Space Complexity

The space complexity of a program is how much memory it uses.

Measuring Space

When we compute the space used by a TM, we *do not count the input* (think of input as ***read-only***).

We say that TM M ***runs in space*** $S(n)$ if for all inputs of length n, M uses at most $S(n)$ cells in total on its work-tapes.

Example: SAT

The problem SAT can be decided in linear space.

The natural algorithm is to try all assignments. The storage required is to keep track of the assignment, and to do the verification. Both take at most linear space.

Time versus Space

Since it takes one time-step to access one memory cell, the number of memory cells a TM accesses cannot exceed the number of steps it runs for:

If TM runs in time $T(n)$, then it runs in space $T(n)$.

Infinite Loops

We repeat an observation we used before:

Fact. *Suppose a deterministic TM runs in $S(n)$ space with g tape symbols and q states. If TM runs for longer than $qng^{S(n)}$ steps on input of length n, then it is stuck in an infinite loop.*

We Can Assume Space-Bounded TMs Halt

Theorem. *If L accepted by TM M in space $S(n)$ ($S(n) \geq \log n$), then L is accepted by TM M' that runs in space $O(S(n))$ but always halts.*

Proof. Build TM M' that is almost M, but also has a binary counter that counts the number of steps that M has been going. When counter exceeds above bound, M' stops and rejects.

A counter that counts up to T takes $O(\log T)$ space. Thus, space for counter is $O(\log(qng^{S(n)})) = O(S(n) + \log n)$. So M' still runs in $O(S(n))$ space.

At Most Exponential Blow-Up

Another consequence of above fact is:

Theorem. *If language L is decided in space $S(n)$ (where $S(n) \geq \log n$), then there is constant C such that L is decided in time $O(C^{S(n)})$.*

Constant Space

It turns out that:

Fact. *The languages decidable in constant space are precisely the regular languages.*

Nondeterministic Space

NTM M ***runs in space*** $S(n)$ if for any input of length n and for any branch, M uses at most $S(n)$ cells on its work-tape.

The above results about time and space still apply.

Polynomial Space

$\mathcal{PSPACE}$ is the set of languages that can be decided in polynomial space; $\mathcal{NPSPACE}$ is the set of languages that can be decided in polynomial space by a nondeterministic machine.

It is perhaps surprising that these two classes are known to be the same:

Savitch's Theorem

Savitch's Theorem. $\mathcal{PSPACE} = \mathcal{NPSPACE}$

We show that if there is an NTM for language L that uses $S(n)$ space (where $S(n) \geq \log n$), then there is a DTM for L that uses $O((S(n))^2)$ space. The theorem follows since the square of a polynomial is still a polynomial.

Proof of Savitch's Theorem

Let M be NTM that always halts and runs in space $S(n)$. Then M runs in at most $T = qng^{S(n)}$ steps for q states and g tape symbols. Re-program M to erase its work-tape and end in a particular state when it accepts.

We use configurations and pad each to have length $S(n)$. So there is unique starting configuration C_s for given input and unique accepting configuration C_a. The question is: *is there a sequence of moves from C_s to C_a?*

A Recursive Function

Define boolean function f that takes as arguments two configurations C_1 and C_2 and integer i such that $f(C_1, C_2, i)$ is TRUE if M can get from C_1 to C_2 in at most i steps and FALSE otherwise. This can be computed recursively:

The Function f

Calculating $f(C_1, C_2, i)$*:*
if *i=1* ***then return*** *oneStep*(M, C_1, C_2)
else $\{$
  for *all strings* C_3 *of length* $S(n)$ $\{$
    if $f(C_1, C_3, i/2)$ TRUE ***and*** $f(C_3, C_2, i/2)$ TRUE
      then return TRUE
  $\}$
  failing which: ***return*** FALSE
$\}$

The boolean function *oneStep* determines whether C_1 follows from C_2 by the rules of M.

Conclusion

So we build a deterministic machine and ask it to calculate $f(C_s, C_a, T)$. (T was the upper bound on M's running time.)

How much storage does this use? The recursive program uses a stack, pushing a "snapshot" of its variables each time it calls itself. Each configuration and hence each snapshot uses $O(S(n))$ space. The maximum number of snapshots on stack is $\log T$ (the depth of the recursion). Since $\log T$ is $O(S(n))$, it follows that space used is $O(n^{2k})$, as claimed.

Examples in $\mathcal{PSPACE}$

Deciding whether an RE generates all strings is in $\mathcal{PSPACE}$ (discussed in exercises).

It can be shown that any language given by a context-sensitive grammar is in $\mathcal{PSPACE}$.

Practice

Show that $\mathcal{PSPACE}$ is closed under the star operation.

Solution to Practice

Suppose $A \in \mathcal{PSPACE}$. We build algorithm for A^*. Assume input is w. Then start by calculating for each substring of w whether in A. Then try all possible ways to split w into substrings and see whether in any case all substrings are in A. Extra storage is polynomial.

Logarithmic Space

The class $\mathcal{L}$ is the set of problems solvable in $O(\log n)$ space. The class $\mathcal{NL}$ is the set of problems solvable in $O(\log n)$ space by an NTM.

Example: $\{ \mathtt{0}^n\mathtt{1}^n : n \geq 0 \}$ is in $\mathcal{L}$. The key is that a binary ***counter*** can be maintained in logarithmic space. Here the TM counts the 0's and 1's, and compares the counts.

The Complexity Chain

From earlier result it follows that $\mathcal{NL} \subseteq \mathcal{P}$. So:

$$\mathcal{L} \subseteq \mathcal{NL} \subseteq \mathcal{P} \subseteq \mathcal{NP} \subseteq \mathcal{PSPACE} = \mathcal{NPSPACE}$$

The only other thing humanity knows about how these classes compare is:

$$\mathcal{L} \neq \mathcal{PSPACE}$$

Summary

The space complexity of a TM is the space or memory taken as a function of the input length n in the worst case. The class $\mathcal{PSPACE}$ is the set of all languages that are decidable by a TM running in polynomial space. It is known that $\mathcal{PSPACE} = \mathcal{NPSPACE}$. Examples of languages in $\mathcal{PSPACE}$ include ALL_{re} and any context-sensitive language.

It is known that $\mathcal{P} \subseteq \mathcal{NP} \subseteq \mathcal{PSPACE}$ and it is believed that there is not equality.