Programing Techniques

- Use states to remember a symbol:
  - Take $Q' := Q \cup (\Gamma \times Q)$;
  - similarly: remember k symbols, $k \in \mathbb{N}$ fixed (!)
- two/three string tape:
  - Take $\Gamma' := \Gamma \cup (\Gamma \times \Gamma \times X)$
  - similarly: k-string tape

1	0	1	0	0	1	…
1	1	0	0	1	1	…
a	x	r	s	b	y	…

- subroutine calls

TMs may be inconveient to program yet are capable of anything a digital computer can do – and that even surprisingly fast [Schönhage et.al. 1994]

Linear Speed-Up

Why big-*Oh* notation?

Theorem: Let $\mathcal{M}$ be $t(n)$-time bounded DTM and $k\in\mathbb{N}$. There exists a $\big(t(n)/k+O(n^2)\big)$-time bounded DTM 'simulating' $\mathcal{M}$.

„Speedup by any constant factor...“

- Proof (sketch): combine c consecutive tape cells into a single one: $\Gamma\to\Gamma^c$; then:
- combine c original steps into a single one.

Resource: Space

Let $\mathcal{M}=(Q,\Sigma,\Gamma,\delta)$ denote a DTM. The length $|K|$ of a configuration $K = \underline{\alpha}\ q\ \underline{\beta}$ is defined as $|\underline{\alpha}|+|\underline{\beta}|$

- For $\underline{w}\in\Sigma^*$ let $S_{\mathcal{M}}(\underline{w})$ denote the number of tape cells $\mathcal{M}$ 'touches' on input $\underline{w}$: $S_{\mathcal{M}}(\underline{w}):=\max |K_i|$, where $K_0,K_1,\ldots$ denotes the (sequence of) configurations $\mathcal{M}$ attains on $\underline{w}$; possibly $S_{\mathcal{M}}(\underline{w})=\infty$.
- For $n\in\mathbb{N}$ let $S_{\mathcal{M}}(n):=\max\{S_{\mathcal{M}}(\underline{w}) \mid \underline{w}\in\Sigma^{\leq n}\}$ denote the space $\mathcal{M}$ uses on inputs of length $\leq n$; $S_{\mathcal{M}}:\mathbb{N}\to\mathbb{N}$ space consumption function
- $S_{\mathcal{M}}(n)\leq O(s(n))$: $\mathcal{M}$ is $O(s(n))$-space bounded
- DSPACE$(s(n))$:= { $L=L(\mathcal{M})$ for $\mathcal{M}$ $O(s(n))$—space bounded DTM}

Time versus Space

- Focus often on running time; but:
- *„Time is unbounded, memory is not"*

- $|\underline{w}| \leq S_{\mathcal{M}}(\underline{w}) \leq \max\{T_{\mathcal{M}}(\underline{w}), |\underline{w}|\}$
- Exercise 5b: Any DTM $\mathcal{M}$ can be simulated by a DTM $\mathcal{N}$ such that $T_{\mathcal{N}}(\underline{w}) \leq 2^{O(S_{\mathcal{M}}(\underline{w}))}$
- Theorem [Hopcroft,Paul,Valiant'73]: $\mathcal{M}$ can be simulated by $\mathcal{N}$ where $S_{\mathcal{N}}(n) \leq O(T_{\mathcal{M}}(n)/\log T_{\mathcal{M}}(n))$

2.3 Classes P and PSPACE

Def: $\mathbf{P} := \bigcup_k \mathrm{DTIME}(n^k)$

$\mathbf{PSPACE} := \bigcup_k \mathrm{DSPACE}(n^k)$

1. Superpolynomial growth usually becomes impractical already for modest input sizes
2. whereas polynomial running times are usually those tractable in practice.
3. **P** is a robust class, arising also from *k*-tape DTMs, *register machines* or *Java programmes*

So far only decision problems,
i.e. functions $f:\Sigma^*\to\{0,1\}$; later (Exercise 7):

Def: Computing functions (**FP**) $f:\Sigma^*\to\Sigma^*$

Preliminaries: Graphs and Coding

- A *directed* graph $G=(V,E)$ is a set V (elements called *vertices*) and $E \subseteq V \times V$ (set of *edges*)
- G is *undirected* if $(u,v) \in E \Leftrightarrow (v,u) \in E$
- Function $c:E \to \mathbb{N}$ assigning *weights* to edges.

For input to a Turing machine:

- Encode (G,c) as adjacency matrix $A \in \mathbb{N}^{V \times V}$
  - $A[u,v] := c(i,j)$ for $(u,v) \in E$,
  - $A[u,v] := *$ for $(u,v) \notin E$
- Case directed G: only upper triangular matrix.
- Let $\langle G,c \rangle$ denote this coding; $|\langle G,c \rangle| \geq |V|$