Theory of Computation:
Space complexity and PSPACE

Space constructible functions

- $S : \mathbb{N} \to \mathbb{N}$ where there is a TM that on input x computes $S(|x|)$ by using $O(S(|x|))$ read/write cells.
- Intuition: The machine "knows" the space bound it is operating under.
- $S(n) > \log(n)$ because we need to be able to remember which cell of the input tape is currently being read.
- $S(n)$ could be much smaller than n as this does not conflict designing algorithms that read the entire input and use sublinear space: counters are an example.
- Examples of space constructible functions: $\log n, n, 2^n$, all the space bounding functions we will see.

Space-bounded computation

- Let $S : \mathbb{N} \to \mathbb{N}$ and $L \subseteq \{0,1\}^*$. Then $L \in SPACE(S(n))$ if there is a constant c and a TM M deciding L such that on any input of length n at most $c.S(n)$ cells on M's work tapes are ever visited by M's tape head during its computation.
- $L \in NSPACE(S(n))$ if there is an NDTM M deciding L that never uses more than $c.S(n)$ nonblank tape locations on length n inputs, no matter the sequence of nondeterministic choices – on every branch total space used is $c.S(n)$.

Time and Space connections

- $DTIME(S(n)) \subseteq SPACE(S(n))$: at one time step at most one cell can be accessed.
- $SPACE(S(n))$ machine: Can compute $2^{S(n)}$!
- So far: $DTIME(S(n)) \subseteq SPACE(S(n)) \subseteq NSPACE(S(n))$.

$NSPACE(S(n)) \subseteq DTIME(2^{O(S(n))})$

- Configuration Graph $G_{M,x}$ for a NDTM M and an input x: vertices represent configurations;
  Directed edges from a configuration C_1 to configuration C_2 if the transition function allows M to go from C_1 to C_2 in the following step.
- If M is a space $S(n)$ NDTM , then every configuration can use at most $S(n)$ non-blank cells on the work tape.

$NSPACE(S(n)) \subseteq DTIME(2^{O(S(n))})$

- Assumption: It is possible to assume that an NDTM has at most 2 possible transitions from any given configuration by modifying the transition function
- How can you do this? Think of the transition tree; suppose you are at a vertex v where more than 2 transitions are possible, then this vertex has a subset T' of more than 2 children.
  Can you add more states so that you can replace the subtree on $v \cup T'$ with a binary subtree having v as its root and vertices corresponding to T' as its leaves?
- This means each vertex in $G_{M,x}$ has at most 2 directed edges going out of it.

$NSPACE(S(n)) \subseteq DTIME(2^{O(S(n))})$

- Modify M further to erase all its work tape contents before halting: This makes M have exactly one configuration C_t for accepting, where M halts and outputs 1.
- M accepts input x if and only if there is a directed path from C_s (start configuration) to C_t (accepting configuration) in $G_{M,x}$.

$NSPACE(S(n)) \subseteq DTIME(2^{O(S(n))})$

- Every vertex in $G_{M,x}$ can be described using $cS(n)$ bits (c only depends on M's description) and $G_{M,x}$ has at most $2^{cS(n)}$ vertices:
- A configuration is completely described by a string of the form $\alpha q \beta$ where $\alpha\beta$ are the tape contents, the tape head is on the cell where β starts and the current state is q. This string is at most $cS(n)$ bits long.
- Thus, the number of configurations is at most $2^{cS(n)}$ = the number of vertices in $G_{M,x}$.

$NSPACE(S(n)) \subseteq DTIME(2^{O(S(n))})$

- Enumerate over all possible configurations and construct the graph $G_{M,x}$ in $2^{O(S(n))}$ time. Check whether there is a directed path from C_s to C_t n $G_{M,x}$ using the BFS algorithm.

Space complexity classes

- $PSPACE = \bigcup_{c>0} SPACE(n^c)$
- $NPSPACE = \bigcup_{c>0} NSPACE(n^c)$
- $L = SPACE(\log n)$
- $NL = NSPACE(\log n)$

PSPACE

- $3 - SAT \in PSPACE$: Use linear space to cycle through all assignments and determine if there is a satisfying assignment.
- $NP \subseteq PSPACE$: Use the same space to enumerate all possible certificates and determine if there is a certificate for the input belonging to the language.

Space Hierarchy Theorem

If f, g are space constructible functions satisfying $f(n) = o(g(n))$ then
$SPACE(f(n)) \subsetneq SPACE(g(n))$

- Diagonalization similar to the Time Hierarchy Theorem.
- We show that $SPACE(n) \subsetneq SPACE(n^{1.1})$ and all other pairs have similar arguments.
- UTM: Can design a UTM $\mathcal{U}$ where the space required to simulate a TM M on an input x is a constant factor of the space needed by M itself on x.

Space Hierarchy Theorem

Diagonalization machine M:

- On input x, run $\mathcal{U}$ for at most $2^{|x|^{1.1}}$ steps to simulate M_x on x.
- If $\mathcal{U}$ tries to use more than $n^{1.1}$ cells then reject.
- If $\mathcal{U}$ does not halt within $2^{|x|^{1.1}}$ steps it means that M_x loops on x. M will output 1.
- If $\mathcal{U}$ outputs some bit b then M outputs $1 - b$.

Space Hierarchy Theorem

- So M halts within $2^{n^{1.1}}$ time and uses at most $n^{1.1}$ space.
- L is the language accepted by M. We show that $L \notin SPACE(n)$.
- By contradiction, let N be a machine that uses space cn. We can modify N so that on any input x if N runs for $2^{|x|}$ steps and does not halt, then it goes to the reject state r as it must have entered a loop.

Space Hierarchy Theorem

- $\mathcal{U}$ takes at most $c'c|x|$ space to simulate N on any input x.
- There is some n_0 such that $n^{1.1} > c'cn$. Let x be a representation of N whose length is at least n_0.
- $\mathcal{U}$ will obtain an output b on $M_x = N$ within space $n^{1.1}$ and time $2^{n^{1.1}}$ but by definition of M, M and $M_x = N$ do not agree on x's membership in L.

PSPACE-completeness

- A language L is PSPACE-hard if for every language L' in PSPACE, $L' \leq_p L$.
- Moreover, if L belongs to PSPACE, then L is called PSPACE-complete.
- If we show that a PSPACE-complete problem is in P, then it would mean $PSPACE \subseteq P \implies PSPACE = P$
  As we know $NP \subseteq PSPACE$, this would also mean $P = NP$.

Quantified Boolean Formula

- QBF: of the form $Q_1 x_1 Q_2 x_2 \ldots Q_n x_n \phi(x_1, x_2, \ldots, x_n)$ where each $Q_i \in \{\exists, \forall\}$, the x_is are from $\{0, 1\}$ and ϕ is an unquantified Boolean formula.
- All variables of ϕ are bound by a quantifier, so a QBF is always either true or false.
- TQBF: set of quantified Boolean formulas that are true.

TQBF is PSPACE-complete

Proof Sketch:

- $\psi = Q_1 x_1 Q_2 x_2 \ldots Q_n x_n \phi(x_1, x_2, \ldots, x_n)$, where size of ϕ is m. We consider a general form where some of the variables take constant values and all others are quantified.
- TQBF is in PSPACE: Suppose $Q_1 = \exists$; check if either $x_1 = 1$ or $x_1 = 0$ makes the rest of the formula true.
  Suppose $Q_1 = \forall$; check if both $x_1 = 1$ or $x_1 = 0$ makes the rest of the formula true.
  Recursively solve for the rest of the formula.

TQBF is PSPACE-complete

- Writing down the formula when a few variables have been assigned values takes $O(m)$ space.
- The amount of space for checking for different partial assignments can be reused.
- Recursion for space used: $s_{n,m} = s_{n-1,m} + O(m)$
  So, $s_{n,m} = O(nm)$.

TQBF is PSPACE-complete

Proof Sketch:

- $\psi = Q_1 x_1 Q_2 x_2 \ldots Q_n x_n \phi(x_1, x_2, \ldots, x_n)$, where size of ϕ is m.
- TQBF is PSPACE-hard: Take any language L in PSPACE. Let M be a machine that decides L in $S(n)$ space for input $x \in \{0,1\}^n$.
- This means that the configuration graph $G_{M,x}$ has $N = 2^{O(S(n))}$ vertices.
- A QBF is constructed of size $O(S(n)^2)$ that is true iff M accepts x.
- This QBF tries to capture the path from C_s to C_t of $G_{M,x}$.

TQBF is PSPACE-complete

Proof Sketch:

- We want $\phi_{C_s,C_t,i}$ = A QBF that is true iff there is a path from C_s to C_t of length i. Note that $i \leq N$.
- Base case: It is possible to create an unquantified Boolean formula corresponding to an pair of configurations C and C' where there is a directed edge from C to C'. Denote by $\phi_{C,C',1}$.
- Now we will try to give an inductive definition of a QBF for paths from configuration C to C' such that all variables except C and C' are quantified and plugging in the value of C and C' will be exactly the answer of whether such a path exists or not.
- So for $C = C_s$ and $C' = C_t$, the corresponding formula is what we want.

TQBF is PSPACE-complete

- First attempt at inductive definition:
  $\phi_{C_s,C_t,i} = \exists C[\phi_{C_s,C,i/2} \wedge \phi_{C,C_t,i/2}]$.
  If we try to define like this the QBF will be of size N which is too large for the allowed reduction time.
- Another way: $\phi_{C_s,C_t,i} = (\exists C)(\forall C_1)(\forall C_2)[((C_1 = C_s) \wedge (C_2 = C)) \vee ((C_1 = C) \wedge (C_2 = C_t)) \implies \phi_{C_1,C_2,i/2}]$.
  Similar definitions for all pairs C, C' of configurations.
  Recursion for $\text{size}\phi_{\cdot,\cdot,i} = \text{size}\phi_{\cdot,\cdot,i-1} + O(\log N) = O(S(n)^2)$.

TQBF is NPSPACE-complete

- It can also be shown that the same proof works even when we are working on languages of NPSPACE. Any language of NPSPACE can also be reduced to TQBF in polynomial time.
- This implies that PSPACE = NPSPACE.

Savitch's Theorem

For any space constructible $S : \mathbb{N} \to \mathbb{N}$ with $S(n) \geq \log n$, $NSPACE(S(n)) \subseteq SPACE(S(n)^2)$.
Much more structure than simply saying PSPACE $=$ NPSPACE.

Savitch's Theorem: Proof

- Let $L \in NSPACE(S(n))$ be decided by M such that for every $x \in \{0,1\}^n$, the configuration graph $G_{M,x}$ has at most $N = 2^{O(S(n))}$ vertices.
- $x \in L$ iff there is a directed path in $G_{M,x}$ from C_s to C_t.
- Recursive procedure REACH(u,v,i): Returns YES if there is a path from u to v of length at most 2^i; NO otherwise.
- Path from u to v of length at most 2^i iff there is a vertex z with an at most 2^{i-1} length path from u to z and an at most 2^{i-1} length path from z to v.

Savitch's Theorem: Proof

- In $O(\log N)$ space, REACH goes through all possible z's and returns YES iff it finds a z where REACH(u,z,i-1) = YES and REACH(z,v,i-1) = YES.
- Two recursive calls, but same space can be used.
- Recursion for space usage: $s_{N,i} = s_{N,i-1} + O(\log N)$. So $s_{N,\log M} = O(\log^2 N) = O(S(n)^2)$.

Savitch's Theorem: Corollary

$NL \subseteq L^2 = SPACE(\log^2(n))$.