# Data structures in Information Retrieval

**Basic structures and environments**



RuSSIR
Russian Summer School
in Information Retrieval
2008

# Agenda

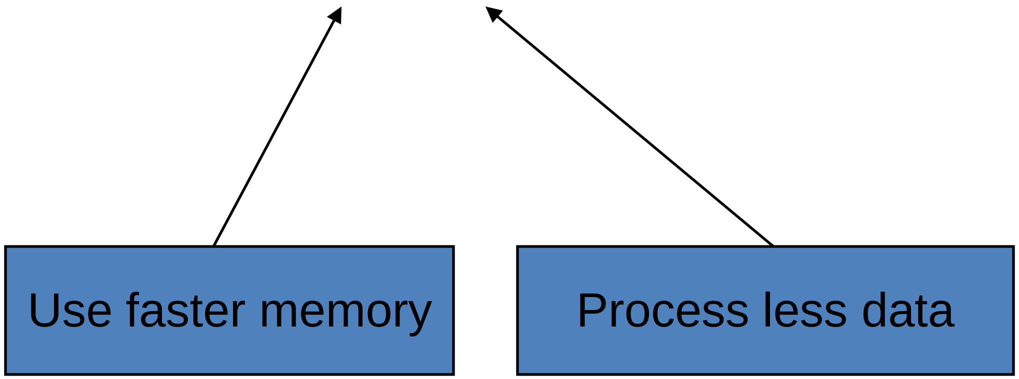
- How to create an effective structure
- Modern hardware
- Compression
- Search structures
- Parallelization

# General approach

1. Think about data usage
2. Select algorithm (s)
3. Select structure (s)
4. Select right libraries

# Effective data representation

1. Less memory resources
2. Faster execution



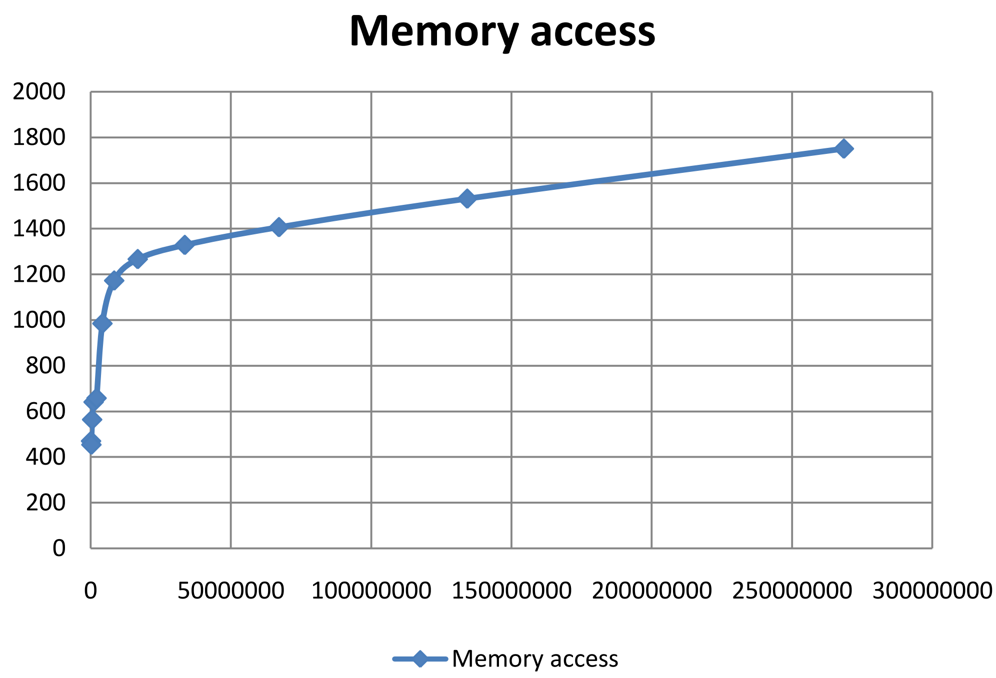
# Modern Hardware

- Moore law
- Multilayer memory
- High parallelism

# Memory hierarchy

| | |
|---|---|
| | **CPU registers** |
| | **CPU cache** |
| NUMA | RAM |
| NUMA | DISK |
| NUMA | Network/CD/Tape |

# RAM access

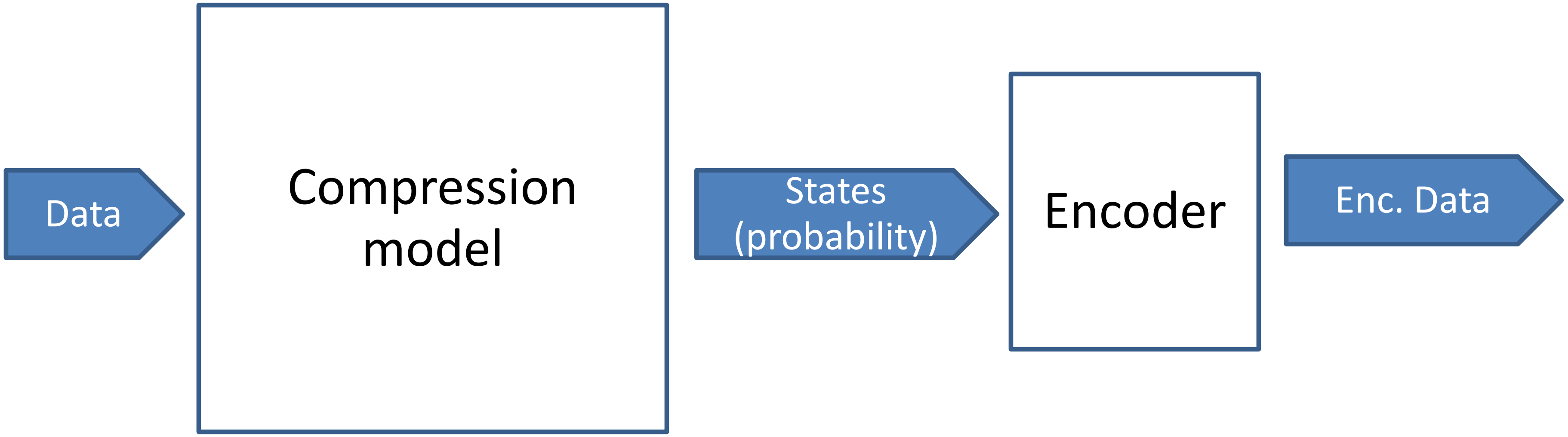



```
unsigned val = 0;
for(unsigned k = 0;k<1024*1024*8;++i){
   pos = rand()%memSize;
   unsigned t = arr[pos];
   arr[pos] = val;
   val = t;
}
```

# Storing data

- Data (documents): text + markup
- Index structures: words, positions,flags

# Data encoding and compression



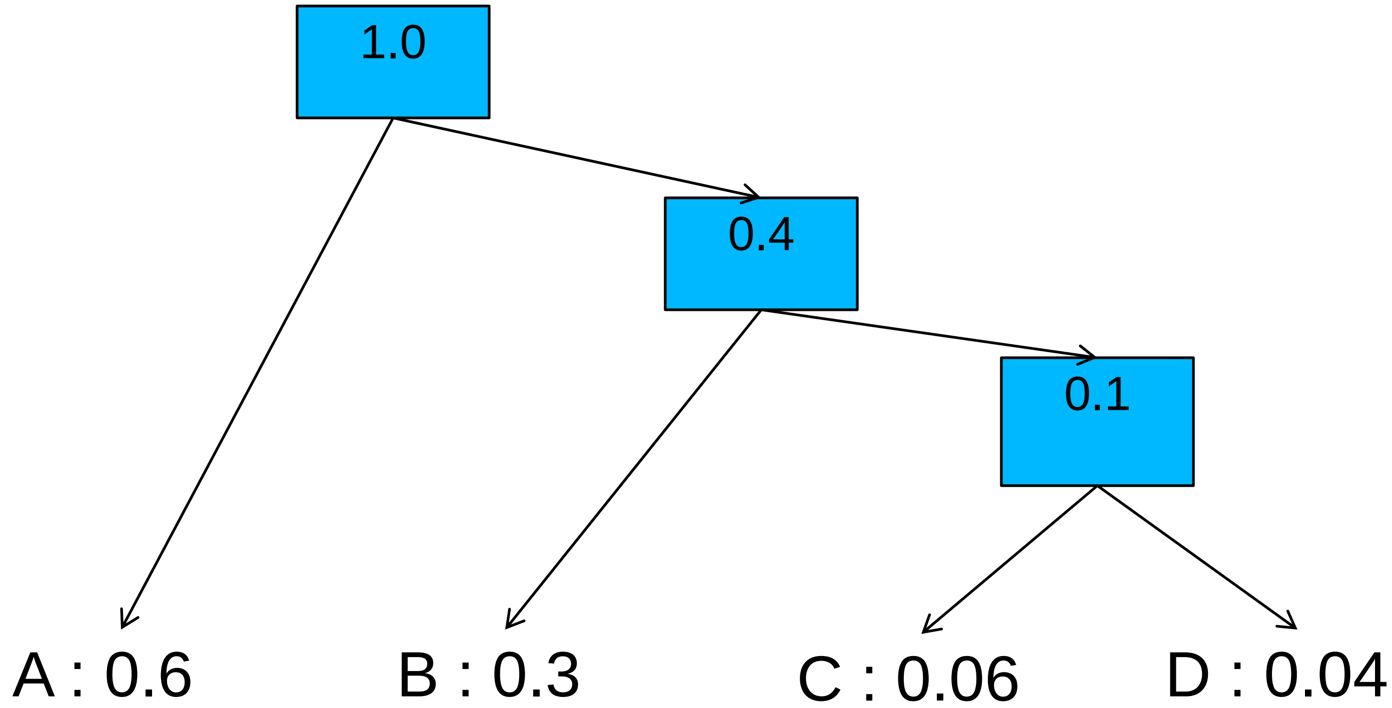
# Entropy coders

Input: Codes from alphabet A

probabilities distribution $a \rightarrow P_a$

Output: minimal length encoding (minimal number of bits)

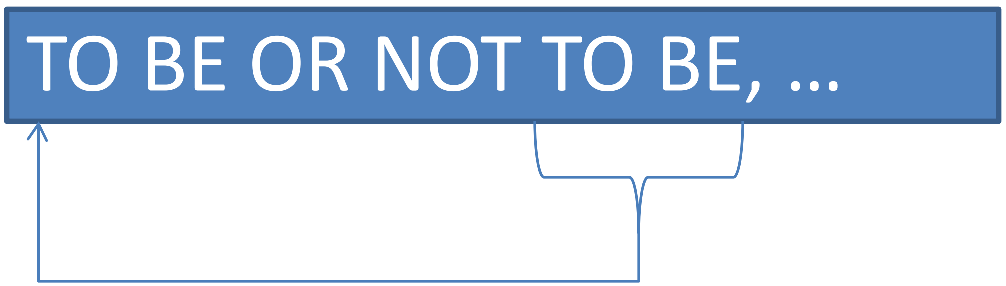
# Huffman coding




A <0> B <10> C <111> D <110>

# LZ encoding

Add special code “back reference”




‘T’,’O’,’ ‘,’B’,’E’,’ ‘,’O’,’R’,’ ‘,’N’,’O’,’T’,’ ‘,ESC,8,4

# Storing text for “toy” search Engine

- Use compression (read + LZW decompression)
- Compressed index+data < data

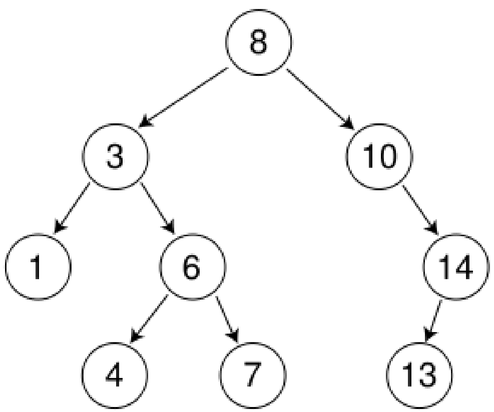
# Storing other data

Flags, field codes, etc. – entropy coder:

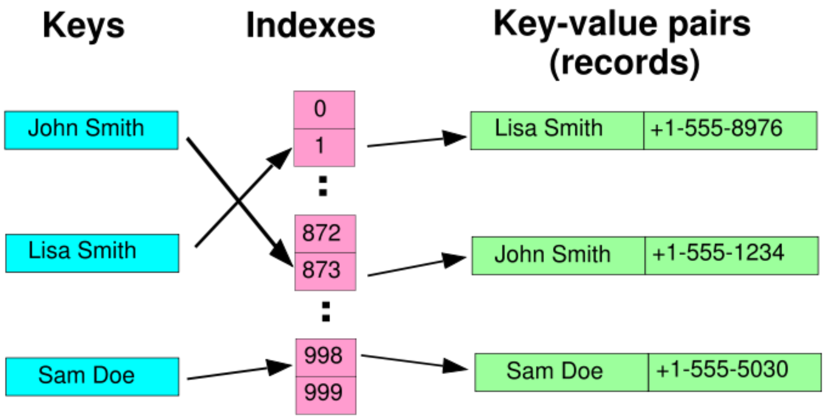
1. Collect statistics of this values
2. Select encoding
3. Store encoded in data and index

# Search structures

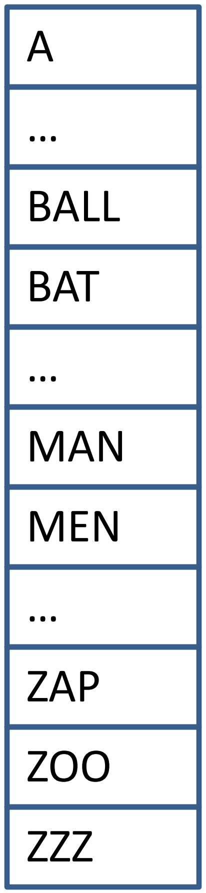
Branch & Bounds




Mappings



# Sorted arrays




Search: O(log(N))

Insert: O(N)

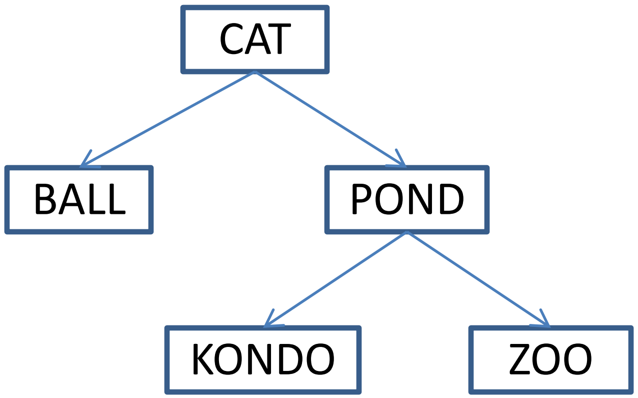
Advantage: most compact (no pointers)

Perfect for static dictionaries



RuSSIR
Russian Summer School in Information Retrieval
2008

# Binary tree

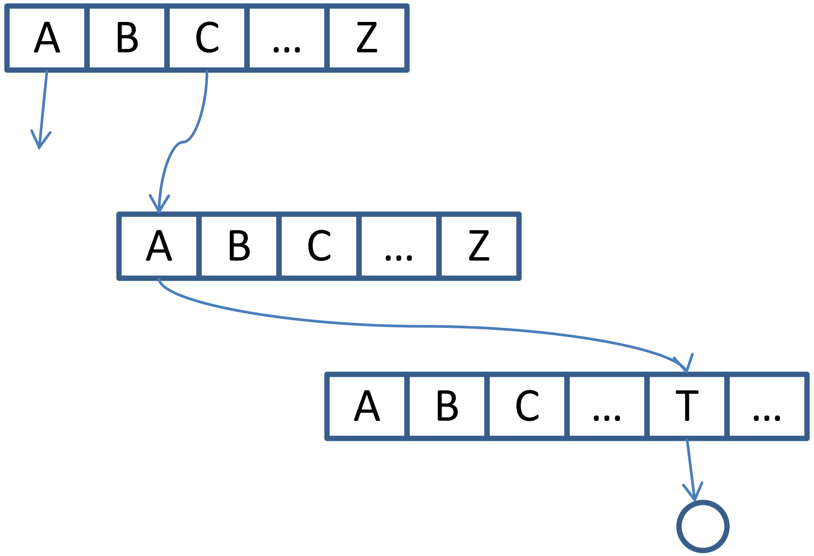



Search: O(log(N))

Insert: O(log(N))

Problems: pointers, memory allocations

# Trie

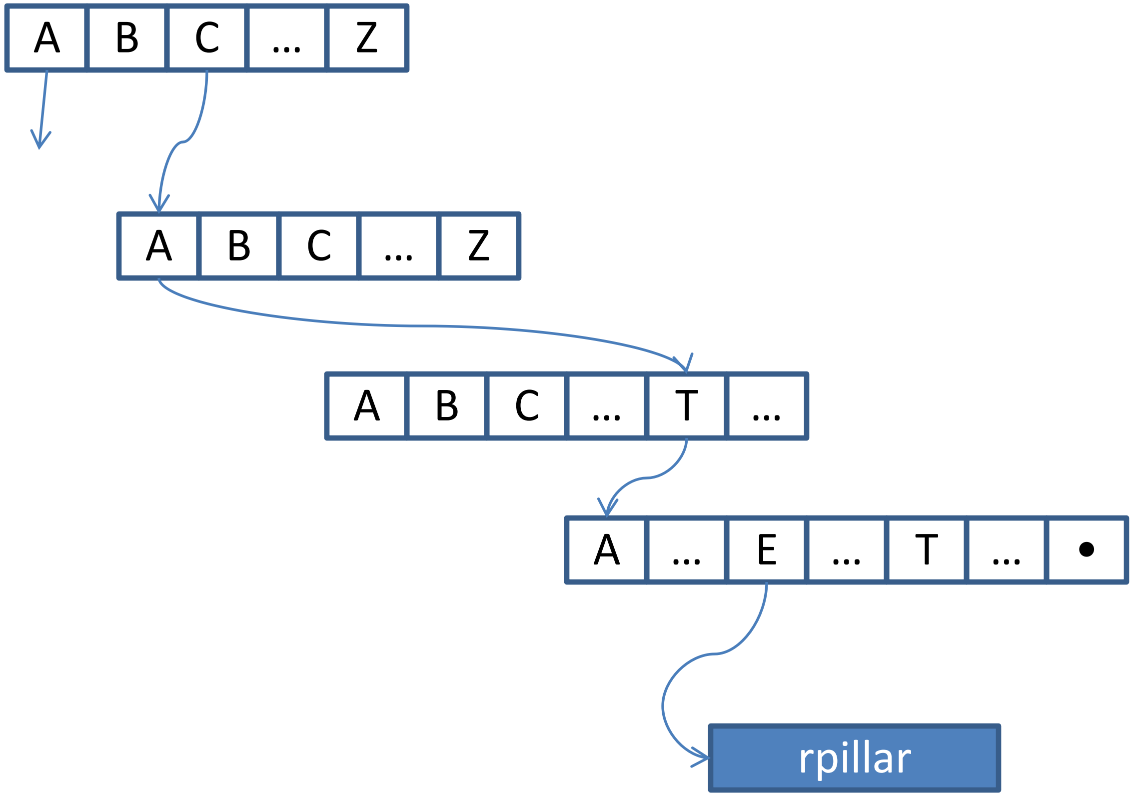



Search: O(Q)

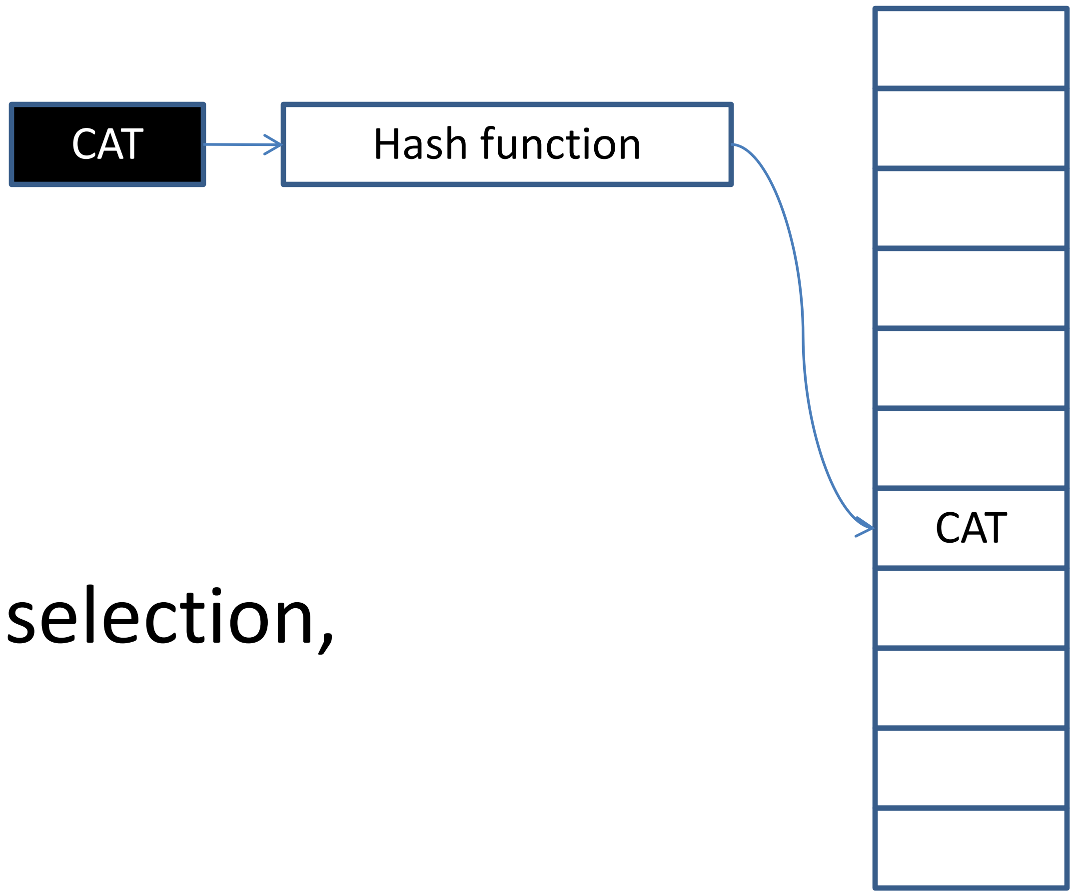
Insert: O(Q)

Problems: memory

# Compressed trie



# Hash



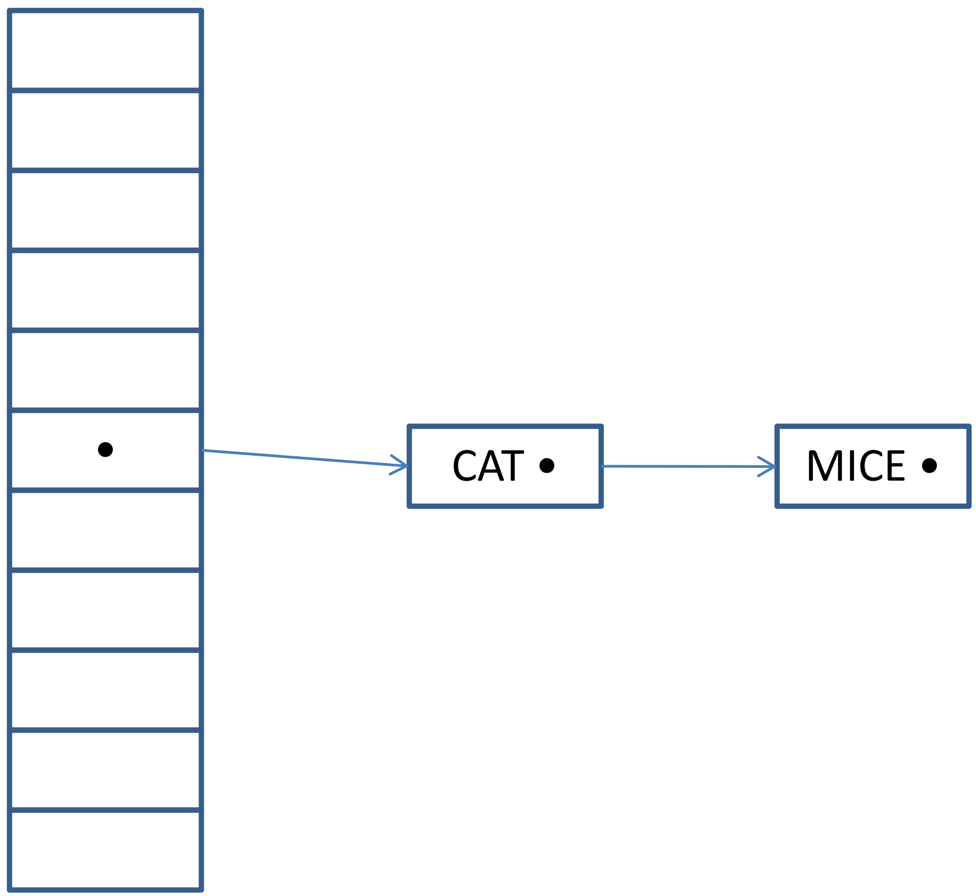

Search: O(1)

Insert:   O(1)

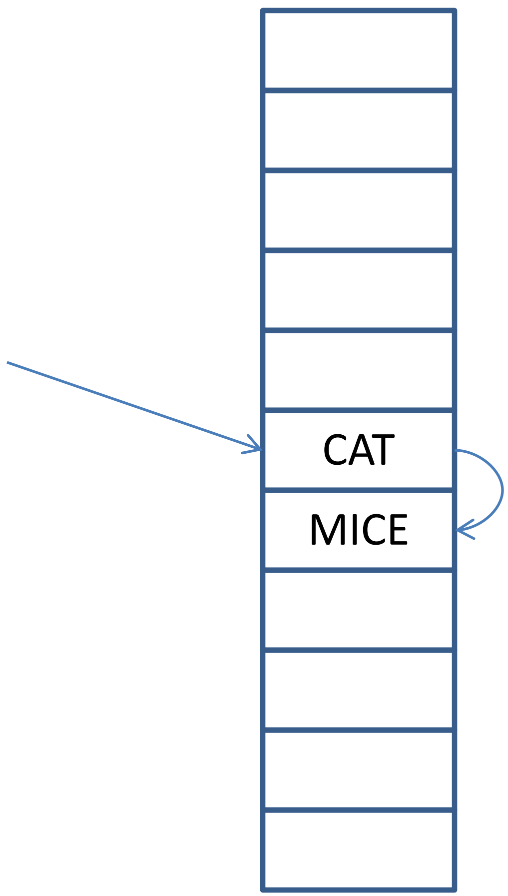
Problems: function selection,

size of array

# Hash overflow [lists]



# Hash overflow [rehashing]



# Bloom filter

Search-acceleration structure:

0 – if key doesn't exist

1 – there is probability P that key exists

# Example: speller

**Problem:**

- User input a word
- Suggest possible spell corrections

**Data Analysis:**

Generate correction modification rules

a→aa(0.000005);l→k(0.0006);ea→ee(0.0004)

**Algorithm:**

Apply up to 2 rules to a word and return variants that have higher probability

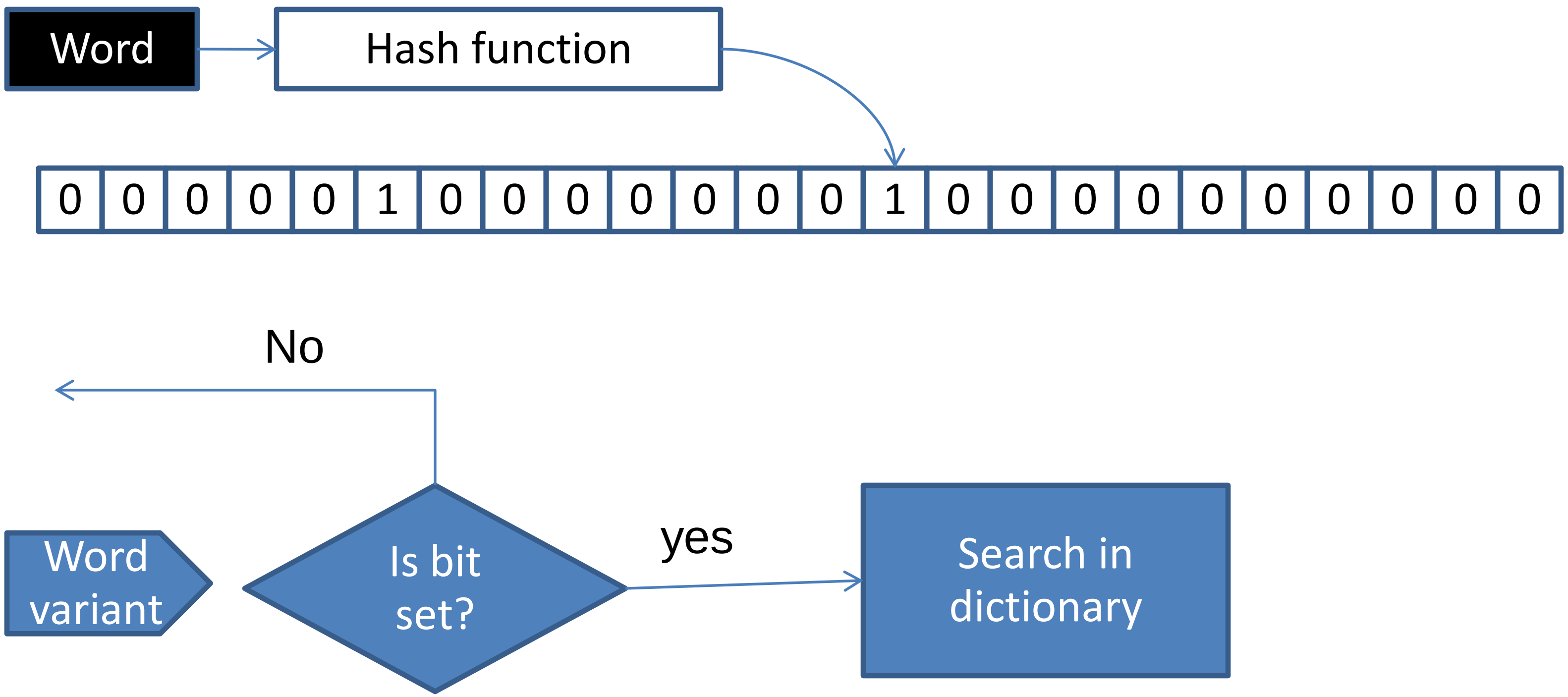
# Speller problem

For every word we can generate mlns variants, most of them – **not-valid words**

Search in big dictionary – relatively slow.

Idea – built a bloom filter that accelerates search

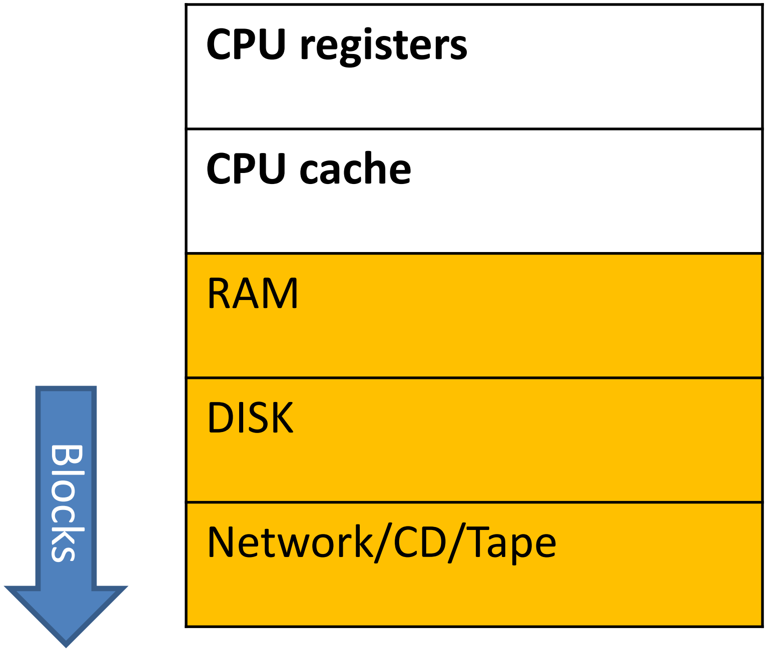
# Bloom filter for speller



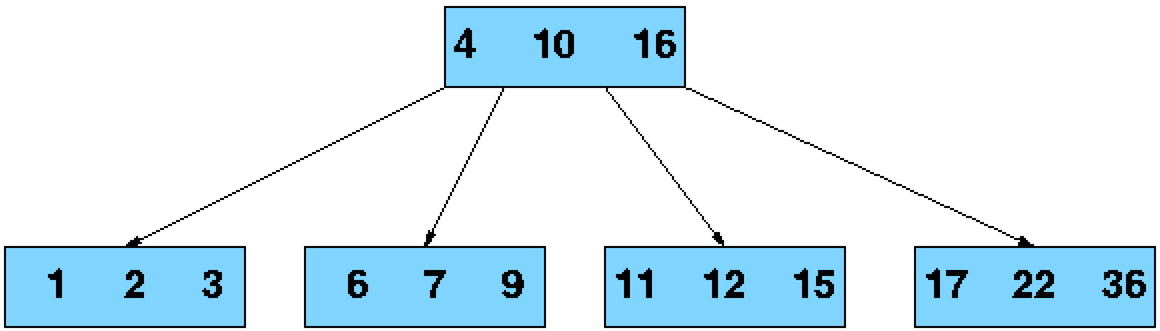

Increase speed of speller analysis 100 times

# Secondary-storage structures

- Block access
- Persistence
- Slow random access

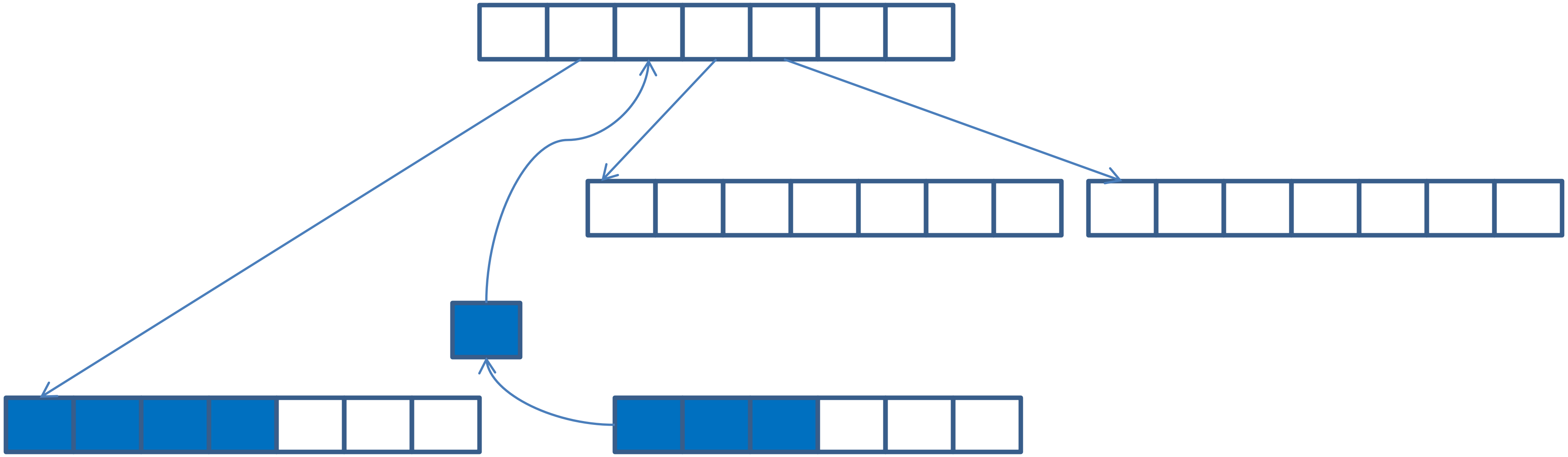


# B-Tree




- less pointers
- prefix-compressed nodes

# Insert in B-Tree



# Scaling your application

- **Vertical scaling –** moving to larger computer
- **Horizontal scaling** – dividing application into parts:

Two dimensions:

  - **Grouping data by functions**
  - **Grouping functions by data**

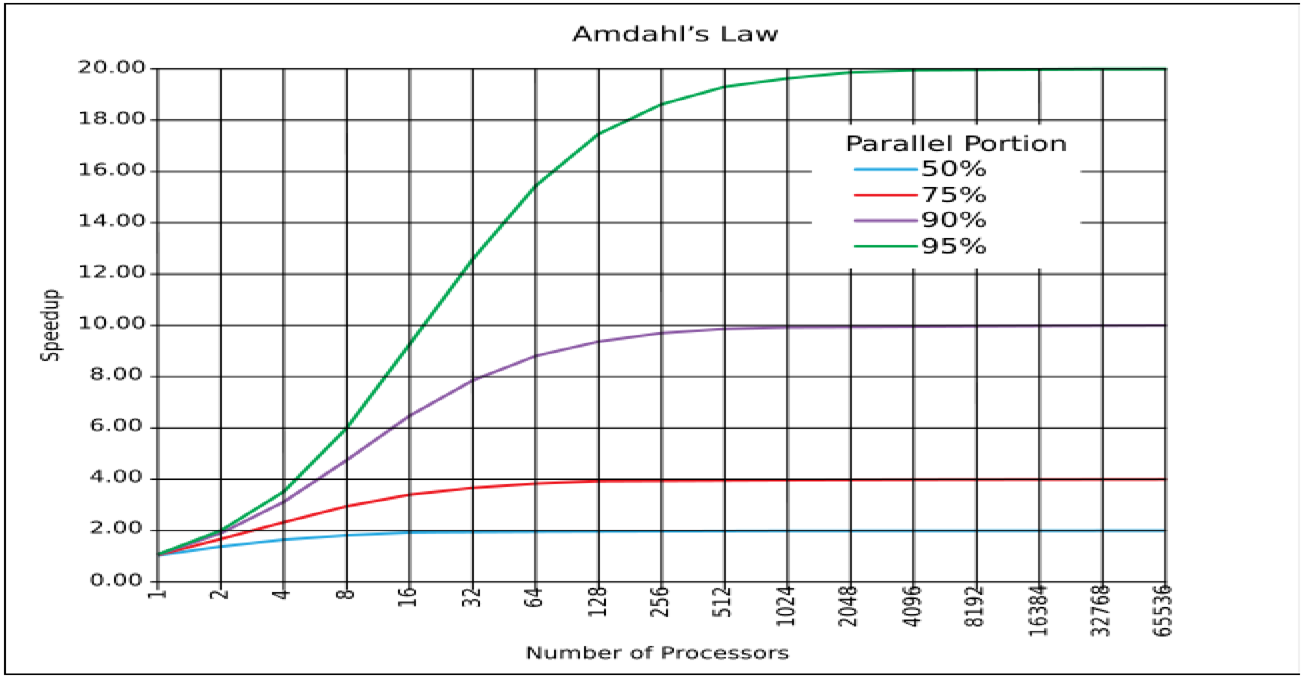
# Why scalability is not linear?

- Communication/Synchronization/Bookkeeping overhead
- Not all resources can grow linear

# Simple scalability model: Amdahl's law

$$\frac{1}{(1-P)+\frac{P}{N}}$$

P - % of parallelized part, N – number of unit



# Parallel Execution

Serial Computing Is Dead; the Future Is Parallelism

**SearchDataCenter.com (06/30/08)**

"Sequential programming is really hard, and parallel programming is a step beyond that."

**Andrew S.**

# Parallel architectures

- Multi CPU (Shared memory)
- Cluster of computers

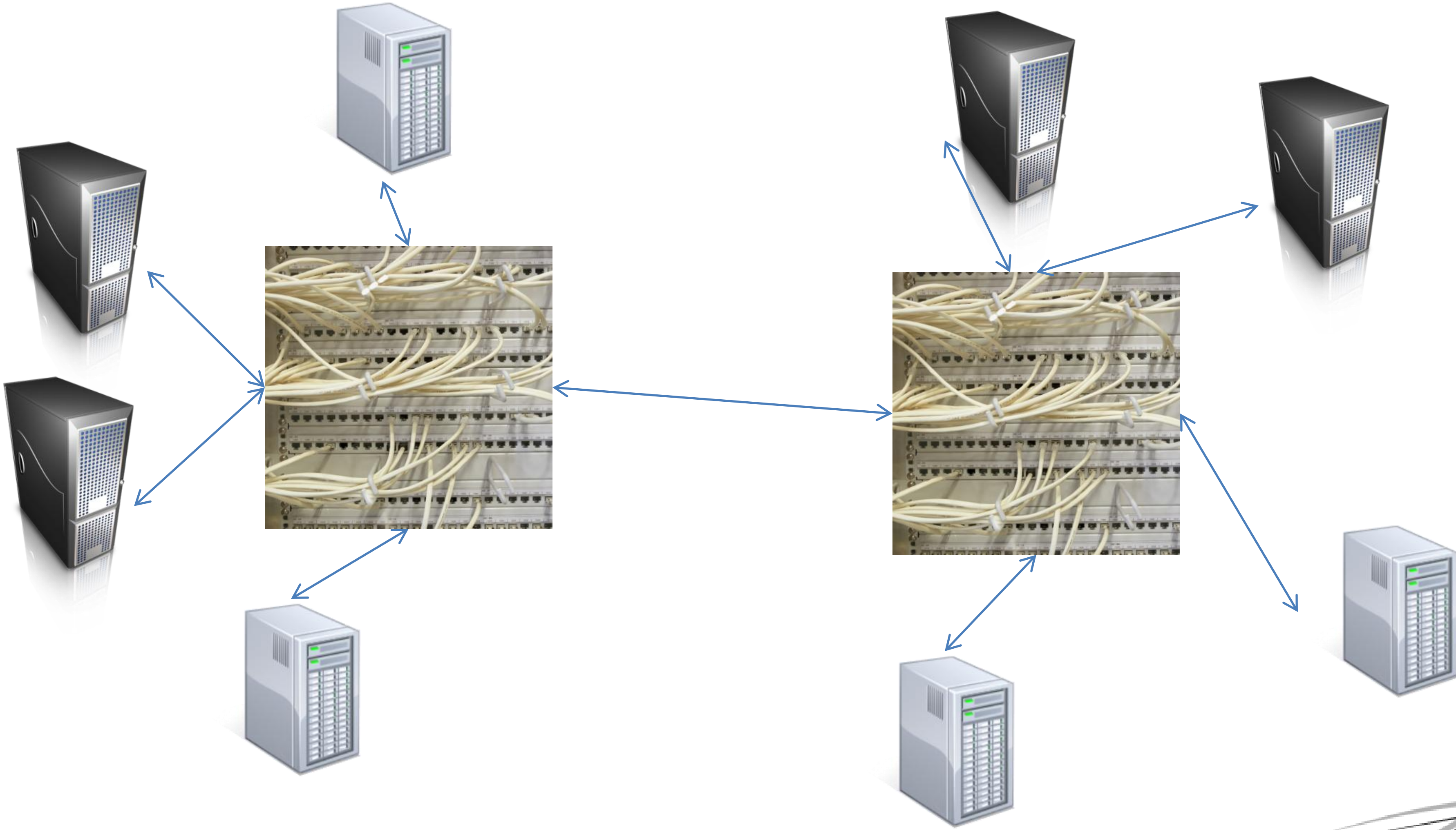
# Parallel on one box

- Threads – shared the same memory
- Processes – special inter-process communication

# Modern cluster arhitectures

- Special-purpose high-performance clusters (HP, IBM, Sun Cray – hardware & software solutions)
- Commodity hardware + cheap network

# Modern clusters



# Cluster features

- Slow exchange between nodes
- Failures
- Balancing

# Cloud computing

- Cloud application – Internet applications
- Cloud infrastructure – distributed cluster of computers
- Cloud platform – application framework (script language, storage)
- Cloud service – services that provided by cloud computers

# Summary

- Smaller  structures can be faster
- In-memory structures for dictionary
- Speller
- Cluster structures

# Q&A