CSC363 Tutorial #8

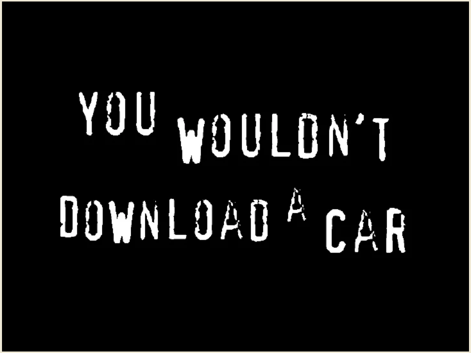
Nondeterministic Turing machines

March 15, 2023

Things covered in this tutorial

- ⋆ What's a nondeterministic Turing machine?
- ⋆ How do nondeterministic Turing machines accept/reject inputs?
- ⋆ How is nondeterminism useful to the concepts we learn in this course?

Copyright infringement



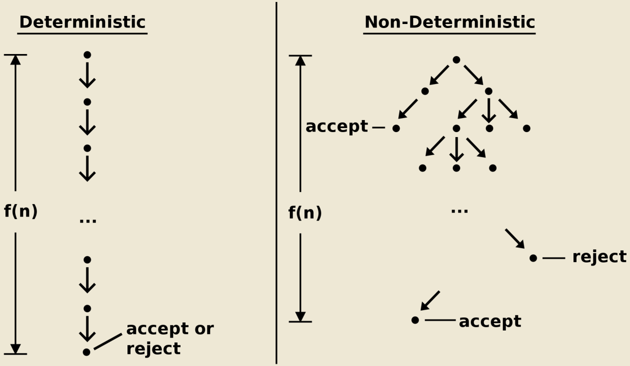
I've stolen most of the images in these slides.

Nondeterminism

(Recall the difference between a DFA and an NFA, if you remember!)

Turing machines are *deterministic*: given an input, they have only one possible sequence of execution.

Nondeterministic Turing Machines (NTMs) have multiple possible sequences of execution.



I stole this image from Wikipedia. Thanks!

(Deterministic) Turing Machines

Task: Fill in . . ..

A Turing Machine is a 7-tuple $(Q, \Sigma, \Gamma, \delta, q_0, q_{\text{accept}}, q_{\text{reject}})$ where:

- ⋆ Q is . . .
- ⋆ Σ is . . .
- ⋆ Γ is . . .
- ⋆ $\delta : \ldots \rightarrow \ldots$ is the transition function.
- ⋆ q_0 is . . .
- ⋆ q_{accept} is . . .
- ⋆ q_{reject} is . . .

(Deterministic) Turing Machines

A Turing Machine is a 7-tuple $(Q, \Sigma, \Gamma, \delta, q_0, q_{\text{accept}}, q_{\text{reject}})$ where:

- $\star$ Q is the set of states.
- $\star$ Σ is the *input alphabet*.
- $\star$ Γ is the *tape alphabet* (and satisfies $\Gamma \subseteq \Sigma$).
- $\star$ $\delta : (Q \times \Gamma) \to (Q \times \Gamma \times \{L, R\})$ is the transition function.
- $\star$ q_0 is the starting state.
- $\star$ q_{accept} is the accept state.
- $\star$ q_{reject} is the reject state.

(Nondeterministic) Turing Machines

A Nondeterministic Turing Machine is a 7-tuple $(Q, \Sigma, \Gamma, \delta, q_0, q_{\text{accept}}, q_{\text{reject}})$ where:

- $\star$ Q is the set of states.
- $\star$ Σ is the *input alphabet*.
- $\star$ Γ is the *tape alphabet* (and satisfies $\Gamma \subseteq \Sigma$).
- $\star$ $\delta \subseteq (Q \times \Gamma) \times (Q \times \Gamma \times \{L, R\})$ is the transition relation.
- $\star$ q_0 is the starting state.
- $\star$ q_{accept} is the accept state.
- $\star$ q_{reject} is the reject state.

Task: What changed, compared to the original Turing Machine definition?

(Nondeterministic) Turing Machines

A Nondeterministic Turing Machine is a 7-tuple $(Q, \Sigma, \Gamma, \delta, q_0, q_{\text{accept}}, q_{\text{reject}})$ where:

- $\star$ Q is the set of states.
- $\star$ Σ is the *input alphabet*.
- $\star$ Γ is the *tape alphabet* (and satisfies $\Gamma \subseteq \Sigma$).
- $\star$ $\delta \subseteq (Q \times \Gamma) \times (Q \times \Gamma \times \{L, R\})$ is the transition relation.
- $\star$ q_0 is the starting state.
- $\star$ q_{accept} is the accept state.
- $\star$ q_{reject} is the reject state.

Task: What changed, compared to the original Turing Machine definition?
Answer: δ is no longer a transition *function*! It is a transition *relation*.

(Nondeterministic) Turing Machines

A Nondeterministic Turing Machine is a 7-tuple $(Q, \Sigma, \Gamma, \delta, q_0, q_{\text{accept}}, q_{\text{reject}})$ where:

- ⋆ Q is the set of states.
- ⋆ Σ is the *input alphabet*.
- ⋆ Γ is the *tape alphabet* (and satisfies $\Gamma \subseteq \Sigma$).
- ⋆ $\delta \subseteq (Q \times \Gamma) \times (Q \times \Gamma \times \{L, R\})$ is the transition relation.
- ⋆ q_0 is the starting state.
- ⋆ q_{accept} is the accept state.
- ⋆ q_{reject} is the reject state.

Task: What changed, compared to the original Turing Machine definition?

(Nondeterministic) Turing Machines

A Nondeterministic Turing Machine is a 7-tuple $(Q, \Sigma, \Gamma, \delta, q_0, q_{\text{accept}}, q_{\text{reject}})$ where:

- $\star$ Q is the set of states.
- $\star$ Σ is the *input alphabet*.
- $\star$ Γ is the *tape alphabet* (and satisfies $\Gamma \subseteq \Sigma$).
- $\star$ $\delta \subseteq (Q \times \Gamma) \times (Q \times \Gamma \times \{L, R\})$ is the transition relation.
- $\star$ q_0 is the starting state.
- $\star$ q_{accept} is the accept state.
- $\star$ q_{reject} is the reject state.

Task: What changed, compared to the original Turing Machine definition?
Answer: δ is no longer a transition *function*! It is a transition *relation*.

(Nondeterministic) Turing Machines

~~$\delta : (Q \times \Gamma) \to (Q \times \Gamma \times \{L, R\})$ is the transition function.~~

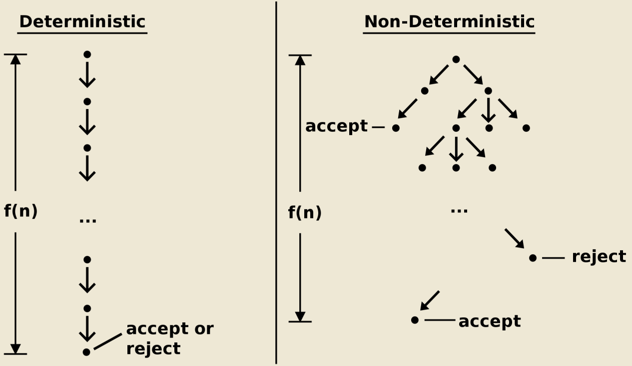
$\delta \subseteq (Q \times \Gamma) \times (Q \times \Gamma \times \{L, R\})$ is the transition relation.

(Nondeterministic) Turing Machines

~~$\delta : (Q \times \Gamma) \to (Q \times \Gamma \times \{L, R\})$ is the transition function.~~

$\delta \subseteq (Q \times \Gamma) \times (Q \times \Gamma \times \{L, R\})$ is the transition relation.

Multiple possible transitions!



I stole this image from Wikipedia. Thanks!

(Nondeterministic) Turing Machines

A NTM M accepts input x when M has *some* execution path that ends in q_{accept}. Otherwise:

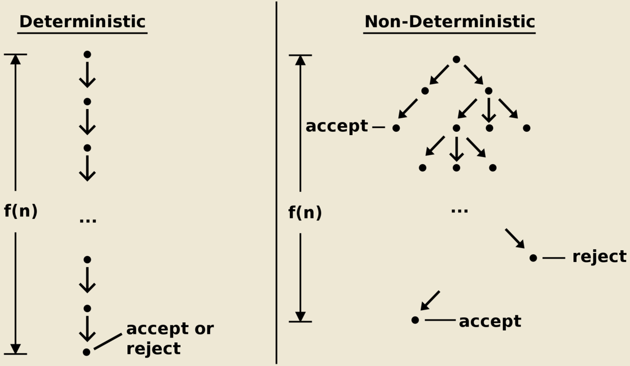
- $\star$ If all execution paths end in q_{reject}, M rejects input x.
- $\star$ Otherwise, it loops.

(Nondeterministic) Turing Machines

A NTM M accepts input x when M has *some* execution path that ends in q_{accept}. Otherwise:

- ⋆ If all execution paths end in q_{reject}, M rejects input x.
- ⋆ Otherwise, it loops.

In some sense, think of NTMs like job applications. If you have one job offer, you're good! Otherwise :(



I stole this image from Wikipedia. Thanks!

Subset Sum

Task: Let $S = \{5, 7, 11, 12, 14\}$. Can you find a subset $S' \subseteq S$ such that S' sums to 24? What about 27?

Subset Sum

Task: Let $S = \{5, 7, 11, 12, 14\}$. Can you find a subset $S' \subseteq S$ such that S' sums to 24? What about 27?

Answer: There is a subset that sums to 24, namely $\{5, 7, 12\}$. There are no subsets that sum to 27, on the other hand.

Subset Sum

Task: Let $S = \{5, 7, 11, 12, 14\}$. Can you find a subset $S' \subseteq S$ such that S' sums to 24? What about 27?

Answer: There is a subset that sums to 24, namely $\{5, 7, 12\}$. There are no subsets that sum to 27, on the other hand.

This is a specific case of the subset sum problem. Given a finite set of natural numbers S and a *target* $t \in \mathbb{N}$, can we find a $S' \subseteq S$ such that S' sums to t?

Subset Sum

Task: Let $S = \{5, 7, 11, 12, 14\}$. Can you find a subset $S' \subseteq S$ such that S' sums to 24? What about 27?

Answer: There is a subset that sums to 24, namely $\{5, 7, 12\}$. There are no subsets that sum to 27, on the other hand.

This is a specific case of the subset sum problem. Given a finite set of natural numbers S and a *target* $t \in \mathbb{N}$, can we find a $S' \subseteq S$ such that S' sums to t?

Task: Write a function `subset_sum(S, t)` that returns True iff S has a subset that sums to t, using pseudocode.

Subset Sum

Task: Write a function `subset_sum(S, t)` that returns True iff S has a subset that sums to t, using pseudocode.

Answer:

```
subset_sum(S, t):
  for every subset S’ of S:
    if S’ sums to t:
      return True
  return False
```

What is the runtime of `subset_sum(S, t)`, in terms of $|S|$ (the size of S)?

Subset Sum

Task: Write a function `subset_sum(S, t)` that returns True iff S has a subset that sums to t, using pseudocode.

Answer:

```
subset_sum(S, t):
  for every subset S' of S:
    if S' sums to t:
      return True
  return False
```

What is the runtime of `subset_sum(S, t)`, in terms of $|S|$ (the size of S)? $O(2^{|S|})$.

Subset Sum

Task: Write a function `subset_sum(S, t)` that returns True iff S has a subset that sums to t, using pseudocode. This time, make sure that the algorithm runs in polynomial time!

Subset Sum

Task: Write a function `subset_sum(S, t)` that returns True iff S has a subset that sums to t, using pseudocode. This time, make sure that the algorithm runs in polynomial time!

Answer:

(we don't know if it's possible or not)

Subset Sum

Task: Write a function `subset_sum(S, t)` that returns True iff S has a subset that sums to t, using pseudocode. This time, make sure that the algorithm runs in polynomial time!

Subset Sum

Task: Write a function `subset_sum(S, t)` that returns True iff S has a subset that sums to t, using pseudocode. This time, make sure that the algorithm runs in polynomial time!

Answer:

(we don't know if it's possible or not)

Subset Sum: Nondeterministic Solution

Wait! What about the following code? Does this solve subset sum?

```
def subset_sum_2(S, t):
  randomly choose a subset S' of S
  if S' sums to t:
    return True
  return False
```

Subset Sum: Nondeterministic Solution

Wait! What about the following code? Does this solve subset sum?

```
def subset_sum_2(S, t):
  randomly choose a subset S’ of S
  if S’ sums to t:
    return True
  return False
```

No :(
Remember that Turing machines are *deterministic*: TMs can’t generate a random number.

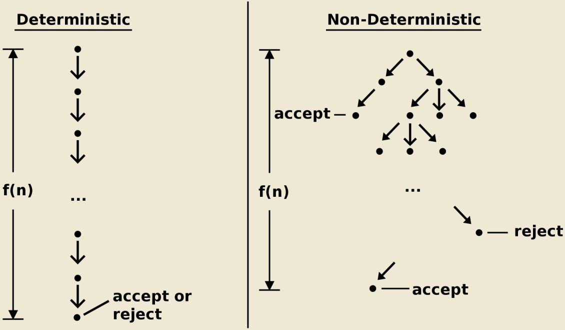
Subset Sum: Nondeterministic Solution

```
def subset_sum_2(S, t):
  nondeterministically choose a subset S' of S
  if S' sums to t:
    return True
  return False
```

Subset Sum: Nondeterministic Solution

```
def subset_sum_2(S, t):
  nondeterministically choose a subset S' of S
  if S' sums to t:
    return True
  return False
```

But this pseudocode is valid for a *nondeterministic* Turing machine.



I stole this image from Wikipedia. Thanks!

Subset Sum: Nondeterministic Solution

```
def subset_sum_2(S, t):
  nondeterministically choose a subset S' of S
  if S' sums to t:
    return True
  return False
```

Note that this NTM accepts (S, t) iff there is a subset of S that sums to t.

A NTM M **accepts** input x when M has *some* execution path that ends in q_{accept}. Otherwise:

- ⋆ If all execution paths end in q_{reject}, M **rejects** input x.
- ⋆ Otherwise, it loops.

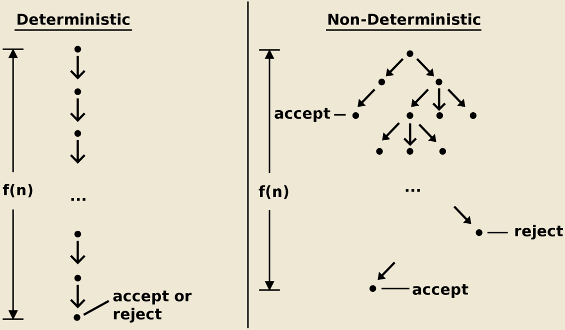
What is the *runtime* of this NTM?

Runtime of a NTM

The runtime of a NTM is the maximum runtime of any execution sequence.

Runtime of a NTM

The runtime of a NTM is the maximum runtime of any execution sequence.



I stole this image from Wikipedia. Thanks!

In other words, it's $f(n)$ in the above diagram.

Runtime of a NTM

```
def subset_sum_2(S, t):
  nondetermistically choose a subset S' of S
  if S' sums to t:
    return True
  return False
```

Task: Does the above NTM run in polynomial time?

Runtime of a NTM

```
def subset_sum_2(S, t):
  nondetermistically choose a subset S' of S
  if S' sums to t:
    return True
  return False
```

Task: Does the above NTM run in polynomial time?

Answer: Yes!

Runtime of a NTM

```
def subset_sum_2(S, t):
  nondetermistically choose a subset S' of S
  if S' sums to t:
    return True
  return False
```

Task: Does the above NTM run in polynomial time?

Answer: Yes!

Conclusion:

- ⋆ The subset sum problem can be solved in polynomial time by a NTM.
- ⋆ We don't know if subset sum can be solved in polynomial time by a TM.

P versus NP?

P versus NP?

- ⋆ P is the set of all languages decidable in polynomial time by a Turing machine.

P versus NP?

- ⋆ P is the set of all languages decidable in polynomial time by a Turing machine.
- ⋆ NP is the set of all languages decidable in polynomial time by a nondeterministic Turing machine.

$\mathrm{P} \subseteq \mathrm{NP}$ because

P versus NP?

- ⋆ P is the set of all languages decidable in polynomial time by a Turing machine.
- ⋆ NP is the set of all languages decidable in polynomial time by a nondeterministic Turing machine.

$\mathrm{P} \subseteq \mathrm{NP}$ because every Turing machine is a nondeterministic Turing machine.

P versus NP?

- ⋆ P is the set of all languages decidable in polynomial time by a Turing machine.
- ⋆ NP is the set of all languages decidable in polynomial time by a nondeterministic Turing machine.

$\mathrm{P} \subseteq \mathrm{NP}$ because every Turing machine is a nondeterministic Turing machine.

P versus NP problem: Is $\mathrm{P} = \mathrm{NP}$?

Examples of problems in NP

The following languages are in NP, but we don't know if any of them are in P.

- ⋆ Subset Sum.
- ⋆ Boolean Satisfiability Problem.
- ⋆ Graph Isomorphism Problem.
- ⋆ Vertex Cover Problem.
- ⋆ Knapsack Problem.
- ⋆ Hamiltonian Path Problem.
- ⋆ Generalized Sudoku.

Examples of problems in NP

The following languages are in NP, but we don't know if any of them are in P.

- ⋆ Subset Sum.
- ⋆ Boolean Satisfiability Problem.
- ⋆ Graph Isomorphism Problem.
- ⋆ Vertex Cover Problem.
- ⋆ Knapsack Problem.
- ⋆ Hamiltonian Path Problem.
- ⋆ Generalized Sudoku.

Showing that any one of those problems is not in P will net you $1 million USD.

Last question of the day!

Question: Suppose L is decidable by a NTM. Is L decidable by a TM (disregarding runtime)?

Last question of the day!

Question: Suppose L is decidable by a NTM. Is L decidable by a TM (disregarding runtime)?

Answer: Yes.

```
def simulate_NTM(ntm, input):
  while True:
    execute ntm(input) one step.
    if there are multiple possible transitions,
    spawn a thread here to simulate each possible transition
```