Graph reachability

REACHABILITY is the following problem:

Given a graph G and vertices x and y, is there a path from x to y?

REACHABILITY $\in$ PTIME, in fact it can be solved in time $O(n^2)$; se chap. 1.1.

The reachability method

Thm 7.4: Let $f(n)$ be a proper complexity function. Then:

(a) $SPACE(f(n)) \subseteq NSPACE(f(n))$

and $TIME(f(n)) \subseteq NTIME(f(n))$.

(b) $NTIME(f(n)) \subseteq SPACE(f(n))$.

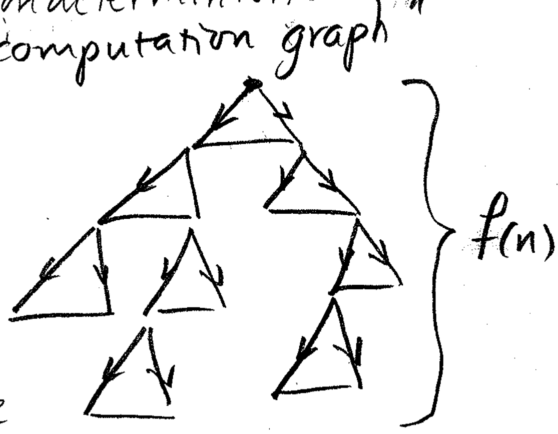
(c) $NSPACE(f(n))$

$$\subseteq TIME(k^{\log n + f(n)}).$$

Proof idea: (a) immediate.

(b) Left side says that every computation is bounded in length (of time) by $f(n)$. Thus we never need more space than $f(n)$ if we reuse the space of the previously tried computation.
(compare with proof of Thm. 2.6)

nondeterministic "computation graph"



(c) Look at left side (of '$\leq$').

Suppose the NTM M decides L within space $f(n)$, so for any input x with $|x| = n$, no more space than $f(n)$ is needed.

For some c_0 depending only on M there are at most $n \cdot c_0^{f(n)}$ configurations (if $|x| = n$) such that all strings are bounded by $f(n)$

and $n \cdot c_0^{f(n)} = c_0^{\log_{c_0} n + f(n)} =$

$$= c_0^{\frac{\log_2 n}{\log_2 c_0} + f(n)} = c_1^{\log n + f(n)},$$

where $c_1 = c_0^{(\log_2 c_0)^{-1}}$ and '$\log$' = '$\log_2$'.

Hence, the computation/configuration graph has at most $c_1^{\log n + f(n)}$ vertices.

For each accepting configuration in the graph we need at most time $c_2\left(c_1^{\log n + f(n)}\right)^2 = \left(c_2 c_1^2\right)^{\log n + f(n)}$ to check whether it is reachable from the input configuration. (since REACHABILITY $\in$ TIME$(c_2 n^2)$).

Doing this for every accepting config. (among at most $c_1^{\log n + f(n)}$) gives the time bound

$$c_1^{\log n + f(n)} \cdot \left(c_2 c_1^2\right)^{\log n + f(n)}$$

$\leq c^{\log n + f(n)}$ for appropriately choosen c. $\square$

<u>Notation</u>:

$L \stackrel{def}{=} SPACE(\log n)$

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

<u>Corollary (to 7.4)</u>:

$L \subseteq NL \subseteq P \subseteq NP \subseteq PSPACE$

(because $NL \stackrel{def}{=} NSPACE(\log n) \stackrel{(c)}{\subseteq} TIME(k^{\log n + \log n}) = TIME(n^c)$ where c depends on k, and $NP \subseteq PSPACE$ is immediate from (b)).

Thm 7.5 (Savitch's thm):

REACHABILITY $\in$ SPACE$(\log^2 n)$.

Proof idea: For a graph G and vertices x and y, and $i \geq 0$, check wherer there is a path from x to y of length at most 2^i.

The algoritm checks this recursively for $i = 0, \ldots, \lceil \log n \rceil$ where $\{1, \ldots, n\}$ is the set of vertices of G.

Moreover, we need only keep track of $\lceil \log n \rceil$ triples of the form (x, y, i) where $x, y \in \{1, \ldots, n\}$ and i are represented in binary (hence of length at most $\lceil \log n \rceil$). $\square$

Important

Corollary: For every proper complexity function $f(n) \geq \log n$,

$$NSPACE(f(n)) \subseteq SPACE(f^2(n)),$$

and consequently $PSPACE = NPSPACE$.

Proof. Given an $f(n)$-space NTM M, its config. graph has size at most $c^{\log n + f(n)} \leq c^{2f(n)}$ and we can apply the deterministic $\log^2 n$-space algoritm to it, and simulate M deterministically (i.e try all possibilities) in space bounded by $(\log c^{2f(n)})^2 \leq k f^2(n)$ for some k depending on c. $\square$

<u>Def.</u> A function F from strings to strings is said to be <u>computed</u> by an <u>N</u>TM M if for every input x:

(1) at least one computation halts with output $F(x)$.

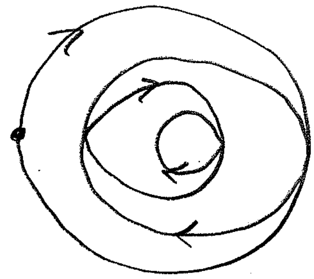
(2) if a computation does not halt with output $F(x)$, then it halts with output "no".

We say that it operates within space $f(n)$ if the length of any string, except the input and output strings, never exceeds $f(|x|)$ on input x.

Thm 7.6 (Immerman–Szelepscényi)

Given a graph G and a vertex x, the number of vertices that are reachable from x (in G) can be computed by an NTM within space $\log n$.

The proof is quite down-to-earth but a bit technical, involving 4 nested loops:



$\square$

<u>Notation</u>: coNSPACE $\stackrel{\text{def}}{=}$ $\{L : \bar{L} \in \text{NSPACE}\}$.

<u>Corollary</u> If $f(n) \geq \log n$ is a proper complexity function, then NSPACE$(f(n))$ = coNSPACE$(f(n))$.

<u>Proofidea</u>: If an NTM M decides L in space $f(n)$, then consider the configuration graph of M which has size $\leq c^{\log n + f(n)} \leq c^{2f(n)}$. Using Thm 2.6, the reachability problem can be solved, nondeterministically, in space $\log(c^{2f(n)}) \leq k\,f(n)$ for appropriate k. But the same is true for the "non-reachability" problem, since space $k f(n)$ is sufficient for checking whether input x can lead to "yes". $\Box$

Reductions

<u>Def. 8.1</u> Language L_1 is <u>reducible</u> to language L_2 if there is a function R from strings to strings such that R is computable by a TM in space $O(\log n)$ and for all x,

$$x \in L_1 \iff R(x) \in L_2 .$$

We call such R a <u>reduction</u> from L_1 to L_2.

<u>Prop 8.1</u>: If R is a reduction computed by a TM M, then for some k, M halts after at most $|x|^k$ steps, for every input x.

<u>Proof idea</u>: M has at most $O(|x| c^{\log |x|}) = O(|x| c^{d \log_c |x|}) = O(|x| \cdot |x|^d)$ configurations on input x and a config. is never repeated in the same computation. $\square$

<u>Prop 8.2</u>: If R_1 is a reduction from L_1 to L_2 and R_2 is a reduction from L_2 to L_3, then $R_2 \circ R_1$ is a reduction from L_1 to L_3.

The proof is not as trivial as one may think, because $|R_1(x)|$ may be greater than $\log|x|$. Therefore one needs to do a careful analysis of how to transmit information about $R_1(x)$ to R_2 without violating the space bound $\log|x|$.

Completeness

Def. 8.2: Le $\mathcal{C}$ be a complexity class. A language $L \in \mathcal{C}$ is said to be <u>$\mathcal{C}$-complete</u> if every $L' \in \mathcal{C}$ is reducible to L.

Def. $\mathcal{C}$ is <u>closed under reductions</u> if whenever L is reducible to $L' \in \mathcal{C}$ then $L \in \mathcal{C}$.

Prop 8.3: P, NP, coNP, L, NL, PSPACE and EXP are closed under reductions.

Proof: Exercise. $\square$

Prop 8.4: If $\mathcal{C}$ and $\mathcal{C}'$ are closed under reductions and $L \in \mathcal{C} \cap \mathcal{C}'$ is $\mathcal{C}$-complete and $\mathcal{C}'$-complete, then $\mathcal{C} = \mathcal{C}'$.

Propositional logic

We can think that $1 =$ true, $0 =$ false.
A function $f : \{0,1\}^n \to \{0,1\}$ is called Boolean.

Fact: For every Boolean function $f(x_1, \ldots, x_n)$ there is a propositional formula φ_f, containing only the connectives $\neg, \wedge, \vee$ and with propositional variables $p_1, \ldots, p_n$, such that for every truth assignment $(s_1, \ldots, s_n) \in \{0,1\}^n$,

$$f(s_1, \ldots, s_n) = 1 \iff$$

φ_f is true when p_i is assigned the value s_i for $i = 1, \ldots, n$.

Thus every function

$$g : \{0,1\}^n \to \{0,1\}^m$$

can be represented by a conjunction

$\varphi_{f_1} \wedge \dots \wedge \varphi_{f_m}$, where

φ_{f_i} represents the Boolean function

$f_i(x_1, \dots, x_n)$ = the value of the ith coordinate of $g(x_1, \dots, x_n)$.

Cook's theorem

SAT is the problem of deciding whether a propositional formula is satisfiable.

SAT $\in$ NP, because we can guess a truth assignment for a prop. formula φ and check in polynomial time if it makes φ true.

I now try to give the idea of how every $L \in NP$ can be reduced to SAT (so SAT is NP-complete).

First, we note that for every $L \in NP$ there is a (1-string) NTM M deciding L in time n^k such that at every step M has <u>exactly two</u> nondeterministic choices.

Let's assume that M is such.

Then every possible computation is coded by a binary sequence of choices $\bar{c} = (c_1, c_2, \ldots, c_s)$, where $s < |x|^k$ and x is the input.

With every x and $\bar{c}$ we associate a <u>computation table</u> $T(M, x, \bar{c})$ where row i corresponds to the configuration after i steps, using the choice seq. $\bar{c}$, and the entry T_{ij} encodes, as a binary string,

- the jth symbol on the string (from the left)
- and if M scans position j at time i, then T_{ij} also encodes the current state.

Moreover T_{ij} only depends on

$$T_{i-1,j-1}, \quad T_{i-1,j} \text{ and } T_{i-1,j+1}$$

(the adjacent positions at the previous step).

As said, T_{ij} encodes the information as a <u>binary string</u>, of length m, say, where m depends only on M (its number of states and symbols).

Hence, all possible transitions, with the choice sequence $\bar{c}$ are represented as

$$(T_{i-1,j-1}, T_{i-1,j}, T_{i-1,j+1}, c_i) \mapsto T_{ij} \text{ and}$$

correspond to a function

$$g_{\bar{c}}: \{0,1\}^{3m+1} \longrightarrow \{0,1\}^m.$$

Then there is a prop. formula

$\varphi_{g_{\bar{c}}} = \varphi_{f_1} \wedge \dots \wedge \varphi_{f_m}$ where each

φ_{f_i} is as in the previous section (so f_i is the restriction of $g_{\bar{c}}$ to the ith coordinate).

We may assume that if the first bit in $T_{|x|^k,2}$ is 1 then M accepts x, and otherwise M does not accept x.

(If M halts after $l < |x|^k$ steps then rows $l+1, \dots, |x|^k$ are identical.)

Also we may assume that M has the feature that the input config. is $\triangleright \triangleright x$ where the rightmost '$\triangleright$' is scanned and that M never moves the cursor to the leftmost '$\triangleright$'.

Let's call the reduction from L to SAT which we want to construct R.

Inductively, for $l = n^k, n^k - 1, \ldots, 1$, where $n = |x|$, we define:

- $R^{\bar{c}}_{n^k}(x) = \varphi_{f_1}$.

Let us assume that the propositional variables of $\varphi_{g_{\bar{c}}}$ are $x_1, \ldots, x_{3m+1}$

- $R^{\bar{c}}_{l-1}(x)$ is the formula obtained from $R^{\bar{c}}_l(x)$ by, substituting, for $i = 1, \ldots, m$, φ_{f_i} for all occurences of x_i, x_{2i} and x_{3i}.

- Let $R^{\bar{c}}(x) = R^{\bar{c}}_1(x)$.

And finally, $R(x)$ is the disjunction of all $R^{\bar{c}}(x)$ where $\bar{c}$ ranges over all choice sequences of length $\leq |x|^k$ which end with M halting.

It remains to verify that R can be computed in space $\log n$ and (by induction) that $x \in L \iff R(x) \in SAT$ = the set of propositional formulas that are satisfiable. $\square$