Space Complexity

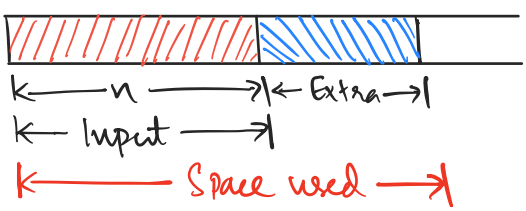
We will follow a slightly different approach from the presentation in Sipser.

Def (Space Complexity): The space complexity of a DTM M is given by $f: \mathbb{N} \to \mathbb{N}$, where $f(n)$ is the maximum space used by any input of length n.

For an NTM, it is the maximum space used by any branch of computation of an input of length n.

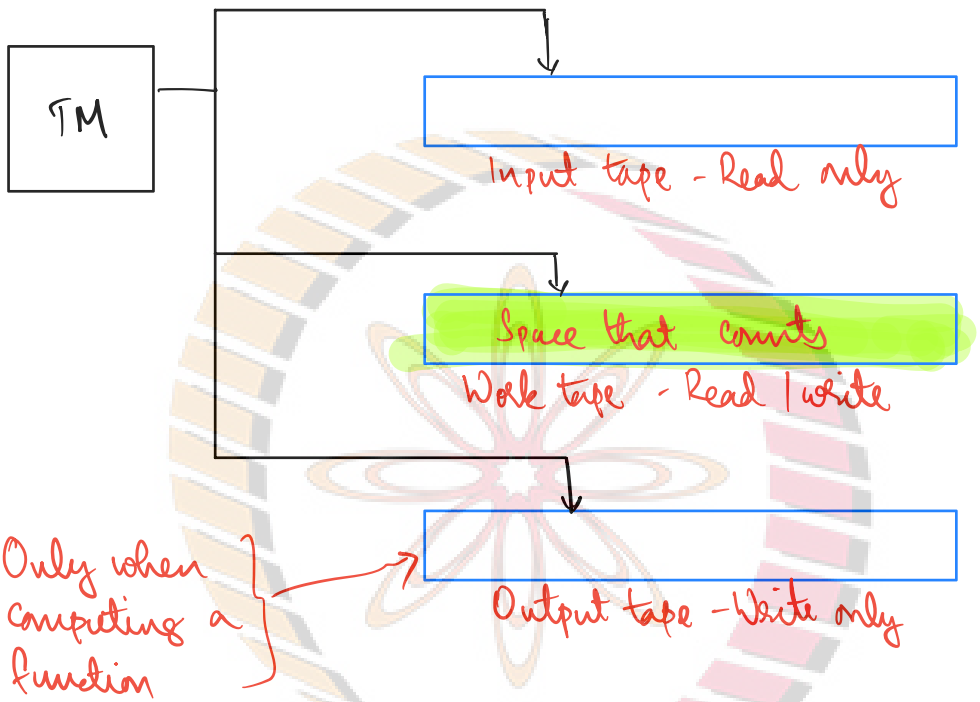
But how do we define the space used by the TM?

A naive approach is to measure the number of cells used on the input tape.



But this forces space usage to be $\geq n$ (input length) always!

But there are algorithms that need only sublinear space ($o(n)$ space). To capture this, we need another model.



The space used by the work tape is what "counts" towards the space usage of the TM.

Def 8.2: Let $f: \mathbb{N} \to \mathbb{R}^+$ be a function

Then

$DSPACE(f(n))$ (SPACE $(f(n))$ in the book)

$= \{L \mid L$ is decided by an $O(f(n))$ space DTM$\}$

NSPACE(f(n)) = { L | L is decided by an O(f(n)) space NTM }

Example: 1) 3-SAT ∈ DSPACE(n).

Let $x_1, x_2, \ldots x_l$ be the variables and $C_1, C_2, \ldots C_k$ be the clauses

→ Run an l-bit counter through the 2^l assignments. (requires l bits)

→ For each assignment, evaluate ϕ. (Constant extra space)

→ Accept if ϕ is satisfied.

→ Reject if not satisfied after checking all the possibilities.

Space used = $O(l) = O(n)$

---

2) PALINDROME : We had seen a DTIME(n) algorithm using 2-tape TM.

→ For $i = 1$ to $\lfloor n/2 \rfloor$

Read x_i, x_{n+1-i}

If $x_i \neq x_{n+1-i}$, reject

→ Accept if not rejected, so far.

Space is needed to store i, $n+1-i$ and x_i.

This requires $O(\log n)$ space.

$$\text{PALINDROME} \in \text{DSPACE}(\log n)$$

Space seems to be more powerful than time.

Definition : $L = \text{LOGSPACE} = \text{DSPACE}(\log n)$

$NL = \text{NSPACE}(\log n)$.

$$\text{PSPACE} = \bigcup_{k=1}^{\infty} \text{DSPACE}(n^k)$$

L, NL, PSPACE are some of the common complexity classes based on space usage.

Relation between some of the classes

Theorem: $DSPACE(f(n)) \subseteq NSPACE(f(n))$

Proof: A DTM can be viewed as an NTM.

So $L \subseteq NL$.

Theorem: $DTIME(f(n)) \subseteq DSPACE(f(n))$

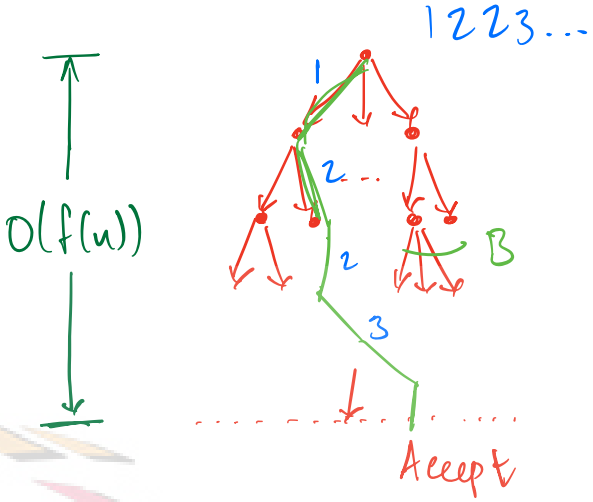
Proof: The TM can write at most 1 cell of the tape in each time step. So maximum space used in $f(n)$ time is $f(n)$.

So $P \subseteq PSPACE$

Theorem: Suppose $f(n) \geq \log n$. Then $NTIME(f(n)) \subseteq DSPACE(f(n))$

Proof: Recall the simulation of NTM by DTM as explained in lecture 31 and lecture 47.

We keep track of the many computation paths and do BFS on the computation tree.



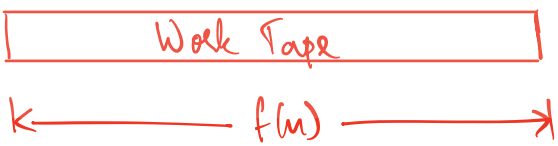
Each computation path can be encoded by a string of length $O(f(n))$. We can keep track of such strings by an $O(f(n))$ length counter.

For a fixed string, we can simulate the NTM using $O(f(n))$ space.

Theorem: $DSPACE(f(n)) \subseteq DTIME(2^{O(f(n))})$

(assuming $f(n) \geq \log n$)

Proof:



For a fixed input w, the no. of configurations

of the space bounded TM is bounded as follows:

No. of configurations of the DSPACE(f(n)) machine $\leq |\Gamma|^{f(n)} * n * f(n) * |Q|$

(Contents of work tape; Head Positions)

$= 2^{c_1 f(n)} \cdot n f(n) \cdot |Q|$

(Since $f(n) \geq \log n$) $\leq 2^{c_2 f(n)}$

$= 2^{O(f(n))}$

The $DTIME_{\wedge}$ simulating machine can run serially for $2^{O(f(n))}$ steps. Since the no. of configurations of the DSPACE machine is bounded, it has to come to a decision by then. Accept if DSPACE machine accepts. Else reject.

Corollary: $L \subseteq P$.

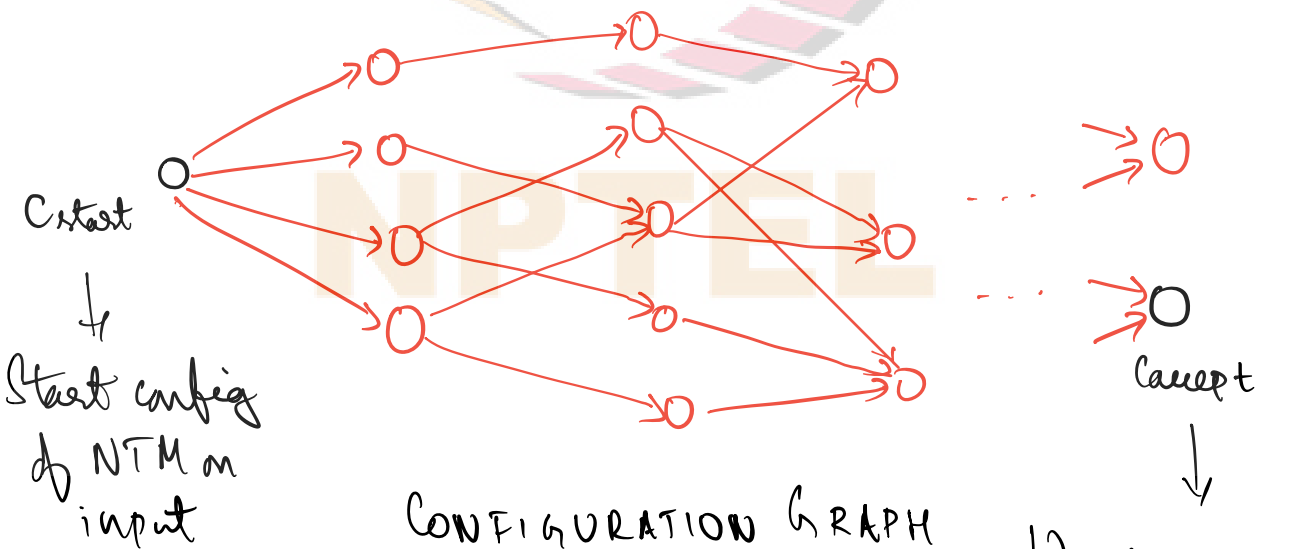
Set $f(n) = \log n$

Theorem: $NSPACE(f(n)) \subseteq DTIME(2^{O(f(n))})$

(assuming $f(n) \geq \log n$)

Proof: We cannot use the same proof as above since the NTM (NSPACE machine) can have > 1 successor for a configuration. If the branching factor is B, we may have $B^{2^{O(f(n))}}$ configurations to check!

But total no. of configurations is bounded as above. This is $\leq 2^{O(f(n))}$.



In this graph, we have to check : Is there a path from C_{start} to C_{accept}?

We can use BFS or DFS. $O(|V|+|E|)$ time.

$$\leq O(|V|^2)$$

$$= O\left(2^{O(f(n))}\right)^2$$

$$= O\left(2^{O(f(n))}\right)$$

$$= 2^{O(f(n))}$$

Corollary : $NL \subseteq P$

↳ Set $f(n) = \log n$.