Prove there is no path from s to t in graph G in log space.

Trick, force the nondeterministic machine to tell us how many vertices we can reach from s.

ck = the number of vertices we can reach from s in k edges or less.

C0 = 1 {s}

Suppose we know ck.

For each vertex (other than s), we determine if that vertex is in ck+1.

For vertex v,

For each vertex u, nondeterministically guess if u is in ck.

If the machine says “yes”, verify by having the machine give us the path.

We increment a count (d)

Check, is there an edge from u to v?

If so, we increment ck+1 and break from this inner loop

At the end, check that d = ck

Heirarchy Theorems

Idea (deterministic machines), we can solve more problems if we use more space or we use more time.

For “space constructable” function f, there exists a language that can be decided in space O(f(n)) but not in space o(f(n)).

A function is “space constructable” f is a function that takes 1n (the number of n 1’s) and produces f(n) represented in binary in space O(f(n)).

We will create a TM D that runs in space O(f(n)). We create it so that no machine that runs in o(f(n)) can decide the same language as D.

D: On input w.

1) check that w ends with 10\* (the idea is that what ever is before the 1 is the representation of TM) - if not, reject

2) Compute f(|w|) and set aside that much space on D’s work tape. (this will be used as the space for the TM that is input to D)

3) Consider w = <M>10\*, simulate M on w

4) If M ever uses more space than what we mark off, reject

5) If the simulation ever fails because <M> is not a valid TM, reject

6) If M halts, D responds with the opposite of M. If M accepts, D rejects, if M rejects, D accepts.

D runs in O(f(n)) space.

Assume there exists some M’ that runs in o(f(n)) space and L(M’) = L(D).

Run D on input <M’>10k for some k. (k is big enough so the size of the input is past the constant where o(f(n)) running space of M’ is smaller than O(f(n)) running space of D.

D will be able to simulate M’ on w = <M’>10k in space O(f(|w|))

D returns the opposite of M’. So M’ cannot have the same language as D.

What if I run D on <D>10k?

SPACE(n2) is a strict subset of SPACE(n3).

L is a strict subset of PSPACE.

We know that L is a subset of P and P is a subset of PSPACE. One of these (if not both) must be strict, but we don’t know which one.

Time hierarchy.

Let f be a time constructable function, (takes 1n to binary representation of f(n) in O(f(n)) time), then there exists a language decidable in time O(f(n)) and not decidable in time o(f(n) / log f(n)).

Create D, such that D runs in O(f(n)) time but no machine running in time o(f(n) / log f(n)) can decide the same language as D.

D: One input w.

1) Check that w is of the form <M>10\*

2) Compute f(|w|) and store that value as C

3) Simulate M on w, for each step of M, decrease C.

4) If the simulation ever fails because <M> is not a valid TM, D rejects.

5) If C reaches 0 without M halting, D rejects

6) When M halts, D accepts if M rejects, D rejects if M accepts.

P is a strict (proper) subset of EXPTIME.

NTIME(n) strict subset of NTIME(n2)