Space Complexity

Resource consumption

- time (so far)
- space (this lecture)

As before, use Turing Machine model.

- mathematically tractable
- close enough to real computers . . .

Deterministic Space Complexity

Let M be a deterministic TM that halts on all inputs.

Definition: The **space complexity** of M is

$$f : \mathcal{N} \longrightarrow \mathcal{N}$$

where $f(n)$ is the maximum number of tape cells that M scans on any input of length n.

We say that *M runs in space $f(n)$*.

Space Complexity

Let M be a non-deterministic TM such that all *branches* halt on all inputs.

Definition: The **space complexity** of M is $f : \mathcal{N} \longrightarrow \mathcal{N}$ where $f(n)$ is the maximum number of tape cells that M scans on any *branch* of its computation on any input of length n.

Space complexity usually denoted by asymptotic notation.

Useful Lemma

Lemma: If M runs in $O(f(n))$ space, then it runs in $2^{O(f(n))}$ time

Proof: There are only $2^{O(f(n))}$ possible configurations.

Space Complexity

Definition:

$$\text{SPACE}(f(n)) = \{L | L \text{ is a language decided by an } O(f(n)) \text{ space deterministic TM}\}$$

Space Complexity

Definition:

$$\text{NSPACE}(f(n)) = \{L | L \text{ is decided by an } O(f(n)) \text{ space non-deterministic TM}\}$$

Example: SAT

SAT is NP-complete.

- probably can't be solved in poly time
- but can be solved in linear *space*

Example: SAT

M_1 : on $\langle \phi \rangle$, where ϕ is a Boolean formula

- for each truth assignment to variables $x_1, \ldots, x_m$
  - evaluate ϕ on that assignment
  - if ϕ evaluates to **true**, *accept*.
- otherwise *reject*

Example: SAT

Claim: M_1 runs in linear space.

- each iteration of the loop reuses same portion of the tape
- current truth assignment needs $O(m)$ space
- number of variables at most $O(n)$ space
- total space is $O(n)$.

Example: $E_{\textsf{NFA}}$

Definition:

$$E_{\textsf{NFA}} = \{\langle A\rangle | A \text{ is a NFA and } L(A) = \emptyset\}$$

- does the NFA accept any strings?
- not known to be in NP or coNP.

Example: $E_{\textsf{NFA}}$

Here is a linear-space NTM that decides $\overline{E_{\textsf{NFA}}}$.

N : on input $\langle M \rangle$ where M is a NFA

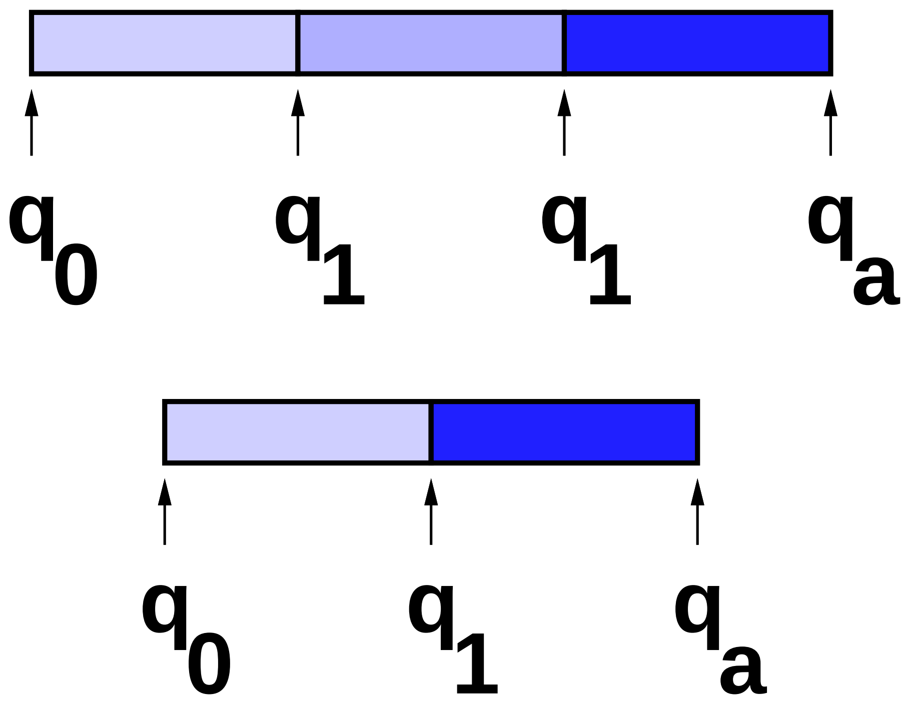
- place a marker on start state of M
- repeat 2^q times, where q is number of states
  - non-deterministically select input symbol x
  - move markers to simulate reading x
  - if marker on accept state, *accept*
- otherwise *reject*.

Why does this work?

Example: E_{NFA}

Claim: N runs in linear space.

- If M accepts any strings, it accepts one of length q
- otherwise some accepting branch repeats a state



Example: $E_{\textbf{NFA}}$

Claim: N runs in linear space.

- If M accepts any strings, it accepts one of length q
- only space needed is $O(n)$ marker locations

Savitch's Theorem

Simulating NTM by DTM

- for time, we think it requires exponential increase
- for space, Savitch's Theorem shows that any $f(n)$-space NTM can be converted to an $f^2(n)$-space DTM.
- one of the earliest results on space complexity.

Savitch's Theorem

Theorem: For any $f : \mathcal{N} \longrightarrow \mathcal{N}$, $f(n) \geq n$,

$$\text{NSPACE}(f(n)) \subseteq \text{SPACE}(f^2(n))$$

Remark: Actually holds for $f(n) \geq \log n$.

Savitch's Theorem

Need a deterministic simulation of a $f(n)$-space NTM.

Naive approach:

- try all branches one-by-one
- an $f(n)$-space branch could take $2^{O(f(n))}$ non-deterministic steps
- keeping track of choices could take $2^{O(f(n))}$ space!

Savitch's Theorem

Different approach:

Definition: The **yieldability** problem: given NTM configurations c_1 and c_2, and $t > 0$, can the NTM get from c_1 to c_2 in t steps?

Savitch's Theorem

Different approach:

Definition: The **yieldability** problem: given configurations NTM c_1 and c_2, and $t > 0$, can the NTM get from c_1 to c_2 in t steps?

Let

- c_1 be the starting configuration
- c_2 the accepting configuration
- t the max number of steps the NTM can use.

Yieldability: Idea

We give a deterministic, recursive yieldability algorithm.

- search for intermediate c_m, and test whether
- c_1 can get to c_m in $t/2$ steps, and
- c_m can get to c_2 in $t/2$ steps
- reusing space on each recursion

Yieldability: Idea

Space requirements:

- need to store recursion stack
- leach level needs $O(f(n))$ to store a configuration
- depth of recursion is $\log(t)$
- initially t is max time: $2^{O(f(n))}$, so
- $\log t = O(f(n))$
- total space $O(f^2(n))$

Yieldability: Proof

CANYIELD(c_1, c_2, t):

- if $t = 1$ then
  - if $c_1 = c_2$, *accept*
  - c_1 yields c_2 in one step, *accept*
  - otherwise *reject*.
- if $t > 1$, then for each configuration c_m
  - run CANYIELD$(c_1, c_m, \lceil t/2 \rceil)$
  - run CANYIELD$(c_m, c_2, \lceil t/2 \rceil)$
  - if both accept, *accept*
- otherwise *reject*.

Savitch's Theorem: Proof

Let N be a NTM.

Modify N so that it accepts by

- clearing its tape . . .
- and moving to leftmost cell.

Call this configuration c_a

Savitch's Theorem: Proof

Let

- c_s starting configuration on input w
- d constant such that N has $\leq 2^{df(n)}$ configurations using $f(n)$ tape, where $n = |w|$.
- $\leq 2^{df(n)}$ is upper bound on running time of any branch of N on w.

Savitch's Theorem: Proof

M : on input w

- output result of CANYIELD$(c_s, c_a, 2^{df(n)})$.

Correctness clear, but need to analyze space.

Savitch's Theorem: Proof

Space analysis:

- Each invocation stores c_1, c_2, t
- which takes space $O(f(n))$.
- Each level of recursion cuts t in half.
- Initially $t = 2^{df(n)}$,
- depth is $\log 2^{df(n)} = O(f(n))$.
- total space is $O(f^2(n))$.

PSPACE

Definition: PSPACE $= \bigcup_k$ SPACE(n^k)

Examples

- SAT $\in$ PSPACE
- $E_{\text{NFA}} \in$ PSPACE

NPSPACE

Question: Can we define NPSPACE analogously?

NPSPACE

Question: Can we define NPSPACE analogously?

Answer: Yes, but Savitch's Theorem states that

$$\mathrm{PSPACE} = \mathrm{NPSPACE}$$

PSPACE

Claim: $P \subseteq \text{PSPACE}$

Why?

PSPACE

Claim: Any machine that runs in time $f(n)$ runs in space $\leq f(n)$

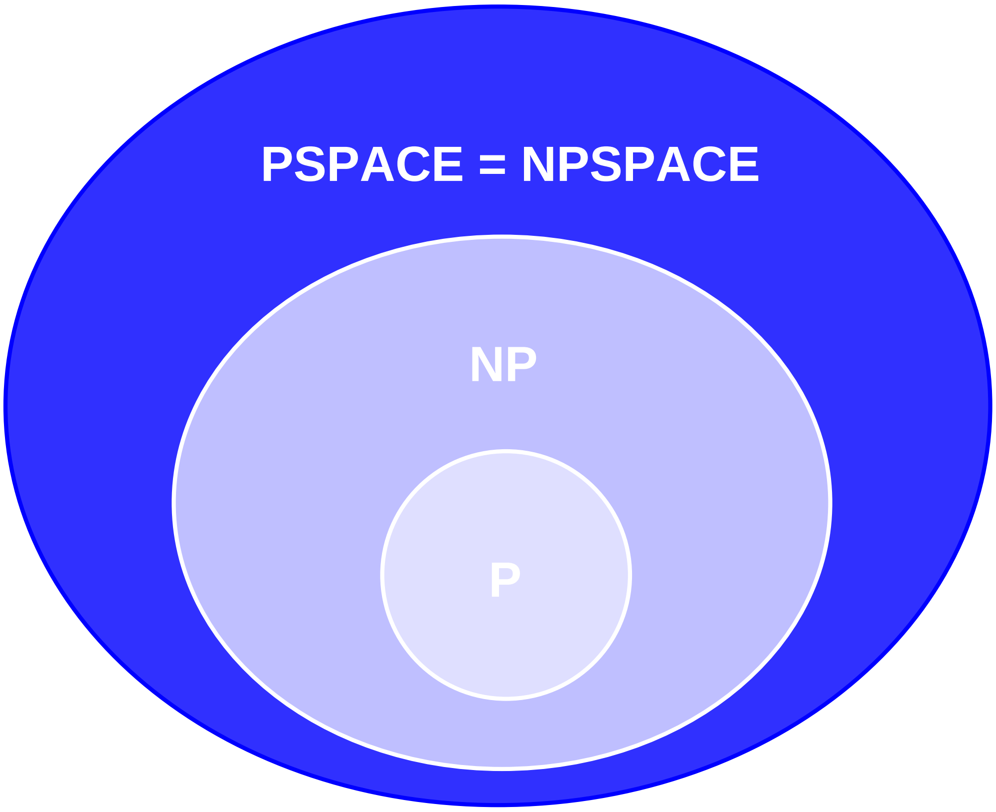
Corollary: NP $\subseteq$ NPSPACE $=$ PSPACE

PSPACE

Claim: PSPACE $\subseteq$ EXPTIME

Proof: A TM that halts may not repeat a configuration.

Relations



We know that

- $P \neq$ EXPTIME
- not much else