Team BeautifulSoup

Christopher Gundler
Philipp Fukas
Jonas Rebstadt
Farina Kock
Christoph Stenkamp

A Hitchhiker's Guide to the Quantum World

DON'T PANIC

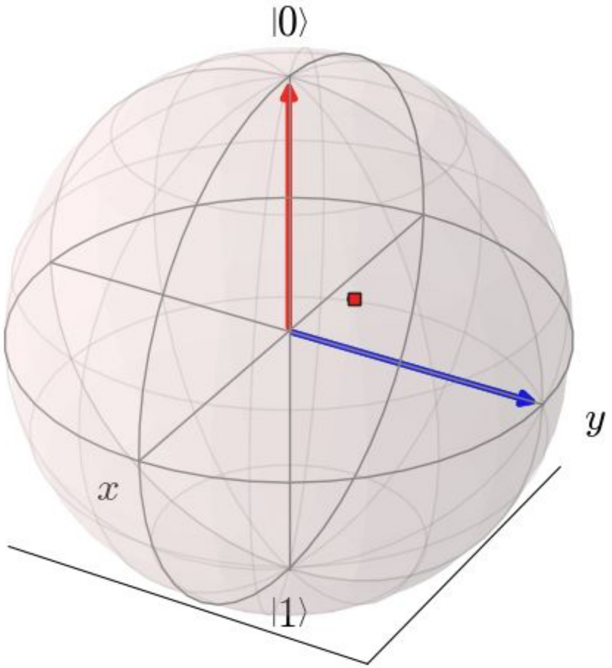
Introduction to Qubits

A qubit is a two-dimensional quantum-mechanical system that is in a state

$$|\phi\rangle = \alpha|0\rangle + \beta|1\rangle = \begin{pmatrix} \alpha \\ \beta \end{pmatrix}$$

with $|0\rangle = (0,1)^\mathsf{T}$ and $|1\rangle = (1,0)^\mathsf{T}$

Measuring the qubit leads to the classical bit
0 with probability $|\alpha|^2$,
1 with probability $|\beta|^2$



Bloch sphere representation

Superposition and Entanglement

Two quantum mechanical effects that can outperform classical algorithms are:

superposition and **entanglement**

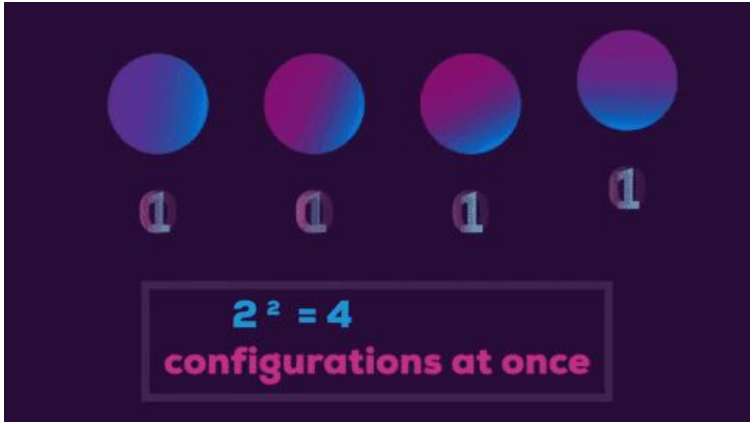
each qubit is in both states simultaneously (before measurement)

type of correlation between two or more qubits

Why may a quantum computer be faster?

- Quantum computers allows more efficient algorithms
  → massive parallelism

- Until now: Speed-up “proven” by complexity theory, not by real-life experience



Comparing complexities

Quantum Algorithm Zoo

- Reliable: **N**ational **I**nstitute of **S**tandards & **T**echnology (NIST), USA
- Up-to-date: Updated since 2011
- (Relatively) structured: Best classical algorithm vs. best quantum algorithm

https://math.nist.gov/quantum/zoo/

Quantum Algorithm Zoo

This is a comprehensive catalog of quantum algorithms. If you notice any errors or omissions, please email me at stephen.jordan@microsoft.com. Your help is appreciated and will be acknowledged.

Algebraic and Number Theoretic Algorithms

Algorithm: Factoring
Speedup: Superpolynomial
Description: Given an *n*-bit integer, find the prime factorization. The quantum algorithm of Peter Shor solves this in $\widetilde{O}(n^3)$ time [82,125]. The fastest known classical algorithm for integer factorization is the general number field sieve, which is believed to run in time $2^{\widetilde{O}(n^{1/3})}$. The best rigorously proven upper bound on the classical complexity of factoring is $O(2^{n/4+o(1)})$ via the Pollard-Strassen algorithm [252, 362]. Shor's factoring algorithm breaks RSA public-key encryption and the closely related quantum algorithms for discrete logarithms break the DSA and ECDSA digital signature schemes and the Diffie-Hellman key-exchange protocol. A quantum algorithm even faster than Shor's for the special case of factoring "semiprimes", which are widely used in cryptography, is given in [271]. If small factors exist, Shor's algorithm can be beaten by a quantum algorithm using Grover search to speed up the elliptic curve factorization method [366]. Additional optimized versions of Shor's algorithm are given in [384, 386]. There are proposed classical public-key cryptosystems not believed to be broken by quantum algorithms, *cf.* [248]. At the core of Shor's factoring algorithm is order finding, which can be reduced to the Abelian hidden subgroup problem, which is solved using the quantum Fourier transform. A number of other problems are known to reduce to integer factorization including the membership problem for matrix groups over fields of odd order [253], and certain diophantine problems relevant to the synthesis of quantum circuits [254].

Algorithm: Discrete-log
Speedup: Superpolynomial
Description: We are given three *n*-bit numbers *a*, *b*, and *N*, with the promise that $b = a^s \mod N$ for some *s*. The task is to find *s*. As shown by Shor [82], this can be achieved on a quantum computer in poly(*n*) time. The fastest known classical algorithm requires time superpolynomial in *n*. By similar techniques to those in [82], quantum computers can solve the discrete logarithm problem on elliptic curves, thereby breaking elliptic curve cryptography [109, 14]. A further optimization to Shor's algorithm is given in [385]. The superpolynomial quantum speedup has also been extended to the discrete logarithm problem on semigroups [203, 204]. See also Abelian hidden subgroup.

Algorithm: Pell's Equation
Speedup: Superpolynomial

Web scraping: Workflow

Algorithm: Factoring
Speedup: Superpolynomial
Description: Given an *n*-bit integer, find the prime factorization. The quantum algorithm of Peter Shor solves this in $\widetilde{O}(n^3)$ time [82,125]. The fastest known classical algorithm for integer factorization is the general number field sieve, which is believed to run in time $2^{\widetilde{O}(n^{1/3})}$. The best rigorously proven upper bound on the classical complexity of factoring is $O(2^{n/4+o(1)})$ via the Pollard-Strassen algorithm [252, 362]. Shor's factoring algorithm breaks RSA public-key encryption and the closely related quantum algorithms for discrete logarithms break the DSA and ECDSA digital signature schemes and the Diffie-Hellman key-exchange protocol. A quantum algorithm even faster than Shor's for the special case of factoring “semiprimes”, which are widely used in cryptography, is given in [271]. If small factors exist, Shor's algorithm can be beaten by a quantum algorithm using Grover search to speed up the elliptic curve factorization method [366]. Additional optimized versions of Shor's algorithm are given in [384, 386]. There are proposed classical public-key cryptosystems not believed to be broken by quantum algorithms, *cf.* [248]. At the core of Shor's factoring algorithm is order finding, which can be reduced to the Abelian hidden subgroup problem, which is solved using the quantum Fourier transform. A number of other problems are known to reduce to integer factorization including the membership problem for matrix groups over fields of odd order [253], and certain diophantine problems relevant to the synthesis of quantum circuits [254].

```
<b>Algorithm:</b> Factoring<br />
<b>Speedup:</b> Superpolynomial<br />
<b>Description:</b> Given an <i>n</i>-bit integer, find the prime
factorization. The quantum algorithm of Peter Shor solves this in
\( \widetilde{O} (n^3) \) time
[<a href="#Shor_factoring">82</a>,<a href="#Shor_factoring94">125</a>].
The fastest known classical algorithm for integer factorization is the
general number field sieve, which is believed to run in time \(
2^{\widetilde{O}(n^{1/3})} \). The best rigorously proven upper bound
on the classical complexity of factoring is \( O(2^{n/4+o(1)}) \) via the Pollard
[<a href="#Pol">252</a>, <a href="#Stras">362</a>]. Shor's
factoring algorithm breaks RSA public-key encryption and the
closely related quantum algorithms for discrete logarithms break the DSA
and ECDSA digital signature schemes and the Diffie-Hellman
key-exchange protocol. A quantum algorithm even faster than Shor's for
the special case of factoring &ldquo;semiprimes&rdquo;, which are
widely used in cryptography, is given in [<a href="#GLFB15">271</a>].
If small factors exist, Shor's algorithm can be beaten by a quantum
algorithm using Grover search to speed up the elliptic curve factorization
method [<a href="#PQRSA">366</a>]. Additional optimized versions of Shor's algori
cryptosystems not believed to be broken by quantum algorithms, <i>cf.</i>
[<a href="#BBD09">248</a>]. At the core of Shor's factoring algorithm is
order finding, which can be reduced to the <a href="#abelian_HSP">Abelian hidden
which is solved using the quantum Fourier transform. A number of other
problems are known to reduce  to integer factorization including the
membership problem for matrix groups over fields of odd order
[<a href="#BBS09">253</a>], and certain diophantine problems relevant to
the synthesis of quantum circuits [<a href="#RS12">254</a>].
<br /><br />
```

Web scraping: Workflow

Run Code

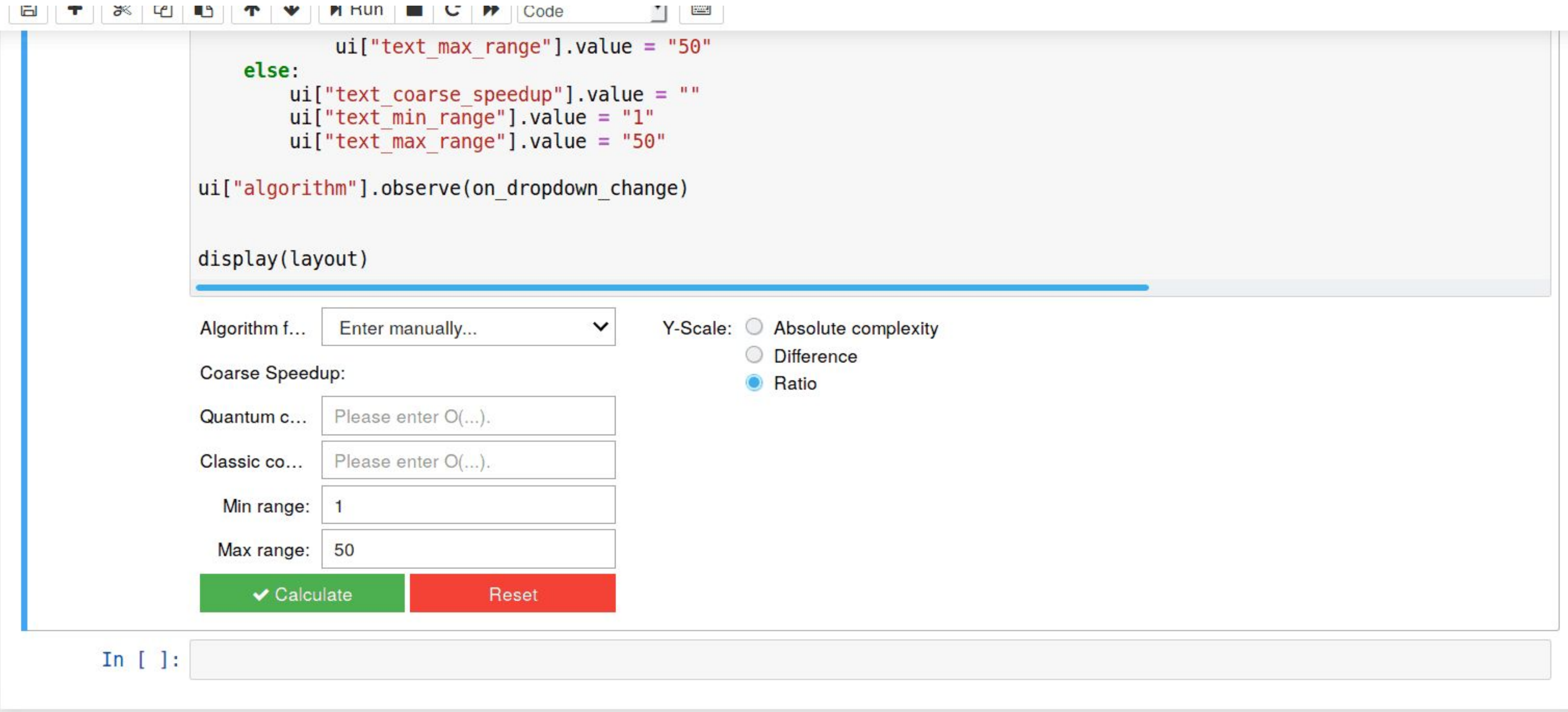
In [1]:
```
import requests
from bs4 import BeautifulSoup
import re
from copy import deepcopy
import csv
import itertools

bigrams = lambda l: list(zip(l[:-1], l[1:]))
new_sent_sign = '<NEWSENT>'
```

In [2]:
```
page = requests.get('https://math.nist.gov/quantum/zoo/')
contents = str(page.content)
contents = contents.replace('<b>Algorithm: </b>', '<b>Algorithm:</b>')
contents = contents.replace('<b>Description: </b>', '<b>Description:</b>')
# soup = BeautifulSoup(contents, 'html.parser') #don't even need bs, as the page is very non-semantic anyway
```

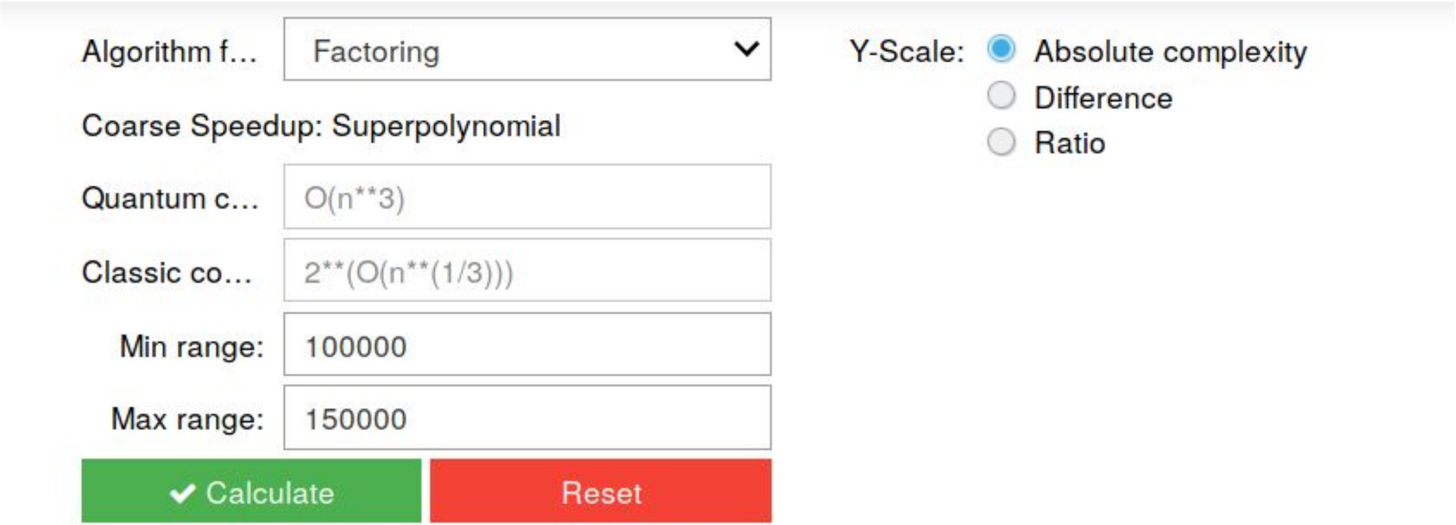
In [3]:
```
indices = bigrams([m.start() for m in re.finditer('<b>Algorithm:</b>', contents)]+[len(contents)])
complete_txts = [str(contents)[i[0]:i[1]] for i in indices]
```

In [4]:
```
all_algos = []
for i in complete_txts:
    this_algorithm = {}
    for name, searchstring in [['name', '<b>Algorithm:</b>'], ['speedup_coarse', '<b>Speedup:</b>'], ['description', '
        tmp = i[i.find(searchstring)+len(searchstring):]
        this_algorithm[name] = tmp[:tmp.find('<br />')].strip()
```

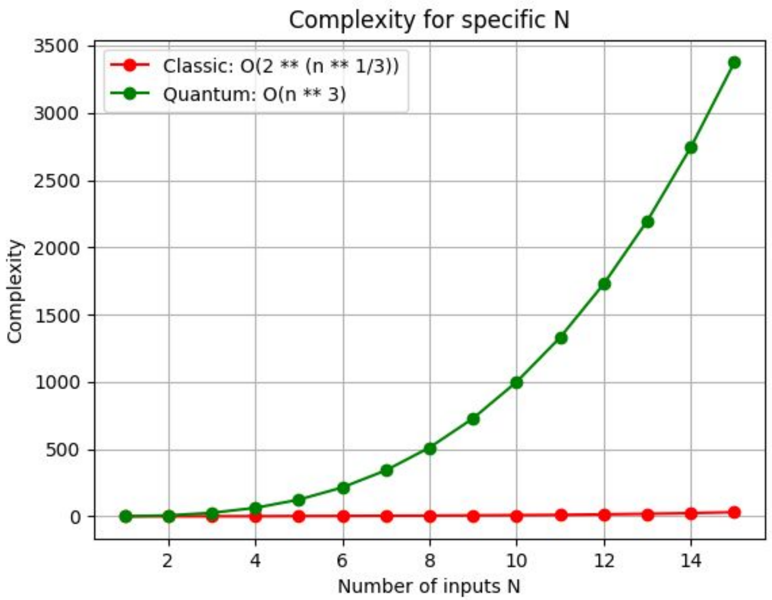
Visualising complexities

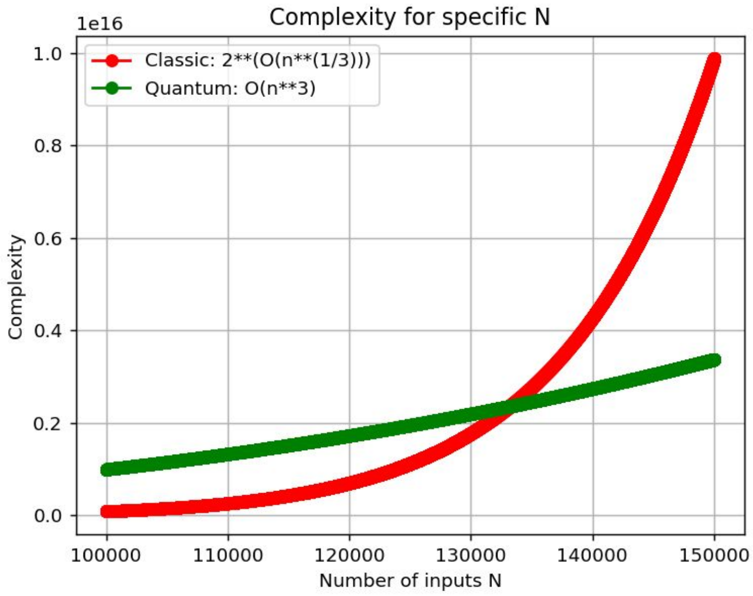


Visualising complexities

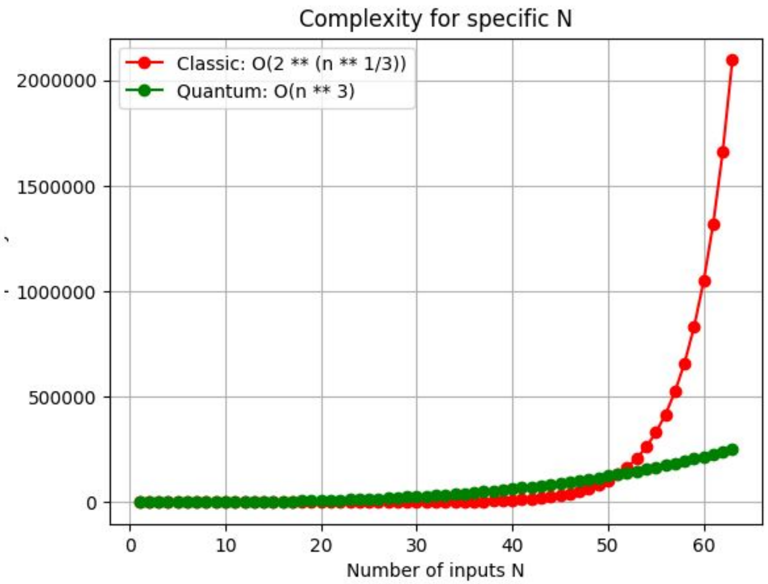


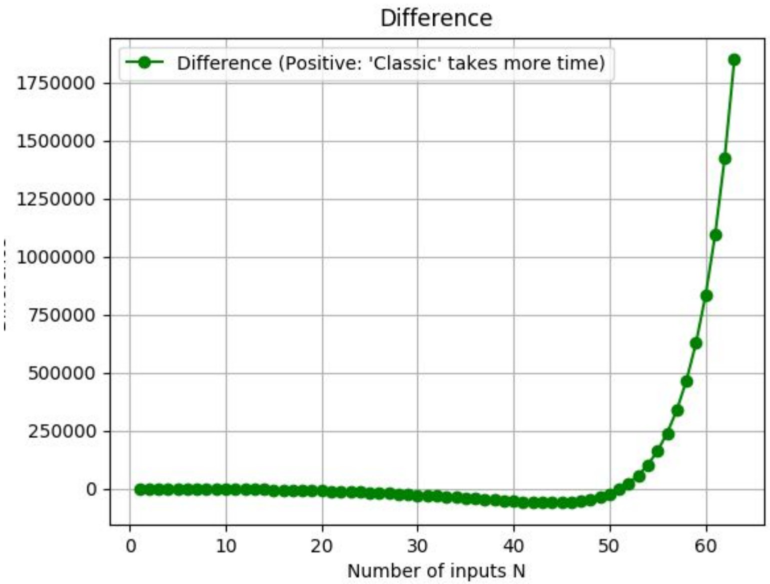
Visualisation: Domain matters!





Choose way of visualisation

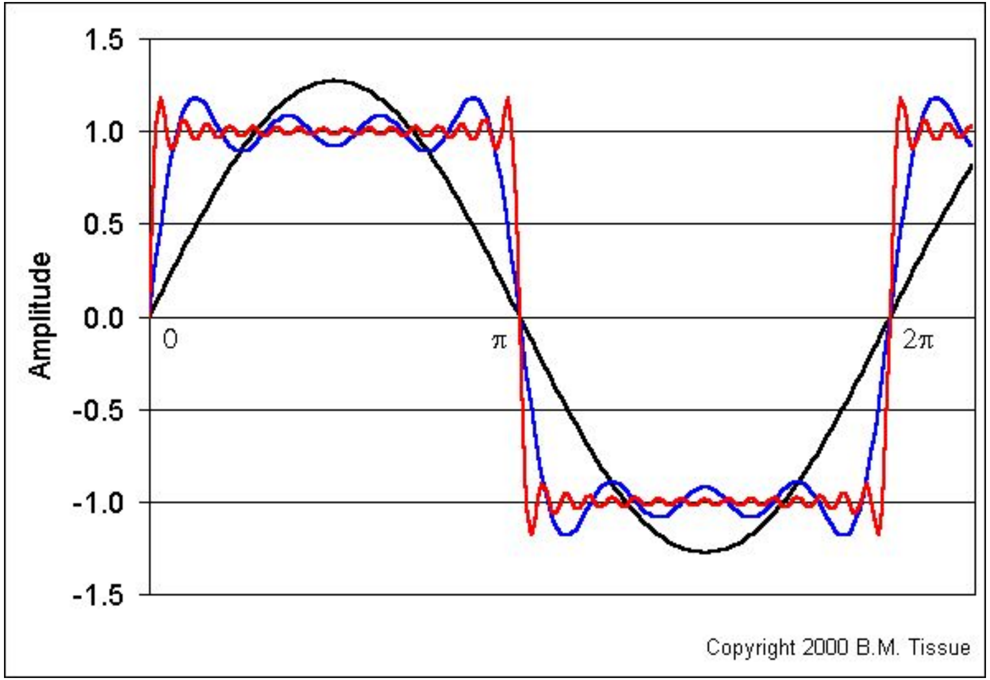




Idea of Fourier Transform

- Decompose complex signals into its basic components, e.g. with sin, cos

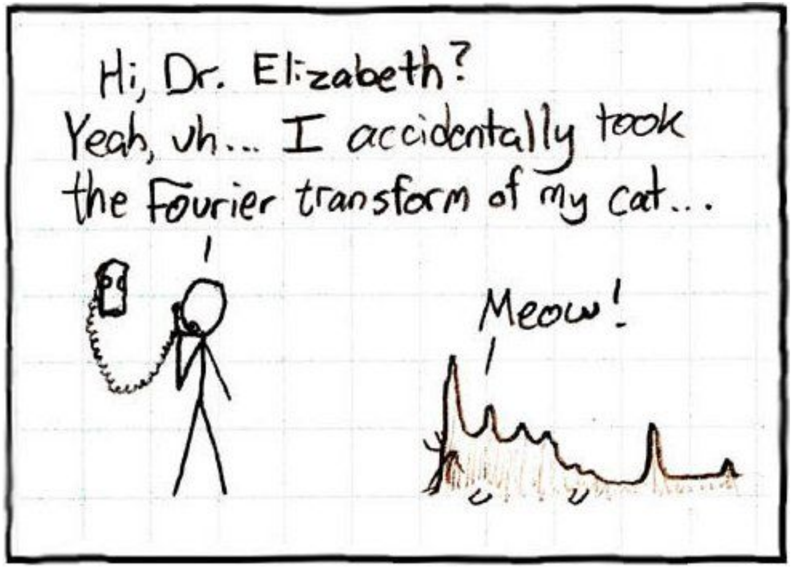
- In order to make it computationally solvable, the signal has to be discretised → leads to the idea of FFT



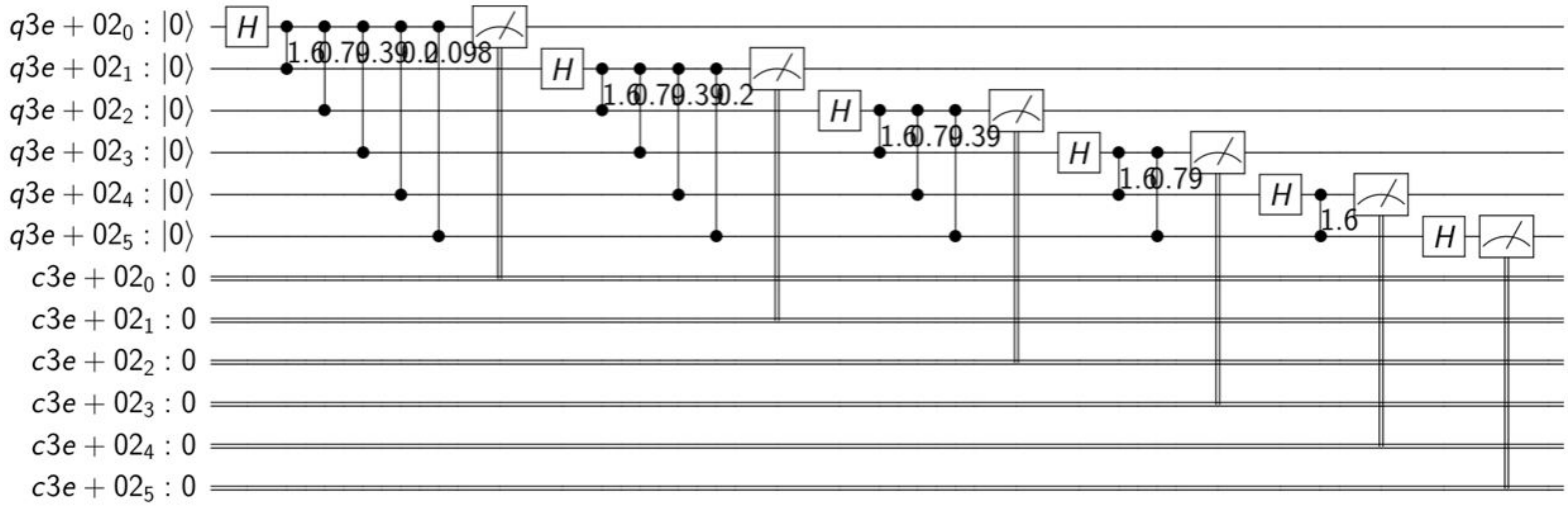
Quantum Fourier Transform

- Decomposition of Discrete Fourier Transformation into the product of unitary matrices
- Can be computed more efficiently by using quantum gates

→ Combination of **Hadamard Gates** and **Conditional Phase Rotations**

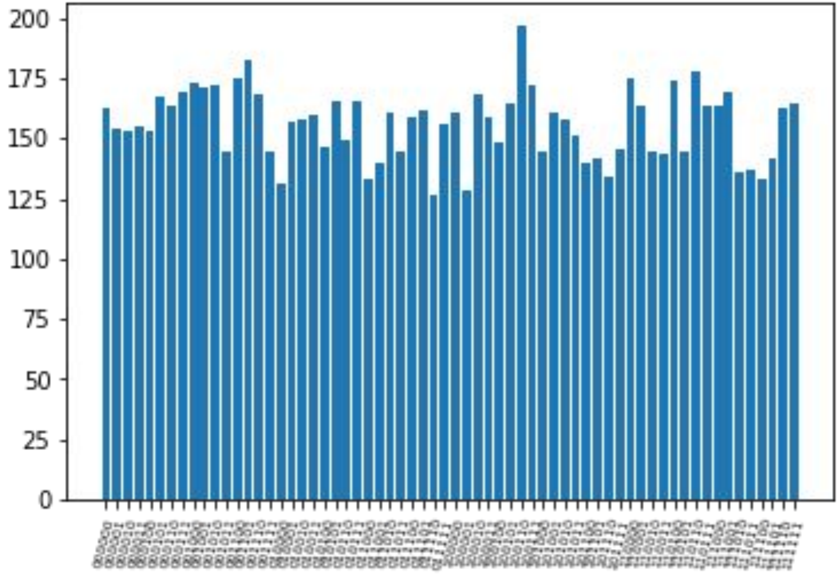


Circuit for Quantum Fourier Transform



Result for multiple tries

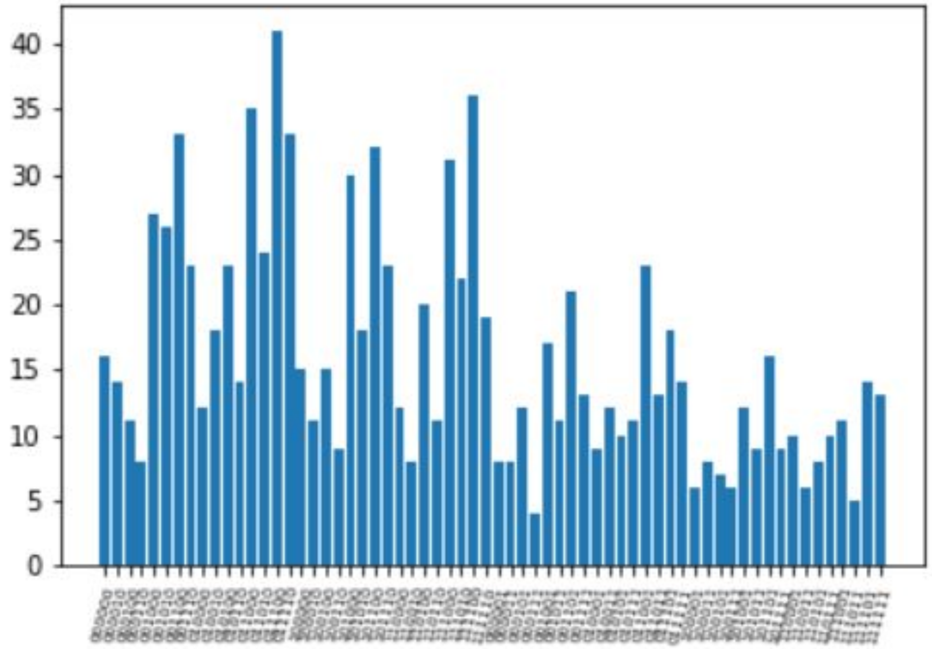
- Example with all inputs set to zero
- Simulated with 30000 shots



Result for multiple tries

- Same run on a Quantum computer

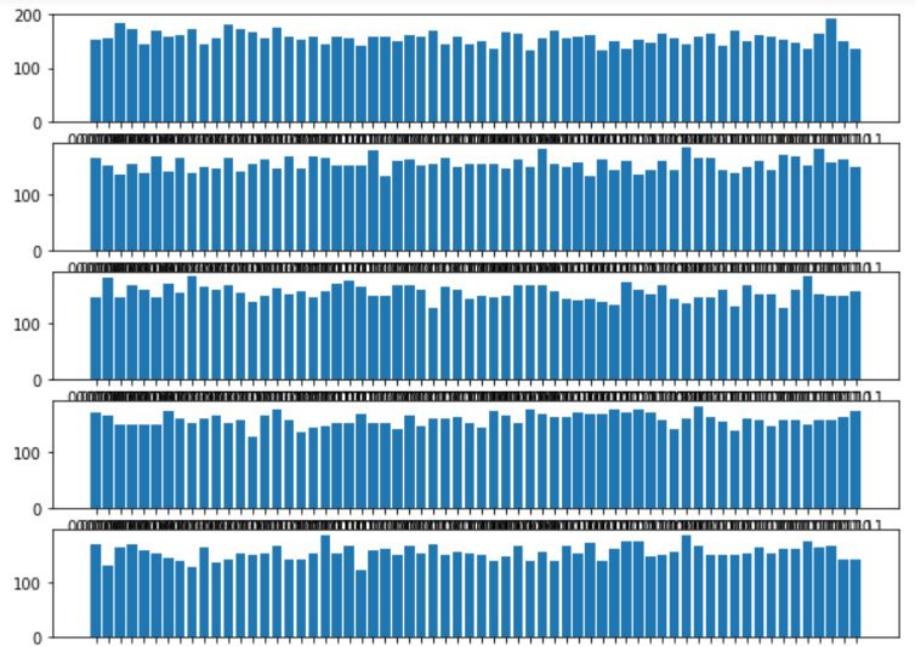
- Simulated with 1024 shots



Result for multiple tries with different initialisations

How can we achieve different initialisations?
→ by inverting some qubits

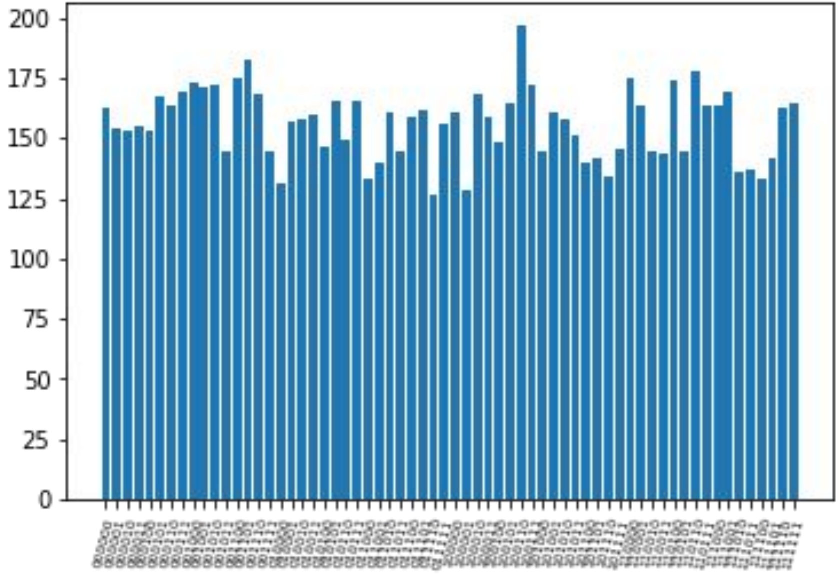
As you can see, different initialisations have no impact, all results are uniformly distributed



Result for multiple tries with entanglement

How can we achieve entanglement?
→ applying CNOT Gates

But simple entanglement has no impact, because all results are still equally distributed

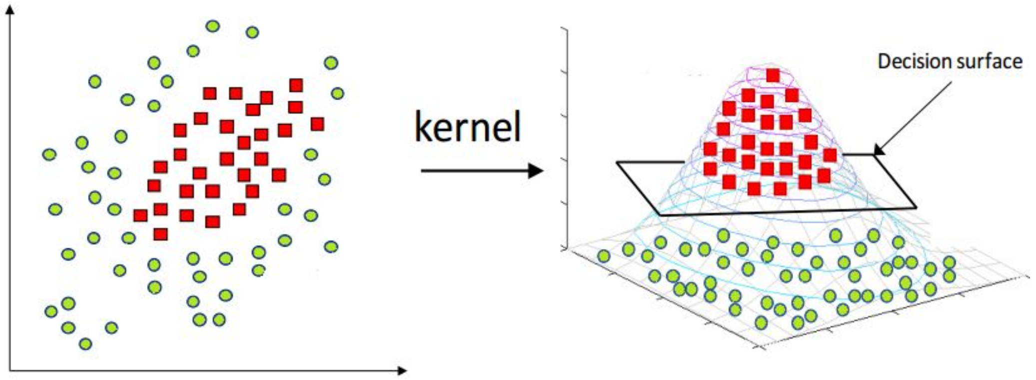


Which quantum gates are necessary for...

Classical computation	Probabilistic Computation	Quantum Computation
- CNOT - 180° rotation	- CNOT - 180° rotation - Hadamard	- CNOT - Hadamard - Phase Shift

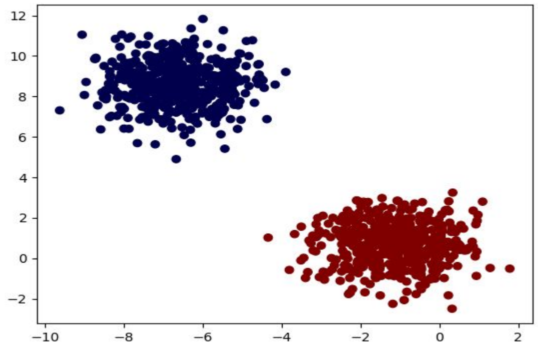
Quantum machine learning: SVM

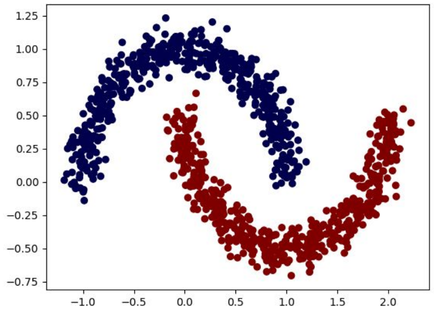
- Classical method of Data Mining and Artificial Intelligence
- Complexity: High possible speed-up (***M***: Samples, ***N***: Features)
    - Classical: $O(M^2(M + N))$
    - Quantum: $O(\log(M * N))$

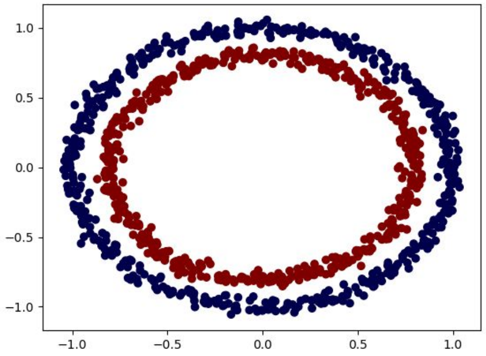


Rebentrost, P.; Mohseni, M. & Lloyd, S.[Quantum support vector machine for big feature and big data classification](). *arXiv:1307.0471*, 2013.

SVM: Testing with custom data







Thanks for your attention!