Big Numbers and Quantum Computers

Headstart talk

Susan Stepney

Department of Computer Science
University of York

how big is a million pounds?

- if I were to cover this table with a million £1 coins, how high would the pile be?

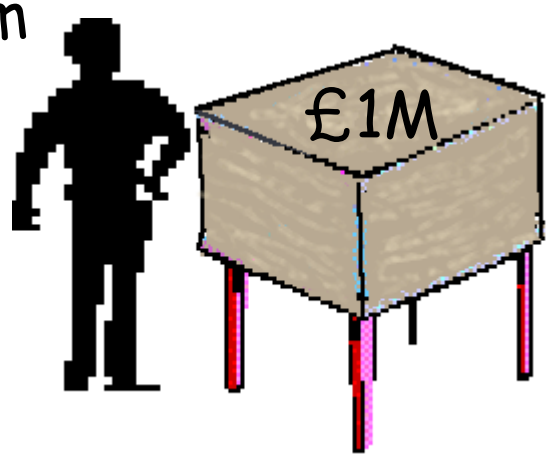
- anyone want to guess?

how big is a million pounds?

- table ~ 1m x 2m
- £1 coin ~ 2cm diameter, 2.5 mm thick

- 1m / 2 cm = 50 ; 2m / 2 cm = 100
- so one layer = 50 x 100 = **5000** coins
- number of layers = £1,000,000 / £5000 = **200**
- height = 200 x 2.5 mm = 500 mm = 0.5 m

- £1M = £10^6 → **0.5 m**
- £1bn = £10^9 → 500 m = 0.5 km
- £1tr = £10^{12} → 500 km



a million grains of sand

- 1 sand grain ~ 1mm x 1mm x 1mm
- 1 cm^3 = 10 mm x 10 mm x 10 mm
  = 1000 mm^3
- 1 litre = 10 cm x 10 cm x 10 cm
  = 1000 cm^3
  = 1,000,000 mm^3
  = 1 million grains
- 60 M people in the UK
  - 60 litres of sand → one grain per person

a billion and trillion grains of sand

- 1 m^3 = 1000 litres
  = 1,000,000,000 mm^3
  = 1 billion grains

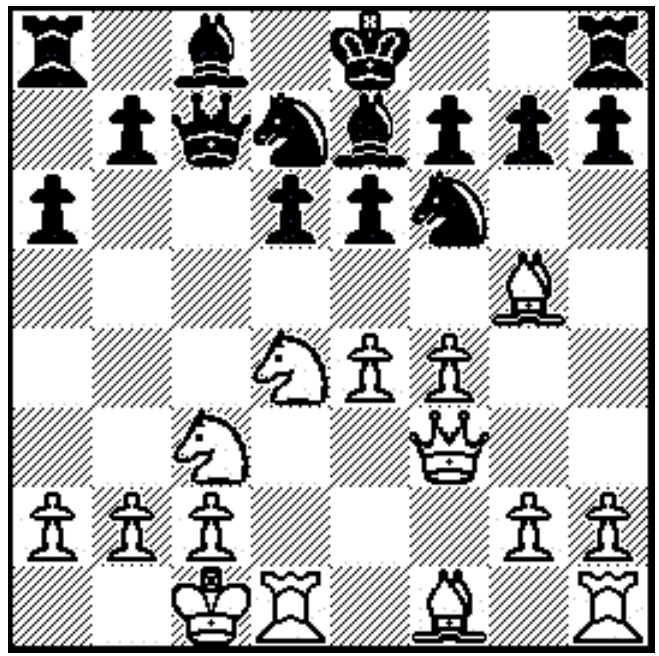
- 6 bn people in the world
  - 6 m^3 of sand → one grain per person

- 1000 m^3
  = 1,000,000,000,000 mm^3
  = 1 trillion grains

rice on a chessboard

legend has it thus :

- the king of India offered the inventor of chess a reward
- the inventor asked for
  - 1 grain of rice on 1st square of a chess board, 2 grains on 2nd square, 4 on the 3rd , and so on, doubling each time until the last square is reached
- the inventor was laughed at for being so modest
  - but -- just how much rice is on that last square?



http://www.cs.utah.edu/~thelenm/chess/board.gif

rice on a chessboard

- 2^0 grains on 1st square ; 2^1 on 2^{nd} ; 2^2 on 3rd ; 2^3 on 4th
  2^{63} grains of rice on the last square

- $2^{10} = 1024 \sim 10^3$
  $2^{63} = 2^{60} \times 2^3 \sim 10^{3 \times 6} \times 8 = 8 \times 10^{18}$ grains
  - a grain of rice ~ 10 mm^3
  - 80×10^{18} mm^3 = 80×10^9 m^3 of rice

- that's ~ ***10 m^3 of rice for every person on the planet***
  - or 10 m^3 of rice associated with every grain in 6m^3 of sand
  - a big reward for inventing chess!

how long is the king in debt?

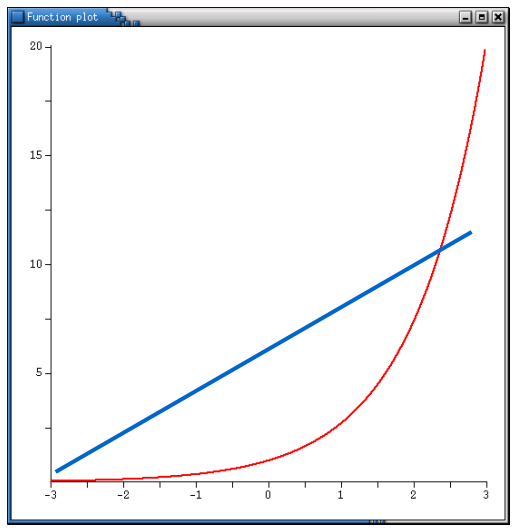
- the Encyclopedia Britannica gives the annual rice production as ~ 500 bn pounds / year
  - 1 pound ~ 0.5 kg
  - ~ 500 bn litres = 500 million m^3 = 5 x 10^8 m^3 of rice / year

- total amount / annual production
  = 80 x 10^9 m^3 / (5 x 10^8 m^3 / year)
  ~ ***100 years*** of the total rice production of the world
  - also implies each person consumes 10 m^3 of rice in 100 years, or, 0.1 m^3 / year, on average

(*caveat* : given the number of approximations made, these estimates could *easily* all be out by a factor of ten!)

linear *versus* exponential growth

- the rice fable is an example of ***exponential*** growth
- exponential growth gets *very* big *very* quickly
  - faster than *any* polynomial
    - people are not very good at comprehending exponential growth
- linear growth, like x
  - ***twice*** as many squares, twice as much rice
  - **polynomial** growth like x^n
    - x^2 : twice as many squares, 4 x as much rice
    - x^3 : twice as many squares, 8 x as much rice
- **exponential** growth, like m^x
  - ***one*** more square, twice as much rice
    - two more squares, 4 x as much rice
    - three more squares, 8 x as much rice



time to do a computation

- the ***time*** to do a computation depends on S, the ***size*** of the input

- if it depends **polynomially**, it is ***tractable***
  - grows like S^N
  - time taken grows roughly the same way the problem size grows
- if it depends **exponentially**, it is ***intractable***
  - grows like M^S (like the rice)
  - time taken grows *very* much faster than the problem size grows
  - even on the fastest computer in the world, or using *all* the computers in the world, it would take longer than ***the age of the universe*** to solve a reasonable size problem

polynomial time example

- search a telephone directory for a particular number
- start at the beginning, look at each number in turn
  - have to search half the book, on average

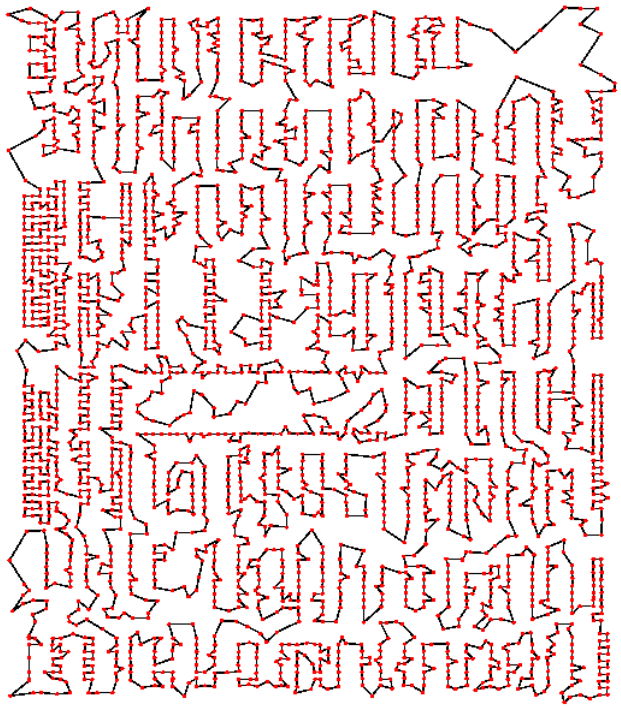
book size	1000 numbers /s	1,000,000 numbers /s
2000	**1 s**	1 ms
4000	2 s	2 ms
6000	3 s	3 ms
8000	4 s	4 ms
10,000	5 s	5 ms
2,000,000	1000 s	**1 s**

- time taken grows ***linearly*** with the size of the book
- computer speedup has a big effect

exponential time example

- "Travelling Salesman Problem"

- visit N cities, travelling the minimum total distance
  - start at city 1
  - N-1 ways of choosing the first city to visit
  - N-2 ways of choosing the second city
  - …
  - so (N-1) x (N-2) x … x 1 = (N-1)! different routes in total
    - 1! = **1**, 2! = **2**, 3! = **6**, 4! = **24**, 5! = **120**, 6! = **720**, 7! = **5040**, …
  - N! grows ***exponentially*** *fast*
    - and we don't know if there are any faster algorithms



http://www.tsp.gatech.edu//gallery/tours.html

exponential time example

cities, N	(N-1)!	1000 paths /s	1,000,000 paths /s
2	1	**1 ms**	1 μs
4	6	6 ms	6 μs
6	120	0.12 s	0.12 ms
8	5040	5 s	5 ms
10	362,880	6 mins	0.3 s
20	~ 10^{17}	4 million years	4,000 years
200	~ 4×10^{372}	~ 10^{362} years	**~ 10^{359} years**

- time taken to explore all possible routes grows ***exponentially*** with the number of cities

Moore's Law

- computing power / size / speed doubles every **18 months**

- either solve problems faster, or solve bigger problems

Moore's Law

- assume you have a fixed amount of computer time
  - say you have to solve the problem by tomorrow
    - like weather forecasting

- Moore's law means that, in a fixed time
  - you can search a telephone directory that itself ***doubles in size*** every 18 months
  - you can search for the shortest travelling route through a set of cities that grows by ***one city*** every 18 months

using lots of computers

- enormous total amount of rice / world population = relatively small amount of rice *per person*
  - there are a *lot* of people in the world
- computer time is often given in "processor years"
  - such and such a problem would take a "million processor years"
  - that's a million years on *one processor*
  - but only *one year* if you have a million processors
  - there are a *lot* of computers in the world, attached to the Internet

http://www.center.nitech.ac.jp/index-e.html

- Grand Internet Mersenne Prime Search (GIMPS); SETI@home; … all using thousands or millions of PCs around the world
- a "global supercomputer"

prime numbers

- prime numbers are of interest in Computer Science
- ***prime*** = an integer, greater than 1, that is divisible only by 1 and itself
  - 2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47, 53, 59, 61, 67, 71, 73, 79, 83, 89, 97, …
- a number that is not prime is ***composite***
  - each composite number has a *unique* prime factorisation
    - 4 = 2 x 2
    - 6 = 2 x 3
    - 8 = 2 x 2 x 3
    - 9 = 3 x 3
    - 10 = 2 x 5
    - 12 = 2 x 2 x 3
    - …

how many primes are there? (Euclid's proof)

- *assume* there are finitely many primes
- list them out
  - $p_1, p_2, \ldots p_N$
- form their product, and add 1
  - $R = p_1 \times p_2 \times \ldots \times p_N + 1$
- what are the prime factors of R?
  - $R \div p_1$ has a remainder of 1, so does $R \div p_2$, and so on
  - so *none* of the $p_1, p_2, \ldots p_N$ is a prime factor of R
- there are two possibilities
  - R is composite, with a **prime factor p_M** not in the original set
  - **R is prime**, yet not in the original set

how many primes are there? (Euclid's proof)

- so the assumption
  - that there are finitely many primes
- leads to a *contradiction*
  - the existence of another prime *not* in the set of all primes
- so the assumption cannot be true!

- *reductio ad absurdum*
  - there are ***infinitely many primes***

is this number a prime?

- 8,388,606
  - composite
- 8,388,607
  - composite
- 24,036,583
  - prime
- $2^{137} - 1$
  - composite
- $2^{24,036,583} - 1$
  - prime : largest known, with 7,235,733 decimal digits!
- the time taken to discover if a given number is prime or composite is ***polynomial*** in the number of digits
  - ***primality testing is tractable***

what are its factors?

- so it is easy to find out *if* a number is composite
- but it is *not* easy to discover its factors
  - the tractable primality test does *not* tell you the factors
    - 15 = 3 x 5
    - 8,388,606 = 2 x 3 x 23 x 89 x 683
    - 8,388,607 = 47 × 178,481
    - $2^{137}-1$ = 32,032,215,596,496,435,569 × 5,439,042,183,600,204,290,159
- the time taken to discover the factors of a composite number is ***exponential*** in the number of digits
  - ***factorisation is intractable***

cryptography

- prime numbers are important in some forms of cryptography called "public key cryptography"
  - keeping certain communications, like banking transactions, secret and secure

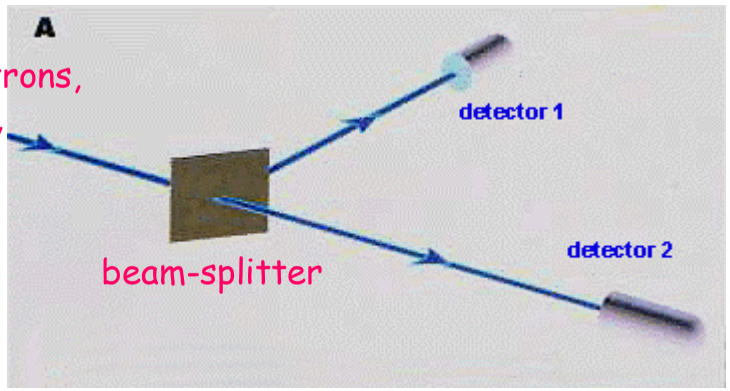
- take two large primes p, q
  - many hundreds of digits
  - p and q are easy to find, because primality testing is tractable
  - then form their product **r = p x q**
  - use r as the basis of a cryptography scheme
    - you need to know p or q to decrypt
    - finding p and q given only r is the intractable factorisation problem

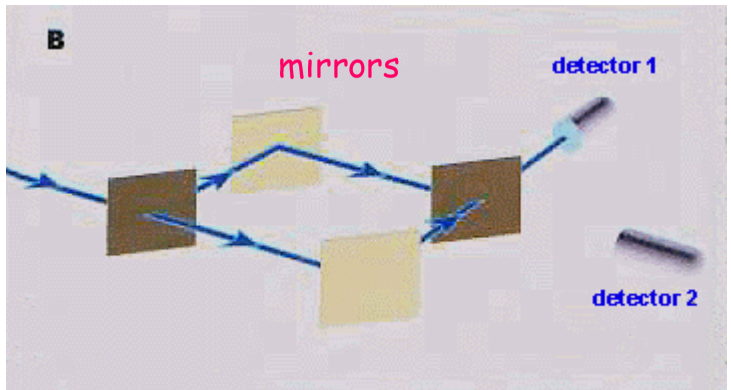
quantum computers

- today's computers are classical computers
  - they do only one calculation at a time
  - to do lots of calculations "in parallel", you need lots of computers
    - either using lots of PCs on the Internet
    - or special purpose hardware
  - even so, you can't get hold of exponentially growing number of computers
    - remember the rice

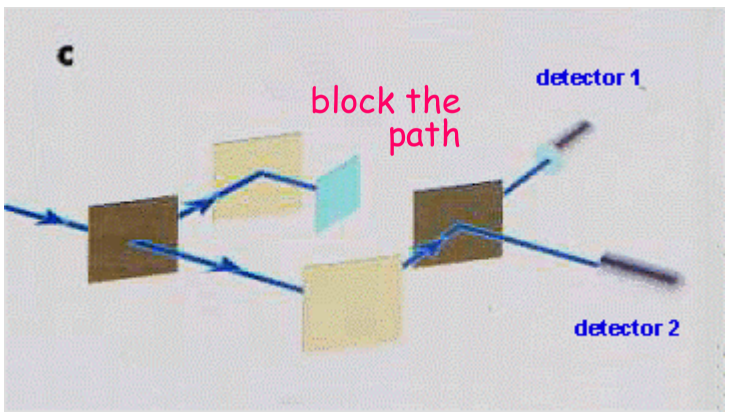
- quantum computers can do an ***exponential*** number of calculations in parallel

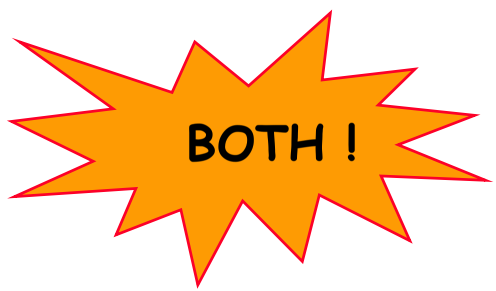
the quantum ingredient





which path is taken?





classical states

- one bit classical state
  - 0 or 1
    - 2 different states
- two bit classical state
  - 00 or 01 or 10 or 11
    - 4 different states
- three bit classical state
  - 000 or 001 or 010 or 011 or 100 or 101 or 110 or 111
    - 8 different states
- n bit classical state
  - 00…00 or … or 11…11
    - 2^n different states
- the number of classical states is ***exponential*** in the number of bits

quantum states

- one bit quantum state
  - $|0\rangle + |1\rangle$
    - 1 ***superposition*** state of 2 classical states
- two bit quantum state
  - $|00\rangle + |01\rangle + |10\rangle + |11\rangle$
    - 1 superposition of 4 classical states
- three bit quantum state
  - $|000\rangle + |001\rangle + |010\rangle + |011\rangle + |100\rangle + |101\rangle + |110\rangle + |111\rangle$
    - 1 superposition of 8 classical states
- n bit quantum state
  - $|00\ldots00\rangle + \ldots + |11\ldots11\rangle$
    - 1 superposition of 2^n classical states
- *one* quantum state is a superposition of an ***exponential*** number of classical states

exponential speed-up

- so *n* quantum bits can be used to encode and calculate with 2^n classical numbers *simultaneously*
  - adding one quantum bit doubles the power
- there is a catch!
  - you can do 2^n calculations simultaneously ☺
    - eg exploring all TSP paths; trying all factorisations
  - but you can observe only *one* of the results ☹
    - eg length of *one* of the paths; result of *one* factorisation attempt
- most of quantum programming goes into ensuring the result you observe is the one you *want* to see
    - the *shortest* path; the *successful* factorisation

quantum factorisation

- so far no-one has found out how to do this with the classically intractable Travelling Salesman Problem
  - and it might not be possible

- however, someone *has* worked out how to do this for the factorisation problem
  - Peter W. Shor, 1997

- ***quantum factorisation is tractable***

http://www-math.mit.edu/~shor/

the banks are safe, for a while

- so why are people still using cryptography based on factorisation schemes?
- quantum computers are in their infancy
  - the biggest quantum computer is only 7 quantum bits
  - to date it has managed to successfully factorise :

- but remember, just *one* extra quantum bit *doubles* the quantum computer's power, and remember Moore's law
  - ***factorisation is living on borrowed time***

secure quantum communications

detecting eavesdropping

- ***classically*** : a spy can intercept and read a message, then copy it and send it on
  - without you knowing it has been intercepted
- ***quantumly*** : reading the message destroys some of its special quantum properties, and it *can't be faithfully copied* and sent on
  - if you try, you end up sending garbage
  - *no cloning* theorem
- so: any attempt at eavesdropping on a quantum channel can be *detected*

http://www.carsearch.com/i/spy.gif

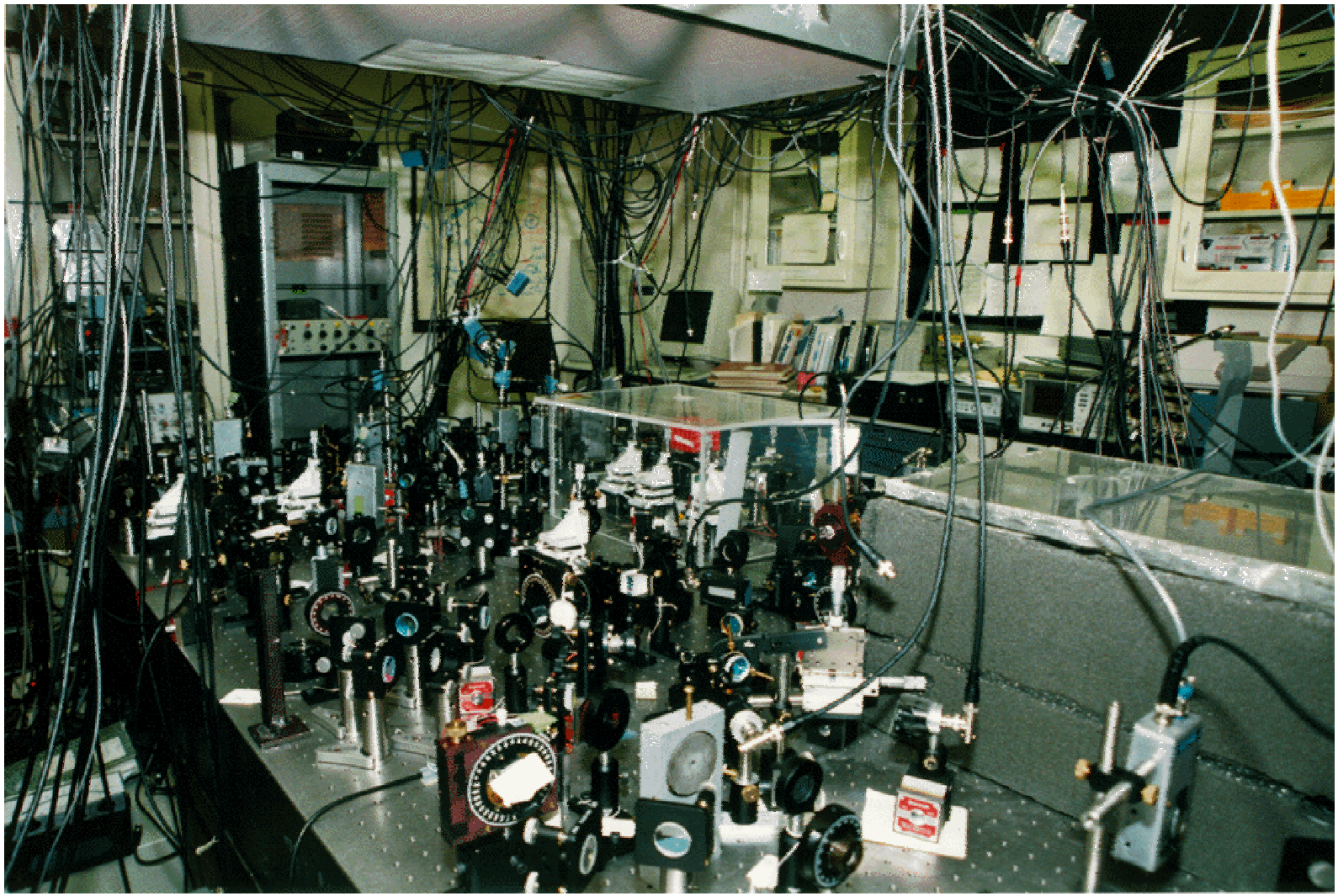
quantum teleportation

- using ***quantum entanglement***, a quantum state can be moved from A to B without passing through any intermediate positions

http://www.joeydragon.com/tng_transporter.htm

- can't do this faster than light ☹
- can't "beam" somewhere unknown ☹
- provides an ***untappable*** communication channel, as the information does not even *exist* at any intermediate point

teleporter at Kimble Lab, Caltech



this is just the beginning

- quantum computing is in its infancy
- we know some ***exponential speedup*** is possible
  - but not for every problem
- we know some new communication possibilities
  - detecting eavesdroppers, teleportation, …
    - things *impossible* with classical computers
- we *know* that we *don't know* everything about quantum computation
- it's an exciting future area for computer science