different.
Theorem 7.8

$O(t(n))$ multi-tape TM

$\Rightarrow \exists$ equivalent $O(t(n)^2)$ single-tape TM

Idea for the proof: similar to how we proved their equivalence

Show that simulating each step of a multi-tape TM takes $O(t(n))$ on a single-tape TM

Let $k$ be the number of tapes

How did we simulate a multi-tape TM?
Complexity of Multi-tape TM II

CPU

0 1 1 0 ... tape 1

a a c a ... tape 2

0 0 1 1 ... tape 3

CPU

# 0 i 1 ... # a a ... # 0 0 ...
Simulate each step of multi-tape TM: scan to know where heads point to and do the update.

However, we may have to right shift the tape.

So we need to know the tape length. It is

\[ k \times O(t(n)) = O(t(n)) \]

Why \( O(t(n)) \)?

Each tape of multi-tape TM has \( O(t(n)) \) length.

An \( O(t(n)) \) multi-tape TM generates \( O(t(n)) \) contents in \( O(t(n)) \) time (note that \( O(t(n)) \geq n \)).
Thus the cost of simulating each step of multi-tape TM on a single-tape TM is $O(t(n))$.

There are $O(t(n))$ multi-tape TM steps, so the total cost is

$$O(t(n)) \times O(t(n)) = O(t(n)^2)$$
Remember NTM a decider if all branches halt on all inputs

Definition of NTM time complexity $f(n)$:
maximum $\#$ of steps the machine uses on any branch on any input length $n$

Theorem 7.11
For an $t(n) \geq n$, $O(t(n))$ NTM (single tape) \implies \exists a 2^{O(t(n))}$ TM (single tape)

Assume $b$ is the maximal number of branches at each layer
Recall that we use a depth-first way for the simulation.

Total number of nodes in the tree:

\[ O(b^{t(n)}) \]

Depth \( d \) finished, do \( d + 1 \)

Tape 3: all possible paths so far

Tape 2: original input \( w \), simulate one path, 1 step further with all possible branches
Cost of running from root to one node in tape 2: $O(t(n))$
Update of tape 3: $O(b^{t(n)})$
Total cost $\leq O(b^{t(n)})$ for each node
Total time:

$$\# \text{ nodes} \times \text{ cost per node} = O(b^{t(n)}) \times O(b^{t(n)}) = O(b^{2t(n)}) = O(2^{t(n)})$$
This is by a 3-tape TM

To use a single-tape TM to simulate a 3-tape one, we need

\[ (2^{O(t(n))})^2 = 2^{O(t(n))} \]

cost because

\[ (2^{O(t(n))})^2 \leq (2^{ct(n)})^2 = 2^{2ct(n)} = 2^{O(t(n))} \]