

Savitch's theorem

- Consider the following recursive procedure REACH. (Note: $d_G(s, t)$ denotes the length of the shortest path from s to t in G ; if no such path exists then $d_G(s, t) = \infty$).

Input: $G = (V, E)$, $s, t \in V$, $k \in \mathbb{N}$ (k a power of 2)
Output: TRUE if $d_G(s, t) \leq k$, FALSE otherwise

```

1: if  $k = 1$  then
2:   return TRUE iff  $(s, t) \in E$ 
3: else
4:   for all  $w \in V$  do
5:      $b_1 \leftarrow \text{REACH}(G, s, w, k/2)$ 
6:      $b_2 \leftarrow \text{REACH}(G, w, t, k/2)$ 
7:     if  $b_1 \wedge b_2$  then
8:       return TRUE
9:     end if
10:  end for
11:  return FALSE
12: end if

```

Algorithm 1: REACH

- It is not hard to see that REACH satisfies its specification: that is, $\text{REACH}(G, s, t, k)$ returns true iff $d_G(s, t) \leq k$. The idea is that if there is a path from s to t of length k or less, then the middle vertex w on the path satisfies $d_G(s, w) \leq k/2$ and $d_G(w, t) \leq k/2$. By iterating through all vertices w , we will eventually discover the middle vertex and accept. Conversely if $d_G(s, t) > k$ then there is no vertex w satisfying $d_G(s, w) \leq k/2$ and $d_G(w, t) \leq k/2$, so the algorithm will reject. (Note: the correctness can be proved formally by induction on k).
- Let $S(n, k)$ denote the space used by $\text{REACH}(G, s, t, k)$ when $|V| = n$.
- Then $S(n, 1) \leq c_1 \log n$ for some constant c_1 : for example, if G is represented by its adjacency matrix then we can check whether $(s, t) \in E$ by examining the bit at row s and column t in the matrix. To do this requires counting the row r and the column c , and since $r, c \leq n$ we can represent r and c using $\log n$ bits each.
- What about $S(n, k)$ for $k > 1$? The overhead of the “for” loop is $O(\log n)$ bits: there are n vertices, so writing down one vertex w requires $\log n$ bits, and this space can be re-used in subsequent iterations. Since the graph G remains the same in the recursive calls, there is no need to write it down again: thus to implement a recursive call we only need to store two vertices (requiring $\log n$ bits each), and the number $k/2$, which requires fewer than $\log k$ bits. We may also assume that $k \leq 2n$ (the 2 comes

from our assumption that k is a power of 2), so the space required is $O(\log k + \log n) = O(\log n)$ plus the space required for the recursive calls. By definition, the space required by a recursive call is $S(n, k/2)$. The key observation is that we can re-use this space for the second recursive call (we only need to remember the one-bit output b_1 from the first call). Thus we have the following recurrence for $S(n, k)$:

$$S(n, k) \leq \begin{cases} c_1 \log n, & \text{if } k = 1 \\ c_2 \log n + S(n, k/2), & \text{otherwise} \end{cases}$$

where c_1 and c_2 are constants.

- **Lemma:** $S(n, k) \leq c_1 \log n + c_2 \log n \log k$
- **Proof:** By induction on k . For $k = 1$ the lemma holds by definition of $S(n, k)$. Let $k \geq 2$ and assume that $S(n, k/2) \leq c_1 \log n + c_2 \log n \log(k/2)$. Then

$$\begin{aligned} S(n, k) &\leq c_2 \log n + S(n, k/2) \\ &\leq c_2 \log n + c_1 \log n + c_2 \log n \log(k/2) \\ &= c_2 \log n + c_1 \log n + c_2 \log n (\log k - 1) \\ &= c_2 \log n + c_1 \log n + c_2 \log n \log k - c_2 \log n \\ &= c_1 \log n + c_2 \log n \log k \end{aligned}$$

- We can now prove Savitch's theorem:
- **Theorem:** If $S(n) \geq \log n$, then $\text{NSPACE}(S) \subseteq \text{DSPACE}(S^2)$.
- **Proof:** Let M be an NTM using space $S(n)$. Without loss of generality, assume that M has a unique accepting configuration C_{accept} (we can always modify M so that it erases everything on its work tape and moves its tape heads back to their initial positions before accepting). We will describe a deterministic TM M' that decides the same language as M and runs in space $S(n)^2$. Let $w \in \Sigma^*$ be an input of length n . Consider the configuration graph $G = (V, E)$ of M on input w . The number of configurations that M can obtain on input w is bounded by

$$|Q| \cdot n \cdot S(n) \cdot |\Gamma|^{S(n)}$$

as a configuration is specified by a state (of which there are at most $|Q|$), the input-tape position (at most n), the work-tape position (at most $S(n)$), and the contents of the work tape (at most $|\Gamma|^{S(n)}$). Rewrite this as:

$$\begin{aligned} 2^{\log |Q|} \cdot 2^{\log n} \cdot 2^{\log S(n)} \cdot 2^{S(n) \log |\Gamma|} &= 2^{\log |Q| + \log n + (|\Gamma| + 1) \log S(n)} \\ &= 2^{O(S(n))} \end{aligned}$$

The last line follows because we assumed $\log n \leq S(n)$, and $|Q|$ and $|\Gamma|$ are constant. Now if $w \in L(M)$ then there exists a path from $I_M(w)$ to

C_{accept} in G , and if $w \notin L(M)$ then there is no path from $I_M(w)$ to C_{accept} in G . The deterministic machine M' will determine if there is a path from $I_M(w)$ to C_{accept} , and accept iff there exists such a path. Clearly if there is such a path, then there is one with length at most $|V|$. Let k be the smallest power of 2 such that $k \geq |V|$ (note that $k \leq 2|V|$), and run $\text{REACH}(G, I_M(w), C_{accept}, k)$, accepting iff the output is TRUE. Then it is clear that M' accepts iff $w \in L(M)$, so we have $L(M') = L(M)$. The space used by the call to REACH is at most

$$\begin{aligned} S(|V|, 2|V|) &= \log 2^{O(S(n))} \cdot \log 2^{O(S(n))} \\ &= O(S(n)^2) \end{aligned}$$

- There is one slight problem with the above: we cannot actually write down the configuration graph G , since it has $2^{O(S)}$ vertices and we are trying to work in space $O(S^2)$. However, looking at the definition of REACH, it is enough to be able to decide, for two vertices u, v , whether $(u, v) \in E$: that is, instead of writing down the entire graph G , we can compute the parts which we are interested in “on the fly”, as the need arises. To determine whether (u, v) is an edge, we only need to verify that the configuration v can be reached from u with one application of M 's transition function.