Savitch's Theorem

Theorem: Let $f: N \rightarrow R$ be a function, with $f(n) \geq n$. Then,

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

Proof: Suppose language A can be decided by an NTM in $k\, f(n)$ space, for some constant k. We shall show that it can be decided by a DTM in $O((f(n))^2)$ space

Savitch's Theorem (2)

… A naïve approach is to simulate all branches of the NTM's computation, one by one, using DTM. To do so, we need to keep track of which branch we are testing (that is, the choices made in each branch).

- Unfortunately, a branch in the NTM may have $2^{O(f(n))}$ steps (though it uses $O(f(n))$ space), so that we may need $2^{O(f(n))}$ space… NOT GOOD…

Savitch's Theorem (3)

… Instead, we solve the yieldability problem, such that given two configurations c_1 and c_2 of the NTM N, we want to decide whether c_2 can be yielded from c_1, in some number of steps

For this purpose, let us define a recursive function, called CAN_YIELD(c_1,c_2,t), the checks if c_1 can yield c_2 in t steps as follows (next slide)

Function CAN_YIELD(c_1,c_2,t) {

1. If $t = 1$, test whether $c_1 = c_2$ or whether c_1 yields c_2 in one step using the rule of NTM N. Accept if either test succeeds; Reject otherwise.
2. For each config c_m using $k\,f(n)$ space:
   a. Run CAN_YIELD(c_1,c_m,$t/2$)
   b. Run CAN_YIELD(c_m,c_2,$t/2$)
   c. If both accept, accept
3. If haven't accept yet, reject

}

Savitch's Theorem (4)

We modify N a bit, and define some terms:

- We modify N so that when it accepts, it clears the tape and moves the tape head to leftmost cell. We denote such a configuration c_{accept}
- Let c_{start} = start configuration of N on w
- Select a constant d such that N has at most $2^{d\,f(n)}$ configurations (which is the upper bound of N's running time)

Savitch's Theorem (5)

Based on this new N, there exists a DTM M that simulates N as follows:

M = "On input w,

1. Output the result

   CAN_YIELD(c_{start}, c_{accept}, $2^{d\,f(n)}$) "

Question: What is space usage of M?

Savitch's Theorem (6)

- When CAN_YIELD invokes itself recursively, it needs to store c_1, c_2, t, and the configuration it is testing (so that these values can be restored upon return from the recursive call)
- Each level of recursion thus uses $O(f(n))$ space
- Height of recursion: $df(n) = O(f(n))$

➔ Total space = $O((f(n))^2)$