Turing Machines (Alan Turing 1936)
- single tape: moveL moveR, read, write
  each constant time
  constant pointer memory
  tape infinite (extra memory)

Von Neumann Architecture (Von Neumann + Eckert + Mauchly 1945)
- based on ENIAC
- CPU + Memory (RAM): read, write, op = constant time

----------

Scanning (max)
- TM : $O(n)$
- VNA: $O(n)$

Sorting
- TM : $O(n^2)$
- VNA: $O(n \log n)$

Searching
- TM : $O(n)$
- VNA: $O(\log n)$

how big is $\log n$, $n$, $n \log n$, $n^2$ :

$10^x$ | 1   2   3   4   5   6   7   8   9
search | 0.000001 0.000001 0.000001 0.000002 0.000001 0.000002 0.000002 0.000007 0.001871
MAX    | 0.000003 0.000005 0.000006 0.000048 0.000387
Gradations:

LOG | poly log (n) : log^c (n)
P   | poly (n)      : n^c
   -- NP --
EXP | exp (n)       : c^n

Theory:
- LOG not studied much since count loading of data
- P is poly (n). Lots of neat algorithms.
  Sometimes constant c (in n^c) important, sometimes not.
- EXP usually hopeless, but 1.000001^n is ok.
- NP : verify solution in P, find solution conjectured EXP.
  if EXP number of (parallel) machines -> in P. (bits of solution argument)

Matricies

Vector v = \[v_1 v_2 \ldots v_n\]^T
  u = \[u_1 u_2 \ldots u_m\]^T

Dot Products
\[
\langle v, u \rangle = v \cdot u = v^T u \\
= \sum_{i} u_i \cdot v_i \\
\text{(need } m = n) \\
\Theta(n)
\]

\[v u^T = R \quad \text{[n x m] matrix.} \]
\[R_{i,j} = v_i \cdot u_j \]
\[\Theta(n^2) \]

\text{Matrix Multiply:}
\[R = [n \times m] \text{ and } T = [m \times k] \text{ matrices} \]
\[RT = U \quad \text{a [n x k] matrix} \]
\[U_{i,j} = \langle R_{i,*}, T_{*,j} \rangle = R_{i,*} T_{*,j} \]
\[O(n^3) \rightarrow O(n^{2.807}) \quad \text{[Strassen 69]} \rightarrow \ldots O(n^{2.376}) \]
\[[\text{Coppersmith Winograd 90}] \]
\[\Omega(n^2) \]

------------------------

\text{Probability:}

\text{Let } A, B \text{ be random variables.}

\[\Pr[A] \cdot \Pr[B] = \Pr[A \text{ and } B] \text{ off } A \text{ and } B \text{ are independent.} \]

\[\Pr[A \text{ and } B] < \Pr[A] + \Pr[B] \quad \text{"Union Bound"} \]

\text{Expected value } A = E[A] = \sum_{a \in U} a \cdot \Pr[a = A] \]
\[E[A] + E[B] = E[A + B] \quad \text{"Linearity of Expectation"} \]

------------------------
Hash Functions:

$h : U \rightarrow [n]$ 

$U := \text{set of possible inputs, maybe } [m], \text{ maybe } [a-z,A-Z]^{28}$  

$[n] := \text{output universe}$

$H = \text{family of hash functions.}$

If $H *\text{universal*}$ for $x \neq y$ then $\Pr_{h \in H}[h(x) = h(y)] \leq 1/n$

Simple example

$h_{a,b}(x) = ((a \times b) \mod p) \mod n$

where $a$ in $[1,p]$ and $b$ in $[0,p]$, both at random, and $p > m$ and prime.

Multiply-Shift hashing (Dietzfelbinger 97)

$\text{high-order-bits}(h_a(x) = (a \times \mod 2^w), N) \quad // \text{top M bits of first arg}$

where $a < 2^w$ (odd, at random), $w := \text{number of bits in machine word, } n = 2^N$