CS5371
Theory of Computation

Lecture 17: Complexity II
(Relationship among models)

Objectives

- Complexity relationship among models
  - Single-Tape versus Multi-Tape
  - NTM versus DTM

Single-Tape versus Multi-Tape

Theorem: Let $t(n)$ be a function, and $t(n) \geq n$. Then for every $t(n)$-time TM that works with k tapes, there is an equivalent $O(k^2\, t(n)^2)$-time TM that works with 1 tape.

Proof: Let M be a k-tape TM that runs in $t(n)$ time. We construct a single-tape TM S that runs in $O(k^2\, t(n)^2)$ time.

Single-Tape vs Multi-Tape (2)

Recall that we learnt one way of how S can simulate M (in Lecture 11):

- S uses its single tape to represent the contents of all k tapes in M
- The k tapes are stored consecutively, separated by #
- Positions of tape heads are represented by "marked" symbols

Here, S uses the same way to simulate M

Single-Tape vs Multi-Tape (3)

Recall that to perform a step in M, S will do:

- Scan the tape to collect the characters under each of the tape heads in M
- Scan the tape again, update the symbol under the tape heads of M, and update the positions of the tape heads
- Special case: when a tape head of M moves rightward onto an unread portion, we add a space in the corresponding place in S's tape (by shifting)

Single-Tape vs Multi-Tape (4)

Since M runs in $t(n)$ time, each of its tape head can access only the first $t(n)$ cells. Thus, S will use (and access) only the first $k \times t(n) + k + 1 = O(k\, t(n))$ cells.

We call these $O(k\, t(n))$ cells the active portion of S's tape

Single-Tape vs Multi-Tape (5)

S simulates M for $O(t(n))$ steps, where each step (in the worst case) does the following:

(1) scan the tape first

(2) add a space to each of the k tapes,

(3) update tape heads and its contents

For (1) and (3), S accesses only the active portion of S's tape → $O(k\, t(n))$ time

For (2), S scans the active portion for at most k times, which is $O(k^2\, t(n))$ time

→ In total, it thus takes $O(k^2\, t(n)^2)$ time

Polynomial Time Bounds

If the running time $t(n)$ of a machine M is $O(n^c)$ for some fixed constant $c > 0$, the running time is called polynomial bounded, or we say M runs in polynomial time. This gives the following corollary.

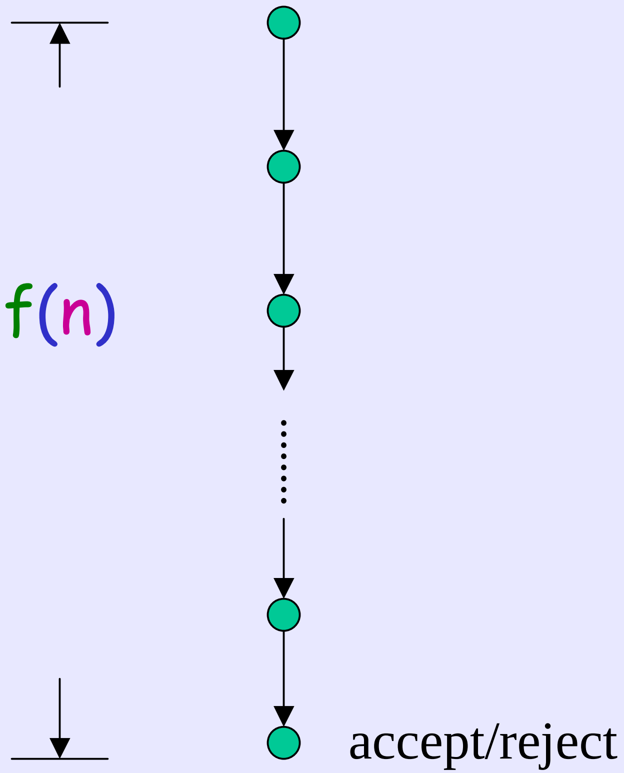
Corollary: For any k-tape TM that runs in polynomial time, it has an equivalent single-tape TM that runs in polynomial time.

NTM decider

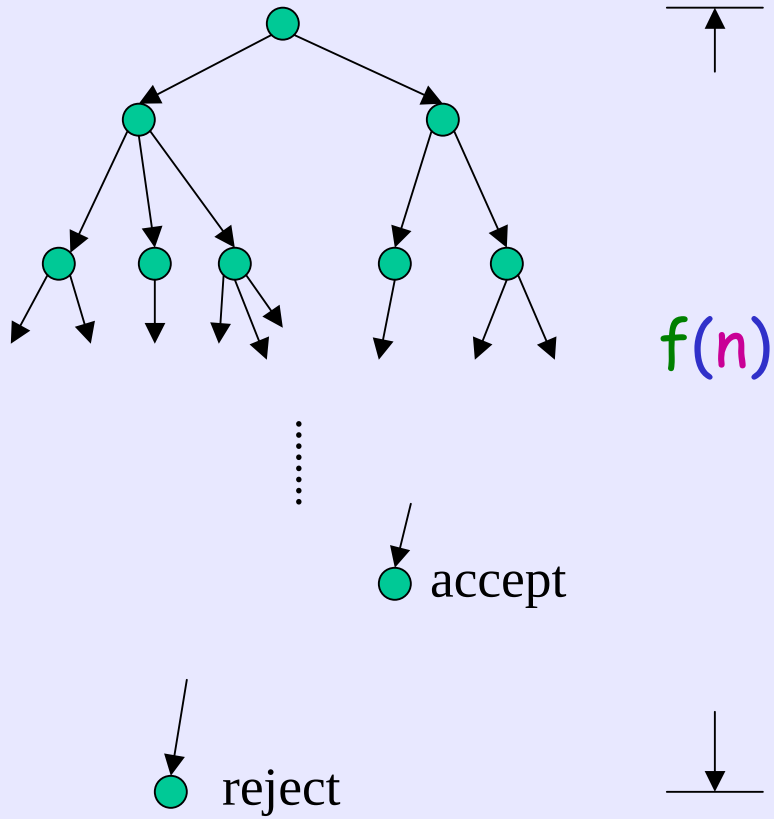
An NTM is a decider if all its computation branches halt on all inputs.

Definition: Let M be an NTM decider. The running time of M is the function $f: N \rightarrow N$, where $f(n)$ is the maximum number of steps that M uses on any branch of its computation on any input of length n

Comparison of Running Times



Deterministic time



Non-deterministic time

DTM versus NTM decider

Theorem: Let $t(n)$ be a function, $t(n) \geq n$. Then every $t(n)$-time single-tape NTM decider has an equivalent $2^{O(t(n))}$-time single-tape DTM

Proof: Let M be a NTM that runs in $t(n)$ time. We construct a DTM D that simulates M by searching M's computation tree, as described in Lecture 11. We now analyze D's simulation.

DTM versus NTM decider (2)

- On an input of length n, every branch of computation of M has at most $t(n)$ steps
- Every node in the computation tree has at most b children, where b is the maximum number of choices in M's transition ➔ number of leaves is at most $O(b^{t(n)})$
- Also, total number of nodes + leaves is at most $O(t(n)\ b^{t(n)})$ (why??)

DTM versus NTM decider (3)

- The simulation proceeds by visiting the nodes (including leaves) in BFS order. Here, when we visit a node v, we always travel starting from the root

  ➔ time to visit v is $O(t(n))$

Thus, the total time for D to simulate M is $O(t(n)^2\, b^{t(n)}) = 2^{O(t(n))}$ (why??)

Next Time

- P and NP
  - Two important classes of problems in time complexity theory