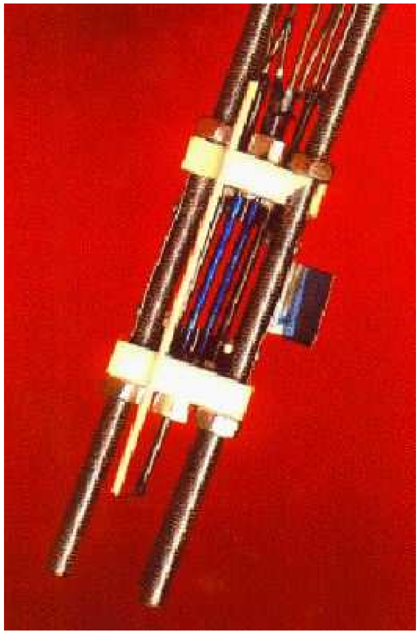
Quantum search algorithms

Christoph Dürr
LRI, Univ. Paris-Sud

version 4 for the spring school at Montagnac les Truffes

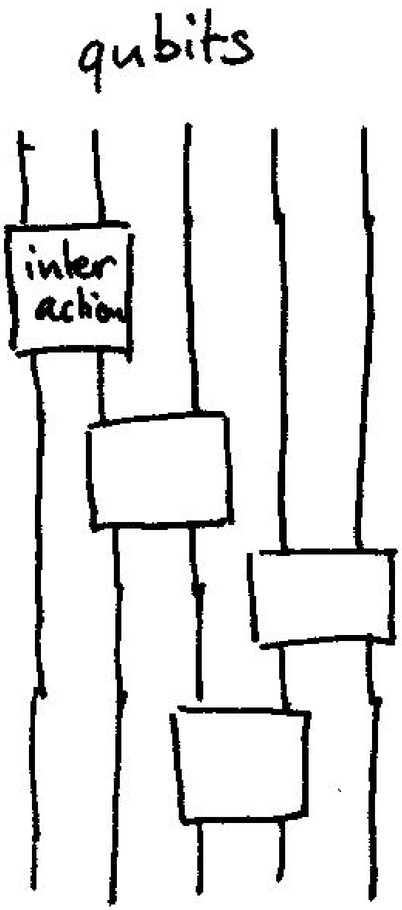
- Circuits
- Grover's search algorithm
- 3-Sum
- Finding the minimum
- Minimum spanning tree
- Searching in an ordered table

A possible implementation of a quantum computer



- A dozen ions are trapped in a magnetic field
- they can have spin up or down ($|0\rangle$ or $|1\rangle$)
- inside a laser beam they stand still
- otherwise the oscillate and interact with neighbors

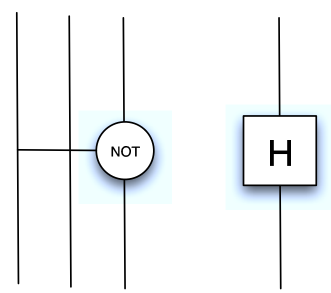
The circuit model of computation



- Wires represent qubits, times goes downwards
- two-qubit interactions are represented as gates
- There is a unitary matrix $M \in \mathbb{C}^{2\times 2}$ associated to each gate
- Its action is $M \otimes Id$ on the overal qubits space
- At the end we observe the qubits and the outcome of the computation

More on circuits

- Gates should only be drawn from a universal, realistic set of gates, as for example
  { Crtl-Crtl-Not (=Toffoli gate), Hadamard}



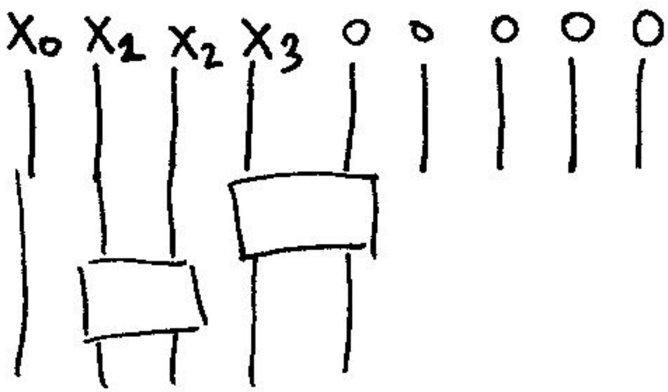
- the number of gates is the time complexity of it

- its depth the parallel computation time complexity

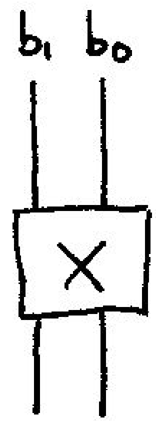
Two ways to encode the input

let be the binary input $x \in \{0,1\}^n$

In the initial configuration



In a query gate



X maps $|b_1 b_0\rangle$ to $(-1)^{x_b}|b_1 b_0\rangle$, where $b = 2b_1 + b_0$.

Query model

- An algorithm corresponds to a description of a family of circuits (for each value of n) which is uniform in the sense that in time poly(n) the n-th circuit can be produced
- Clearly the number of query gates $\leq$ the number of arbitrary gates
- So a lower bound on the number of queries is a lower bound on the time complexity in this model
- For our algorithms today, these two are identical (up to a logarithmic factor)
- We are interested only in randomized algorithms (which succeed with probability at least $2/3$)

The search problem

on a table $f \in \{0,1\}^N$

unstructured case

we want x such that $f(x) = 1$,

$$f = 000000010000$$

sorted case

we want the smallest x such that $f(x) = 1$,
knowing that f is sorted and $f(N) = 1$.

$$f = 000000011111$$

Query complexity : how many queries to f are necessary?

The unstructured search

Quantum query complexity

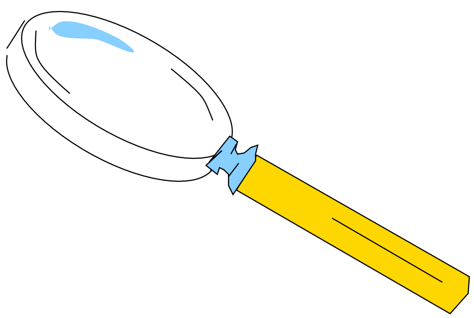
- deterministic case $\Theta(N)$
- probabilistic case $\Theta(\sqrt{N})$
  Time complexity $O(\log(N)\sqrt{N})$

Algorithm of Lov Grover 1996

working space $\mathcal{H} = \mathbb{C}^N$

Idea

The superposition $\sum_x \alpha_x |x\rangle$ consists of N basis states, divided into "good ones" (for $f(x) = 1$) and "bad ones" (for $f(x) = 0$).



The goal is to amplify the good amplitudes in order to increase the probability of observing a solution to the search problem.

Operators

1. Query gate

   $$U_f : |x\rangle \mapsto (-1)^{f(x)}|x\rangle$$

   U_f changes the phase of the "good" amplitudes

2. the diffusion operator D (be patient, definition comes in two slides)

Algorithm

Suppose that there exist a single $x' \in [N]$ such that $f(x') = 1$.

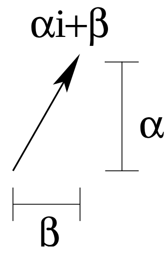
1. Initialize with the uniform superposition $\sum_x |x\rangle$
   let's forget the normalisation factors

2. Apply DU_f $\lfloor \frac{\pi}{4}\sqrt{N} \rfloor$ times

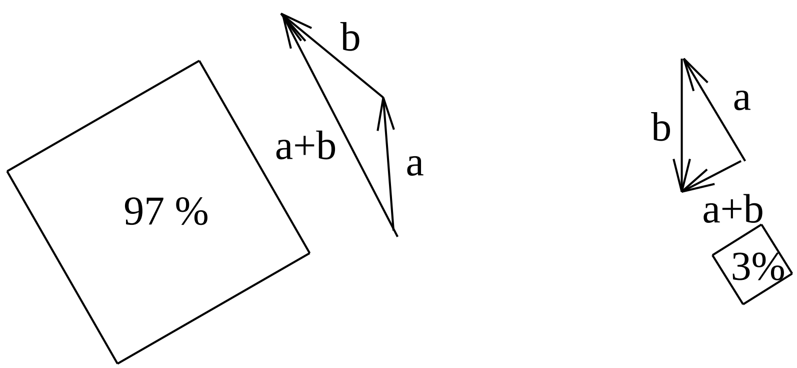
3. Observe. (the probability to observe x' is high)

Let's see graphically what happens

Draw an amplitude as a vector



The probability to observe a basis state is proportional to the square of the length of the vector. Amplitudes add like vectors.



Definition of D (finally!)

$$D = -H_N U_0 H_N$$

where U_0 flips only the amplitude associated to $|0\rangle$

$$U_0 = \begin{pmatrix} -1 & 0 & & 0 \\ 0 & 1 & & 0 \\ & & \ddots & \\ 0 & 0 & & 1 \end{pmatrix}$$

and H_N is the Hadamard transform, from which we only need

$$H_N|0\rangle = \sum_x |x\rangle$$

D is the inversion about the mean

Let be the mean $\mu = \frac{1}{N}\sum_x \alpha_x$. Then D maps

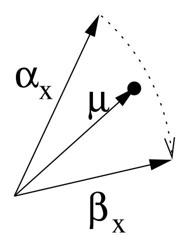
$$\sum_x \alpha_x |x\rangle := \sum_x (\mu + \alpha'_x)|x\rangle$$

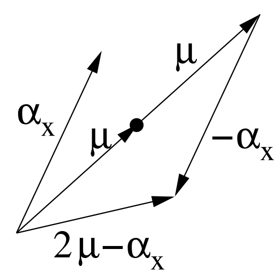
to

$$\sum_x (\mu - \alpha'_x)|x\rangle$$

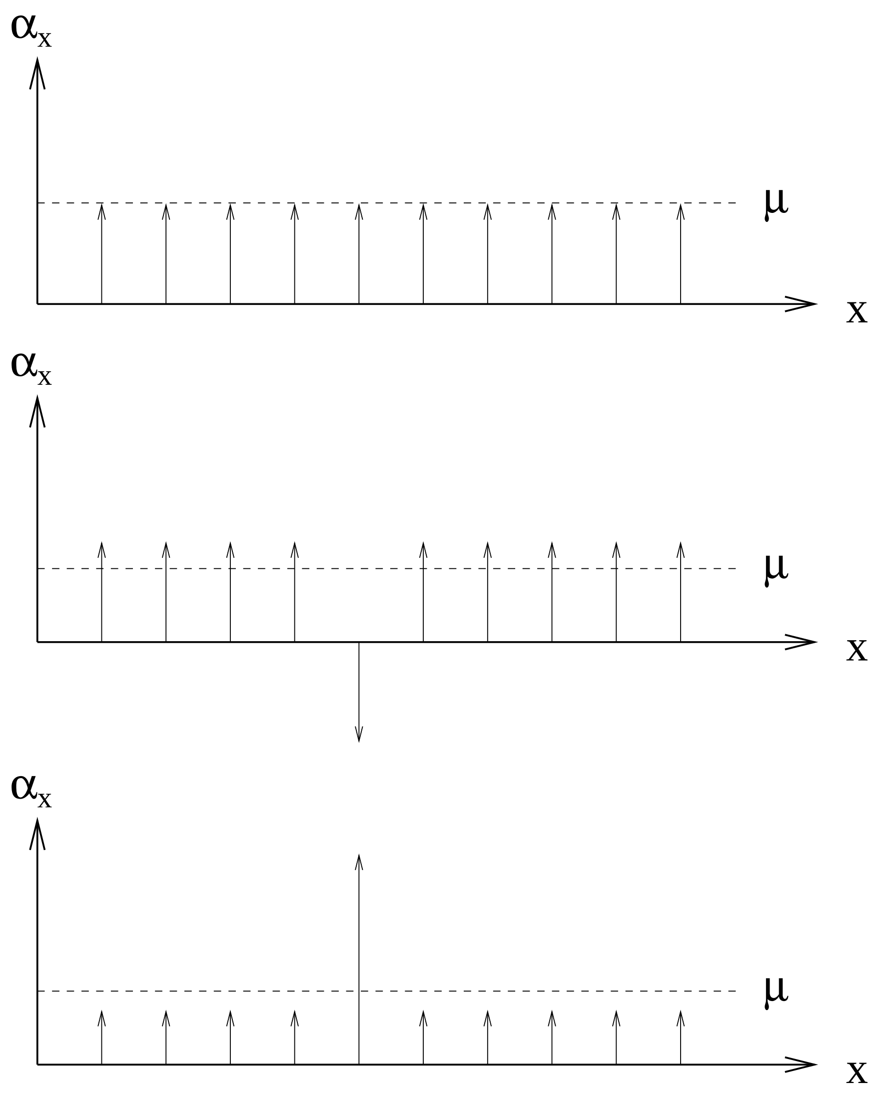
Explanation

$$D = 2\begin{pmatrix} \frac{1}{N} & & \frac{1}{N} \\ & \ddots & \\ \frac{1}{N} & & \frac{1}{N} \end{pmatrix} - I$$





Evolution of the algorithm



initial superposition

after application of U_f

after application of D

The evolution happens in a tiny subspace

At every moment all amplitudes α_x for $f(x) = 0$ are real, and are the same.

The same happens for the *good amplitudes*.

Therefore

Let

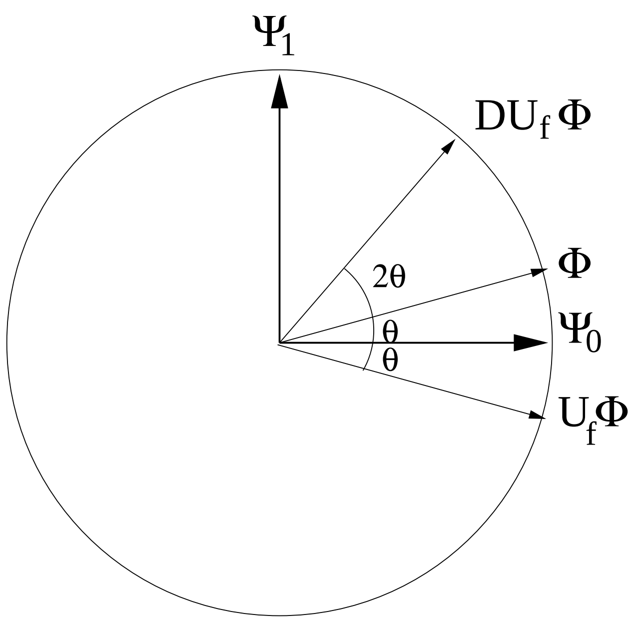
$$
\begin{aligned}
|\Psi_0\rangle &= \sum_{x:f(x)=0} |x\rangle \\
|\Psi_1\rangle &= \sum_{x:f(x)=1} |x\rangle
\end{aligned}
$$

So the algorithms involves only in the subspace spanned by $|\Psi_0\rangle, |\Psi_1\rangle$.

DU_f makes a rotation by angle 2θ

Let $|\Phi\rangle = \sum_x |x\rangle$ and θ the angle in the circle spanned by $\{\Psi_0, \Psi_1\}$. Then

- U_f is the inversion about $|\Psi_0\rangle$
- D is the inversion about $|\Phi\rangle$.



Required number of iterations

$$\underbrace{DU_f \ldots DU_f}_{k} |\Phi\rangle = \sin((2k+1)\theta)|\Psi_1\rangle + \cos((2k+1)\theta)|\Psi_0\rangle$$

But $\sin(\theta) = \sqrt{\frac{1}{N}}$, therefore the probability of observing the good basis state $|x'\rangle$ is *maximized*

$$k \sim \frac{\pi}{4}\sqrt{N}$$

Variants of this algorithm

- [Boyer,Brassard,Høyer,Tapp,1997]
  If there are t solutions then the complexity is $\Theta(\sqrt{N/t})$
- If t is not known in advance, there is an algorithm which never errs, but its expected complexity is $\Theta(\sqrt{N/t})$. Moreover each output has equal probability $1/t$.
- To get the error probability down to ϵ classically we do $\log(1/\epsilon)$ repetitions and output the majority. Quantumly we just need $O(\sqrt{\log(1/\epsilon)})$ repetitions.

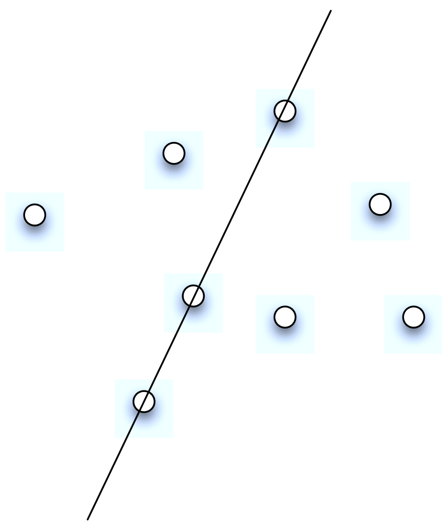
3-Sum

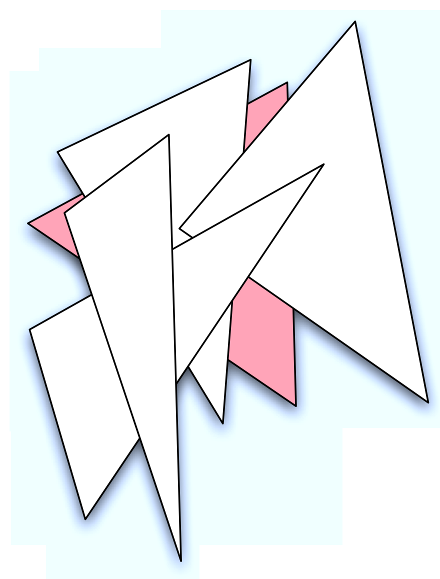
[Bahinav,Dürr,Lafaye,Kulkarni,04]

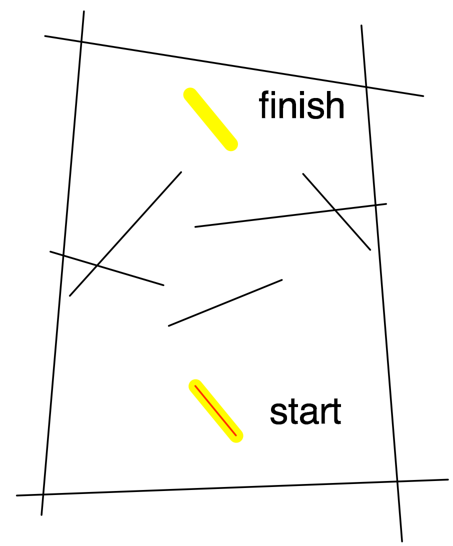
Reduction

3-Sum Given $f : [n] \to \mathbb{N}$ find $a, b, c \in [n]$ such that $f(a) + f(b) + f(c) = 0$

[Gajentaan,Overmars,95] reduces to $\downarrow$







Complexity

- classically $O(n^2)$, in the algebraic decision tree $\Omega(n^2)$
- quantumly $O(n \log n)$, in the query model $\Omega(n^{2/3})$

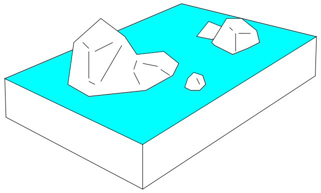
Directions of research

- Come up with a quantum version of the algebraic decision tree model
- Find out the quantum query complexity

Finding the minimum

Find i such that $f(i)$ is minimum costs $\Theta(\sqrt{N})$ queries to f
[Dürr,Høyer,1997]

The algorithm



W.l.o.g. suppose that f is a permutation on $[N]$

Non-halting Algorithm A

- Choose uniformly $y \in [N]$.
- Repeat until *saint glin-glin*
  - Search an element x such that $f(x) < f(y)$
    use the version of Grover's algorithm which suceeds in expected time $O(\sqrt{N/(r-1)})$ where r is the rank of $f(y)$ and runs forever if the rank is 1.
  - Set $y \leftarrow x$

Final algorithm

- Let e be the expected total number of queries to f until $f(y)$ is the solution
- Algorithm A': Interrupt A after $2e$ total queries to f and return the current value of y.
- success probability of A' is at least $1/2$.

Now let's find out what e is. . .

Analysis

def Let p_r be the probability that at some moment in the execution of A $f(y)$ has rank r.

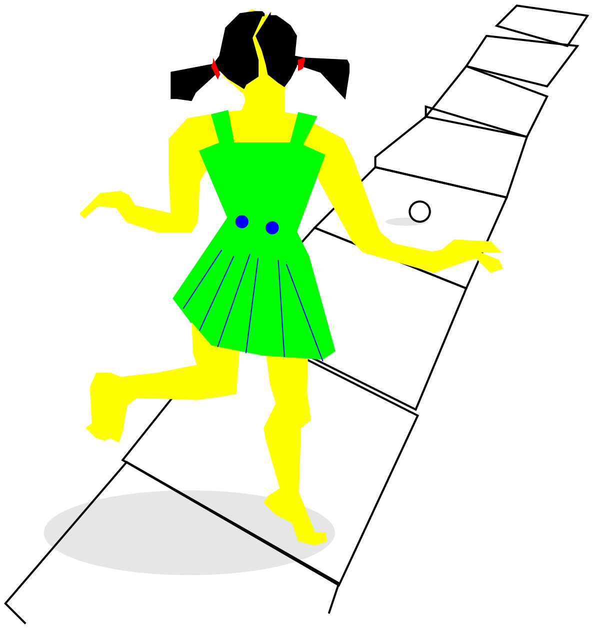
facts $p_N = 1/N$, $p_1 = 1$.

claim $p_r = 1/r$

proof The first moment y becomes such that $f(y) \leq r$ it is choosen unformly (property of Grover's algorithm)

$$e \leq \sum_{r=2}^{N} \frac{1}{r} c \sqrt{N/(r-1)} = O(\sqrt{N})$$

and we are done.



Extension to more functions

- Suppose we have d functions $f_1 : [N_1] \to \mathbb{N}, \ldots f_d : [N_d] \to \mathbb{N}$ and wish to compute $(i_1, \ldots, i_d)$ such that with probability $\geq 1/2$, $f_1(i_1), \ldots, f_d(i_d)$ are all minima.

- Then we if we call d times A' (with $\log d$ repetitions to succeed each with probability $\geq 1 - 1/2d$) it would cost $O(\log d \sum_j \sqrt{N_j})$.

- There is an algorithm which does this with $O(\sqrt{dN})$ queries where $N = \sum_j N_j$.

Algorithm

def $S = \{(j, i) : j \in [d], i \in [N_j]\}$

- Choose uniformly $y = (i_1, \ldots, i_d) \in [N_1] \times \cdots \times [N_d]$
- Repeat until *saint glin-glin*
  - Search $(i, j) \in S$ such that $f_j(i) < f_j(i_j)$
  - Set $i_j \leftarrow i$

Application : minimum spanning tree

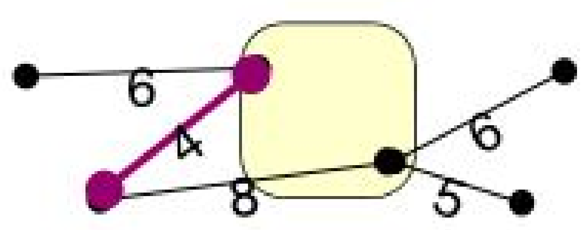
[Dürr,Heiligman,Høyer,Mhalla,04]

- Given a connected graph $G(V, E)$, $w : E \rightarrow \mathbb{N}$ find a spanning tree A (maximal cycle-free edge-set) with minimum total weight $\sum_{e \in A} w(e)$.
- Application: find cheapest telephone network, or for a $2/3$ approximation for the Traveling Salesman Problem.

Standard approach

W.l.o.g suppose all edge weights are different

- Start with empty edge set A, and each vertex in its own component
- Search for every component C the cheapest border edge $e \in E \cap C \times \overline{C}$ such that $w(e)$ is minimal

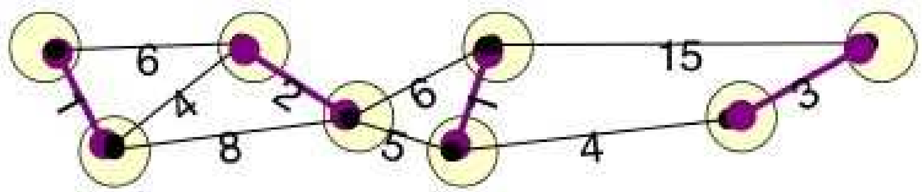


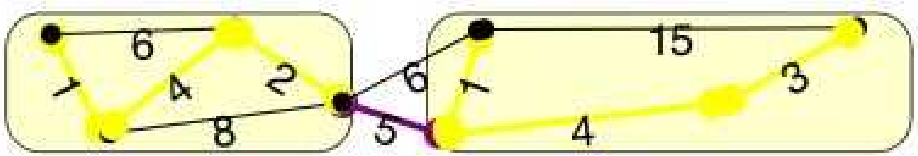
- Add these edges to A, and merge components connected by the new edges.
- repeat at most $\log_2 n$ times

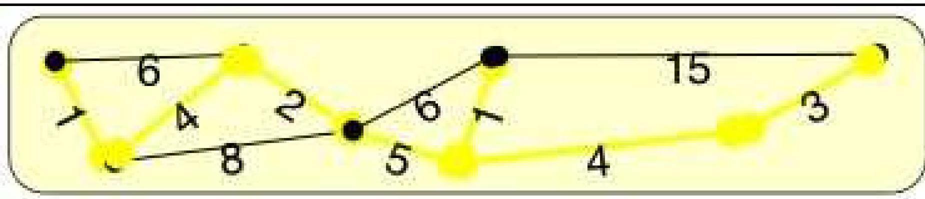
Algorithm

- We consider the adjacency table ($\sim$ list) query model, where the input is a function $f : [m] \rightarrow E$.
- If there are d components, the minima search procedure cost $O(\sqrt{dm})$ queries.
- For the i-th iteration repeat $i + 1$ times to get error probability down to $1/2^{i+1}$, which makes $O((i + 1)\sqrt{(n/i)m})$ queries to f

Overall picture







err. prob.	# queries
1/4	$2\sqrt{nm}$
1/8	$3\sqrt{(n/2)m}$
1/16	$4\sqrt{(n/4)m}$
$\vdots$	
$\leq 1/2$	$O(\sqrt{nm})$

Other results on graph problems

Problem	adj. matrix model		adj. table model	
Minimum spanning tree	$\Theta(n^{3/2})$		$\Theta(\sqrt{nm})$	
Connectivity	$\Theta(n^{3/2})$		$\Theta(n)$	
Strong connectivity	$\Theta(n^{3/2})$		$\Omega(\sqrt{nm})$	$O(\sqrt{nm \log n})$
Shortest paths	$\Omega(n^{3/2})$	$O(n^{3/2} \log^2 n)$	$\Omega(\sqrt{nm})$	$O(\sqrt{nm} \log^2 n)$
2-colorability	$\Omega(n^{3/2})$	$O(n^{3/2})$	$\Theta(n)$	
Triangle membership	$\Omega(n)$	$O(n^{1.3})$		
Perfect matching	$\Omega(n^{3/2})$			

Insertion in an ordered table

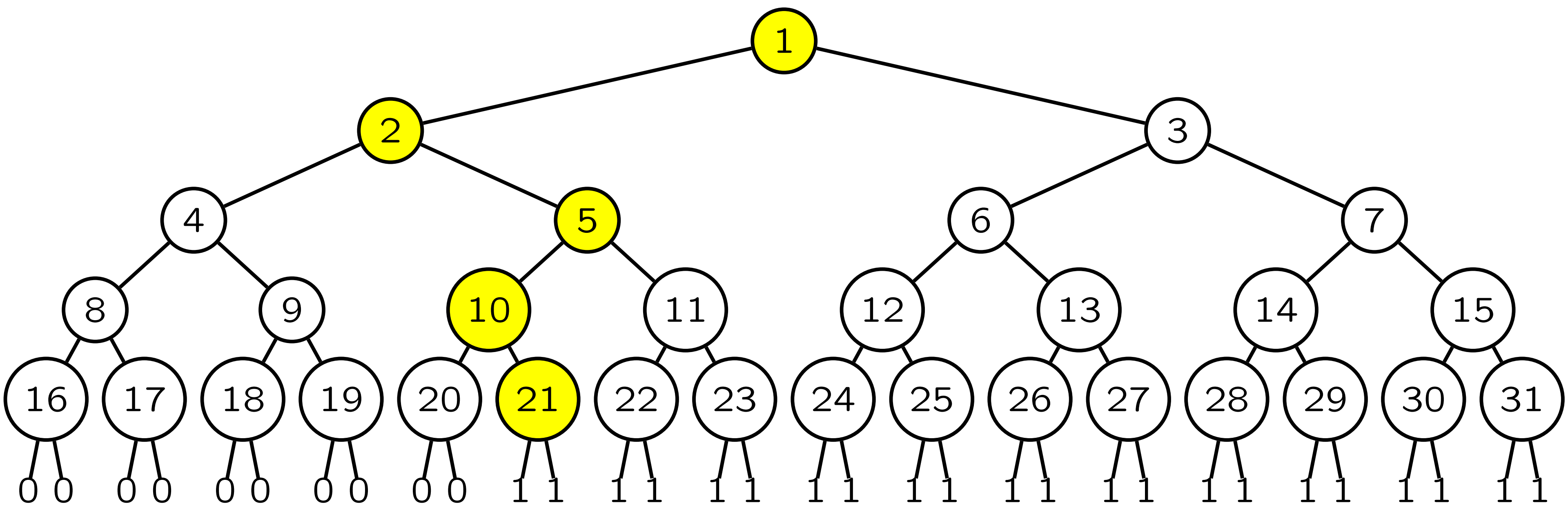
History of bounds on the query complexity for the deterministic case

- $\geq \sqrt{\log N}$ [Buhrman,deWolf,1998]
- $\geq \log_2 N/(2\log_2 \log_2 N)$ [Fahri..1998]
- $\geq \frac{1}{12}\log_2 N = 0,083\log_2 N$ [Ambainis,1999]
- $\geq \frac{1}{\pi}\ln N = 0,22\log_2 N$ [Høyer,Neerbek,2001]
- $\leq 3\log_{52} N = 0,526\log_2 N$ [Fahri..1999]
- $\leq \log_3(N) = 0,631\log_2 N$ [Høyer,Neerbek,2001]

Recall: classical binary research

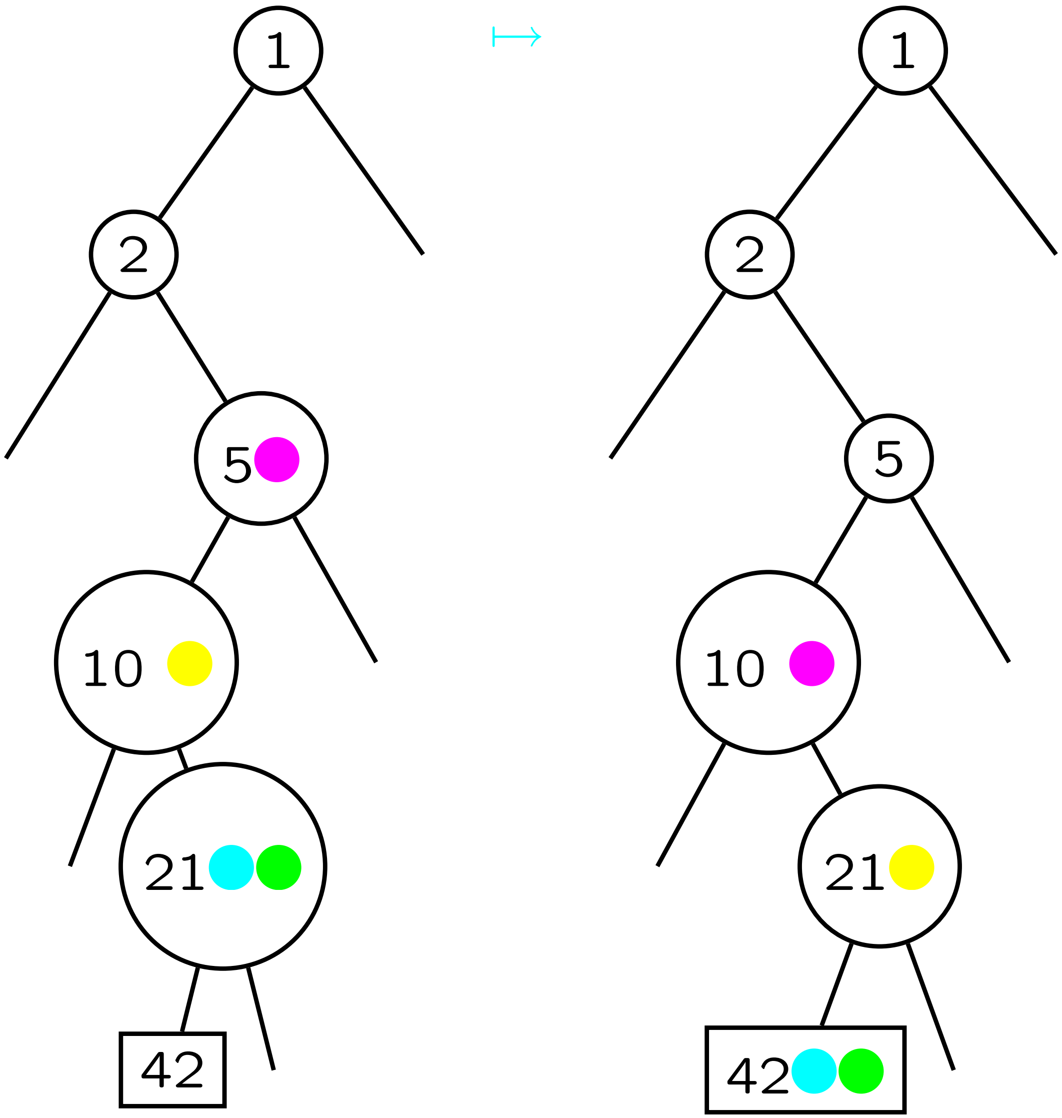
Query : $T_{lr*}[i]$ = value of the rightmost leaf of the left subtree

Algorithm: start with $i = 1$, while i is not a leaf $i \leftarrow 2i + \overline{T_{lr*}[i]}$



$\log_2 N$ queries is optimal, since k queries permit only to distinguish 2^k different input functions

Quantum version

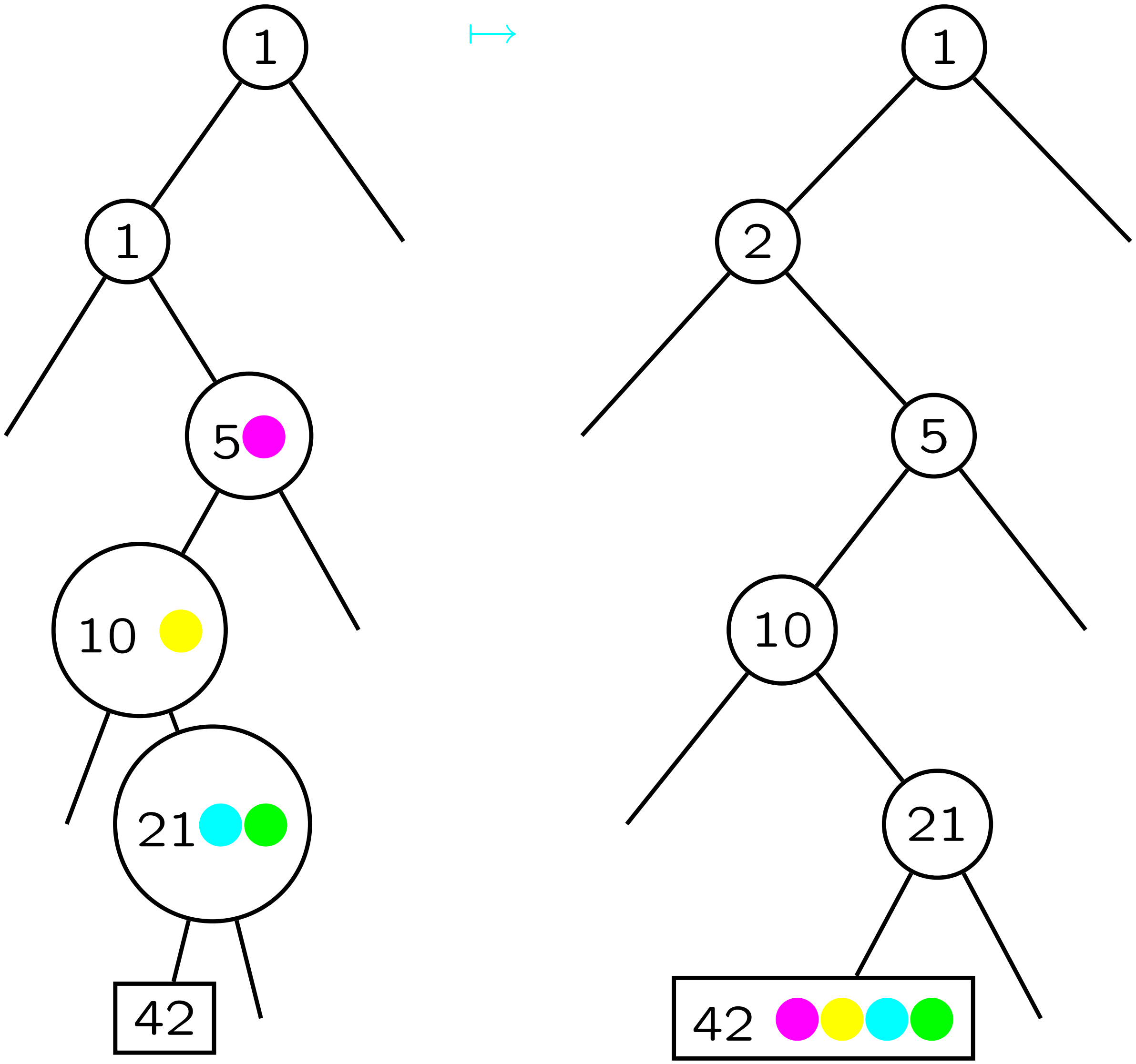


Let $M : |i\rangle \mapsto |2i + \overline{T_{lr*}[i]}\rangle$. M makes a single query to T.

Applied in superposition :

$M(|5\rangle + |10\rangle + \sqrt{2}|21\rangle) = (|10\rangle + |21\rangle + \sqrt{2}|42\rangle)$

Inverse Haar transform

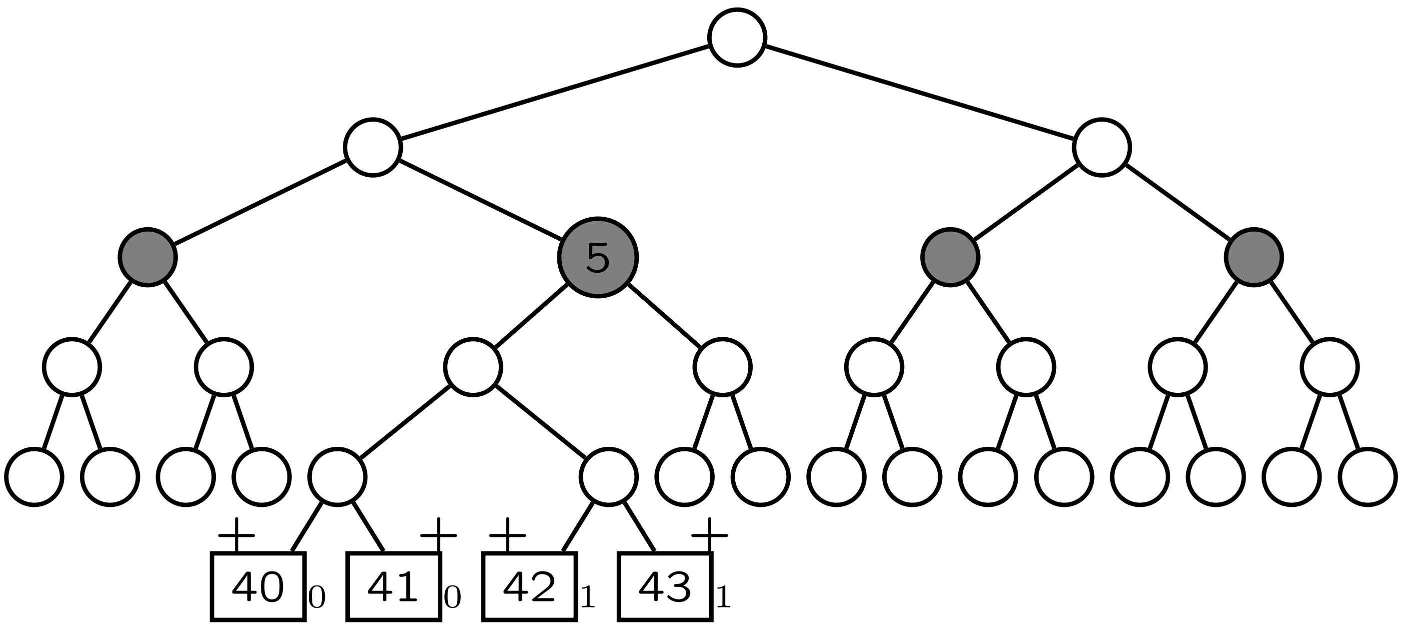


Let U be an operator (which also makes a single query to T). which behaves like : $U(|5\rangle + |10\rangle + \sqrt{2}|21\rangle) = \sqrt{4}|42\rangle$.

It is this operator which gives the quantum acceleration

Définition U applied on border nodes

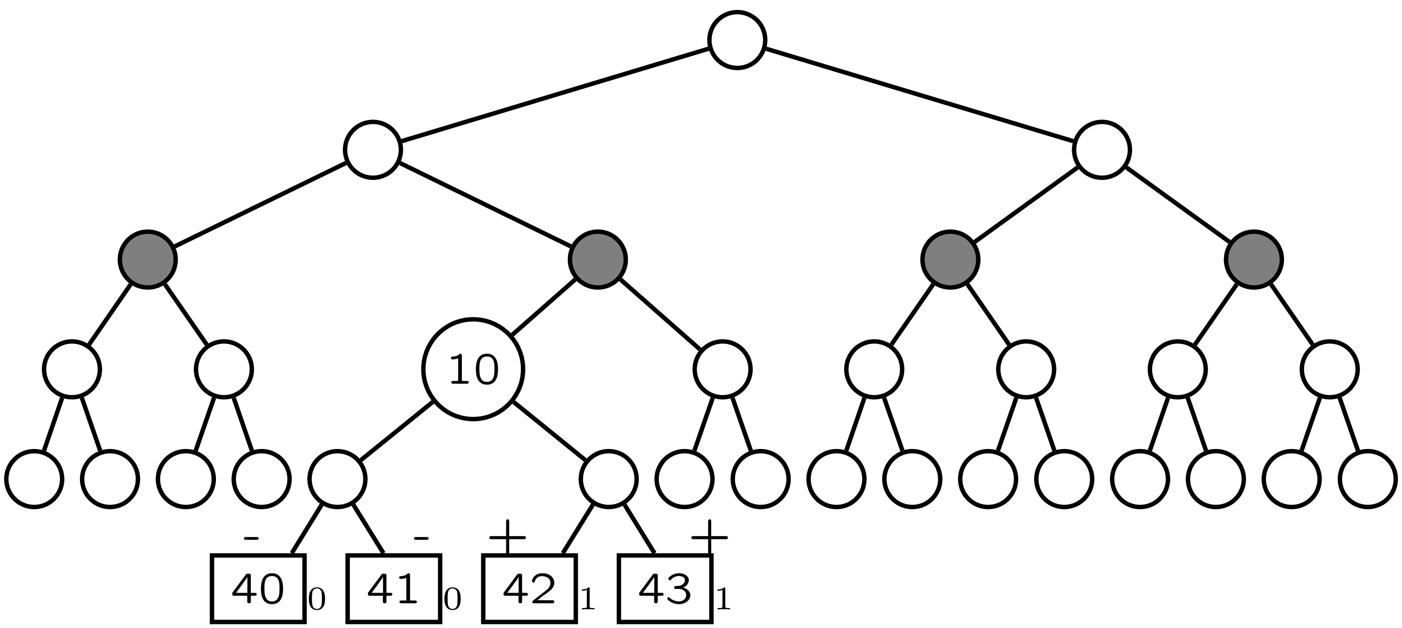
Let there be a level called the *border*. Then if i is a border node, $U|i\rangle =$ is the uniform superposition on the leafs of the good subtree



$$U|5\rangle = |40\rangle + |41\rangle + |42\rangle + |43\rangle$$

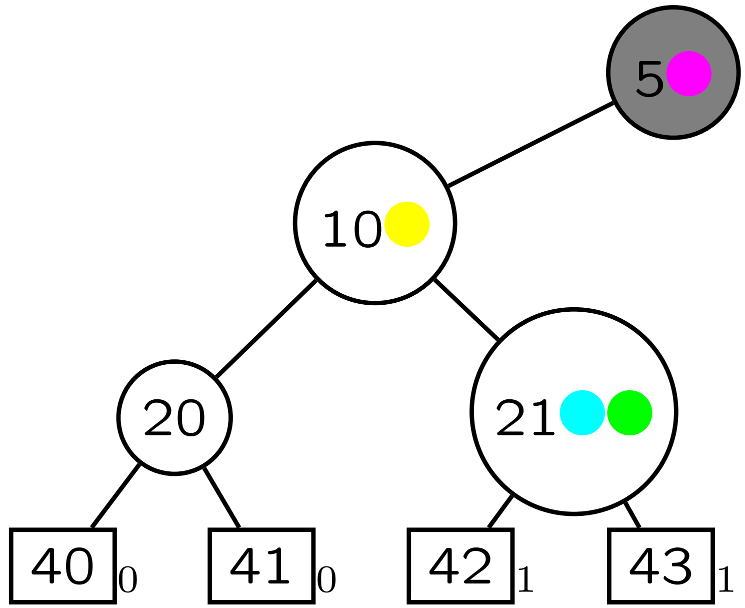
Definition U applied on underborder nodes

If i is a node under the border, then $U|i\rangle =$ $(-1)^{\overline{T_{lr*}[i]}}$(uniform superposition of the leafs of the left subtree - uniform superposition of the leafs of the right subtree)



$U|10\rangle = -|40\rangle - |41\rangle + |42\rangle + |43\rangle$

Interference scheme



contribution:	+	+	+	+
	-	-	+	+
			+	-
			+	-
final amplitudes	0	0	1	0

A single call to U is enough to the solution exacty, if is applied on the correct superposition.

How can we produce the required superposition ?

[Haha. . .]

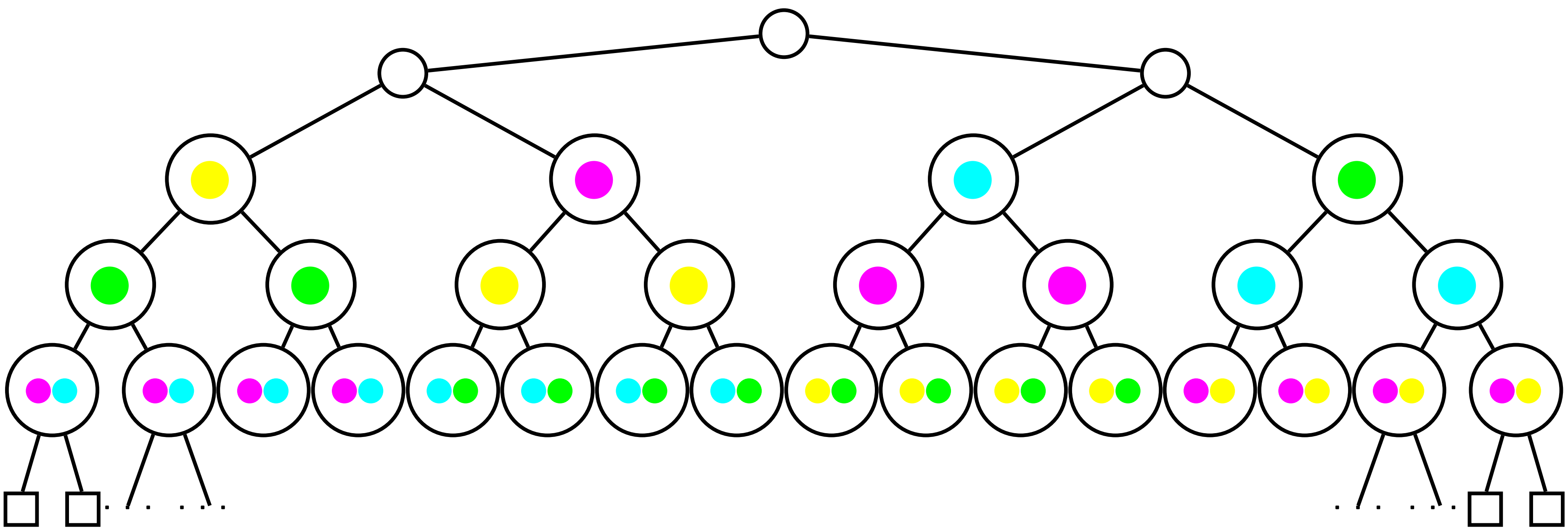
A distribution of colored pebbles on nodes (which are not leafs)

satisfying :

(A) on every path from the root to a leaf there is exactly one pebble from each color

(B) the number of pebbles in a node (except on the border) is the total number of pebbles of his ancestors

Definition The border is just the first level containing pebbles



The algorithm

We have two registers: one containing a color, the other containing a node number.

1. put the first register in superposition on the colors

$$(|{\color{magenta}\bullet}\rangle + |{\color{yellow}\bullet}\rangle + |{\color{cyan}\bullet}\rangle + |{\color{green}\bullet}\rangle) \otimes |0\rangle$$

2. put in the second register the number of the unique node of the good path containing the pebble of this color

$$|{\color{magenta}\bullet}\rangle|5\rangle + |{\color{yellow}\bullet}\rangle|10\rangle + |{\color{cyan}\bullet}\rangle|21\rangle + |{\color{green}\bullet}\rangle|21\rangle$$

3. uncolor the first register

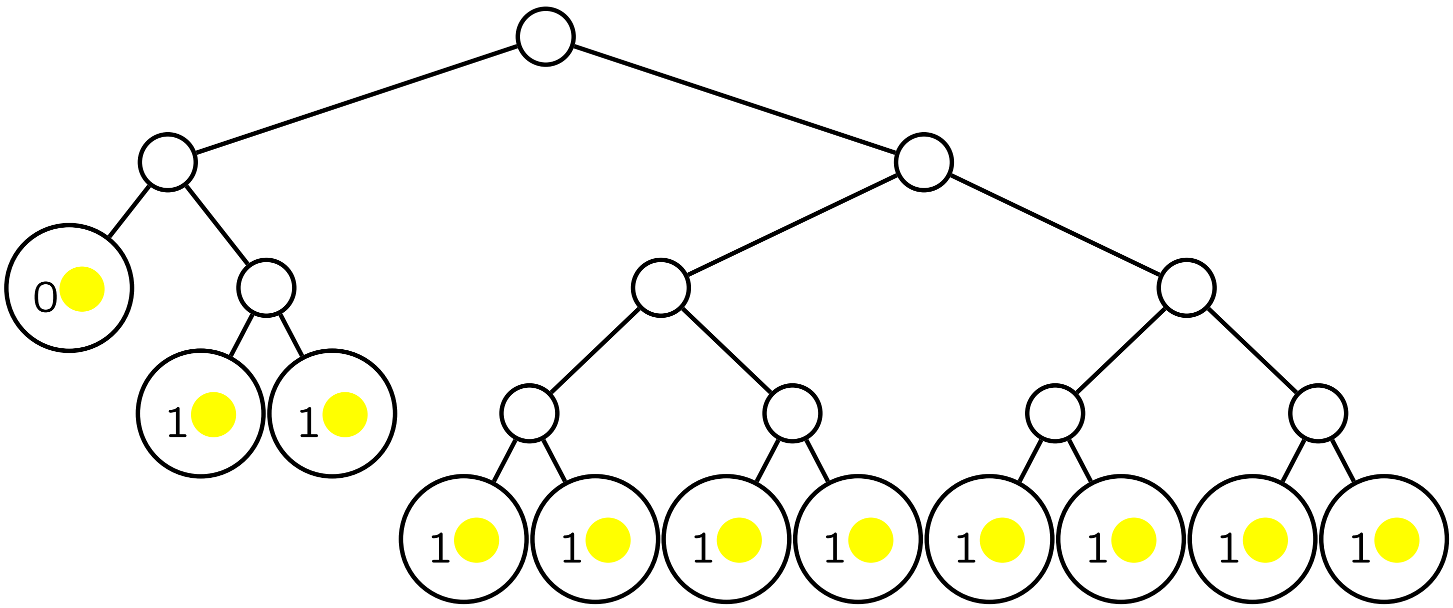
$$|0\rangle \otimes (|5\rangle + |10\rangle + \sqrt{2}|21\rangle)$$

4. apply U on the second register

$$|0\rangle|42\rangle$$

The recursion

Among all nodes containing a pebble of a fixed color finding the unique node on the good path comes to finding the first node i such that $T_{r*}[i] = 1$. ($\neq T_{lr^*}$!)



Sounds familiar?

Size of the new table $N/3 + O(\log N)$

$\rightarrow$ Complexity $\log_3 N + O(1)$