Quantum pattern matching fast on average

Ashley Montanaro

Department of Computer Science, University of Bristol, UK

12 January 2015

University of BRISTOL

EPSRC
Engineering and Physical Sciences Research Council

Pattern matching

In the traditional pattern matching problem, we seek to find a pattern $P : [m] \to \Sigma$ within a text $T : [n] \to \Sigma$.

$$T = \boxed{Q}\boxed{U}\boxed{A}\boxed{N}\boxed{T}\boxed{U}\boxed{M} \qquad P = \boxed{A}\boxed{N}\boxed{T}$$

Pattern matching

In the traditional pattern matching problem, we seek to find a pattern $P : [m] \to \Sigma$ within a text $T : [n] \to \Sigma$.

$$T = \begin{array}{|c|c|c|c|c|c|c|}\hline \mathrm{Q} & \mathrm{U} & \mathrm{A} & \mathrm{N} & \mathrm{T} & \mathrm{U} & \mathrm{M} \\ \hline \end{array} \qquad P = \begin{array}{|c|c|c|}\hline \mathrm{A} & \mathrm{N} & \mathrm{T} \\ \hline \end{array}$$

Pattern matching

In the traditional pattern matching problem, we seek to find a pattern $P : [m] \to \Sigma$ within a text $T : [n] \to \Sigma$.

$T =$

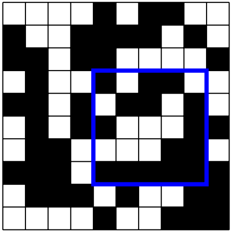
Q	U	A	N	T	U	M

$P =$

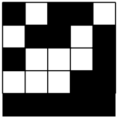
A	N	T

We can generalise this to higher dimensions d, where $P : [m]^d \to \Sigma$ and $T : [n]^d \to \Sigma$:

$T =$



$P =$



Pattern matching

Focusing on the 1-dimensional problem for now:

- Classically, it is known that this problem can be solved in worst-case time $O(n+m)$ [Knuth, Morris and Pratt '77].

Pattern matching

Focusing on the 1-dimensional problem for now:

- Classically, it is known that this problem can be solved in worst-case time $O(n+m)$ [Knuth, Morris and Pratt '77].
- There is a quantum algorithm which solves this problem (with bounded failure probability) in time $\widetilde{O}(\sqrt{n}+\sqrt{m})$ [Ramesh and Vinay '03].

Pattern matching

Focusing on the 1-dimensional problem for now:

- Classically, it is known that this problem can be solved in worst-case time $O(n + m)$ [Knuth, Morris and Pratt '77].
- There is a quantum algorithm which solves this problem (with bounded failure probability) in time $\widetilde{O}(\sqrt{n} + \sqrt{m})$ [Ramesh and Vinay '03].

Both these bounds are optimal in the worst case. But… what about the average case?

Pattern matching

Focusing on the 1-dimensional problem for now:

- Classically, it is known that this problem can be solved in worst-case time $O(n+m)$ [Knuth, Morris and Pratt '77].
- There is a quantum algorithm which solves this problem (with bounded failure probability) in time $\widetilde{O}(\sqrt{n}+\sqrt{m})$ [Ramesh and Vinay '03].

Both these bounds are optimal in the worst case. But… what about the average case?

Consider a simple model where each character of T is picked uniformly at random from Σ, and either:

- P is chosen to be an arbitrary substring of T; or
- P is uniformly random.

Could this be easier?

Pattern matching

- Classically, one can solve the average-case problem in time $\widetilde{O}(n/m + \sqrt{n})$, and this is optimal.

Pattern matching

- Classically, one can solve the average-case problem in time $\widetilde{O}(n/m + \sqrt{n})$, and this is optimal.
- But in the quantum setting, we have the following result:

Theorem (modulo minor technicalities)

Let $T : [n] \to \Sigma$, $P : [m] \to \Sigma$ be picked as on the previous slide. Then there is a quantum algorithm which runs in time

$$\widetilde{O}(\sqrt{n/m}\, 2^{O(\sqrt{\log m})})$$

and determines whether P matches T.

Pattern matching

- Classically, one can solve the average-case problem in time $\widetilde{O}(n/m + \sqrt{n})$, and this is optimal.
- But in the quantum setting, we have the following result:

Theorem (modulo minor technicalities)

Let $T : [n] \to \Sigma$, $P : [m] \to \Sigma$ be picked as on the previous slide. Then there is a quantum algorithm which runs in time

$$\widetilde{O}(\sqrt{n/m}\, 2^{O(\sqrt{\log m})})$$

and determines whether P matches T. If P does match T, the algorithm also outputs the position at which the match occurs.

Pattern matching

- Classically, one can solve the average-case problem in time $\widetilde{O}(n/m + \sqrt{n})$, and this is optimal.
- But in the quantum setting, we have the following result:

Theorem (modulo minor technicalities)

Let $T : [n] \to \Sigma$, $P : [m] \to \Sigma$ be picked as on the previous slide. Then there is a quantum algorithm which runs in time

$$\widetilde{O}(\sqrt{n/m}\, 2^{O(\sqrt{\log m})})$$

and determines whether P matches T. If P does match T, the algorithm also outputs the position at which the match occurs. The algorithm fails with probability $O(1/n)$, taken over both the choice of T and P, and its internal randomness.

Pattern matching

- Classically, one can solve the average-case problem in time $\widetilde{O}(n/m + \sqrt{n})$, and this is optimal.
- But in the quantum setting, we have the following result:

Theorem (modulo minor technicalities)

Let $T : [n] \to \Sigma$, $P : [m] \to \Sigma$ be picked as on the previous slide. Then there is a quantum algorithm which runs in time

$$\widetilde{O}(\sqrt{n/m}\, 2^{O(\sqrt{\log m})})$$

and determines whether P matches T. If P does match T, the algorithm also outputs the position at which the match occurs. The algorithm fails with probability $O(1/n)$, taken over both the choice of T and P, and its internal randomness.

This is a super-polynomial speedup for large m.

Pattern matching (d-dimensional)

- Classically, one can solve the average-case problem in time $\widetilde{O}((n/m)^d + n^{d/2})$, and this is optimal.
- But in the quantum setting, we have the following result:

Theorem (modulo minor technicalities)

Let $T : [n]^d \to \Sigma$, $P : [m]^d \to \Sigma$ be picked as on the previous slide. Then there is a quantum algorithm which runs in time

$$\widetilde{O}((n/m)^{d/2}\, 2^{O(d^{3/2}\sqrt{\log m})})$$

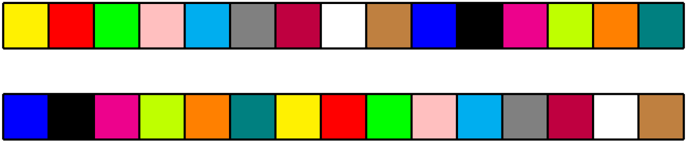
and determines whether P matches T. If P does match T, the algorithm also outputs the position at which the match occurs. The algorithm fails with probability $O(1/n^d)$, taken over both the choice of T and P, and its internal randomness.

This is a super-polynomial speedup for large m.

The dihedral hidden subgroup problem

The main quantum ingredient in the algorithm is an algorithm for the dihedral hidden subgroup problem (aka finding hidden shifts over $\mathbb{Z}_N$):

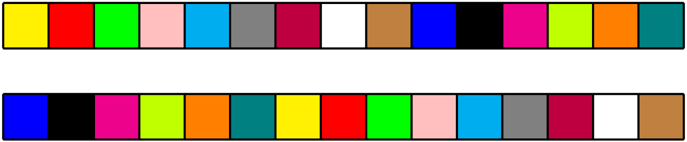
- Given two injective functions $f, g : \mathbb{Z}_N \to X$ such that $g(x) = f(x+s)$ for some $s \in \mathbb{Z}_N$, determine s.



The dihedral hidden subgroup problem

The main quantum ingredient in the algorithm is an algorithm for the dihedral hidden subgroup problem (aka finding hidden shifts over $\mathbb{Z}_N$):

- Given two injective functions $f, g : \mathbb{Z}_N \to X$ such that $g(x) = f(x+s)$ for some $s \in \mathbb{Z}_N$, determine s.



- The best known quantum algorithm for the dihedral HSP uses $2^{O(\sqrt{\log N})} = o(N^{\epsilon})$ queries [Kuperberg ’05].
- Classically, there is a lower bound of $\Omega(\sqrt{N})$ queries.

From the dihedral HSP to pattern matching

Can we treat f and g as text and pattern, and use the dihedral HSP to solve the general pattern matching problem?

From the dihedral HSP to pattern matching

Can we treat f and g as text and pattern, and use the dihedral HSP to solve the general pattern matching problem?

The dihedral HSP algorithm requires the pattern and text to be…

- injective
- the same length
- 1-dimensional

Also, a different notion of shifts is used (modulo N).

From the dihedral HSP to pattern matching

Can we treat f and g as text and pattern, and use the dihedral HSP to solve the general pattern matching problem?

The dihedral HSP algorithm requires the pattern and text to be...

- injective
- the same length
- 1-dimensional

Also, a different notion of shifts is used (modulo N).

Can we relax these assumptions?

From the dihedral HSP to pattern matching

First, we make the pattern and text injective by concatenating characters (an idea used previously in some different contexts [Knuth '77, Gharibi '13]):

Q	U	A	N	T	U	M

$\longrightarrow$

QU	UA	AN	NT	TU	UM

A	N	T

$\longrightarrow$

AN	NT

From the dihedral HSP to pattern matching

First, we make the pattern and text injective by concatenating characters (an idea used previously in some different contexts [Knuth '77, Gharibi '13]):

Q	U	A	N	T	U	M	$\longrightarrow$	QU	UA	AN	NT	TU	UM
		A	N	T			$\longrightarrow$			AN	NT		

- Concatenation preserves the property of the pattern matching the text.

From the dihedral HSP to pattern matching

First, we make the pattern and text injective by concatenating characters (an idea used previously in some different contexts [Knuth '77, Gharibi '13]):

Q	U	A	N	T	U	M	$\longrightarrow$	QU	UA	AN	NT	TU	UM

A	N	T	$\longrightarrow$	AN	NT

- Concatenation preserves the property of the pattern matching the text.
- If we produce a new alphabet whose symbols are strings of length k, a query to the new string can be simulated by k queries to the original string.

From the dihedral HSP to pattern matching

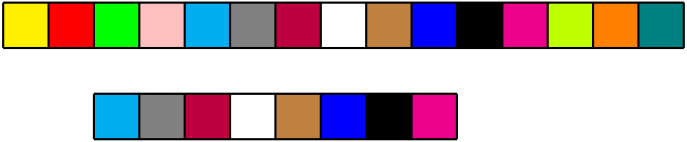
First, we make the pattern and text injective by concatenating characters (an idea used previously in some different contexts [Knuth '77, Gharibi '13]):

Q	U	A	N	T	U	M	$\longrightarrow$	QU	UA	AN	NT	TU	UM
		A	N	T			$\longrightarrow$			AN	NT		

- Concatenation preserves the property of the pattern matching the text.
- If we produce a new alphabet whose symbols are strings of length k, a query to the new string can be simulated by k queries to the original string.
- For most random strings, it suffices to take $k = O(\log n)$.

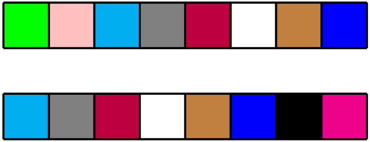
From the dihedral HSP to pattern matching

Second, we apply the dihedral HSP algorithm to the (now injective) pattern and text, at a randomly chosen offset.



From the dihedral HSP to pattern matching

Second, we apply the dihedral HSP algorithm to the (now injective) pattern and text, at a randomly chosen offset.



From the dihedral HSP to pattern matching

Second, we apply the dihedral HSP algorithm to the (now injective) pattern and text, at a randomly chosen offset.

Claim

If the pattern is contained in the text, and our guess for the start of the pattern is correct to within distance $m\,2^{-O(\sqrt{\log m})}$, the dihedral HSP algorithm outputs the correct shift with high probability.

Completing the argument ($d = 1$)

- The probability of our guess being in this "good" range is

$$p = \Omega(m\, 2^{-O(\sqrt{\log m})}/n).$$

Completing the argument ($d = 1$)

- The probability of our guess being in this "good" range is

$$p = \Omega(m\, 2^{-O(\sqrt{\log m})}/n).$$

- Using a variant of amplitude amplification which can cope with a bounded-error verifier [Høyer et al. '03], we can find a "good" position of this kind using

$$O(1/\sqrt{p}) = O(\sqrt{n/m}\, 2^{O(\sqrt{\log m})})$$

queries.

Completing the argument ($d = 1$)

- The probability of our guess being in this "good" range is

$$p = \Omega(m\, 2^{-O(\sqrt{\log m})}/n).$$

- Using a variant of amplitude amplification which can cope with a bounded-error verifier [Høyer et al. '03], we can find a "good" position of this kind using

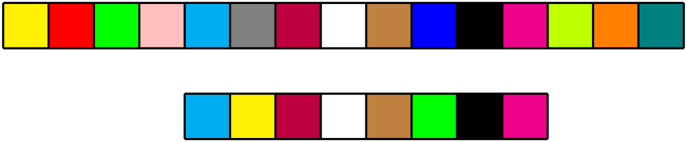
$$O(1/\sqrt{p}) = O(\sqrt{n/m}\, 2^{O(\sqrt{\log m})})$$

queries.

- The time complexity is the same up to log factors.

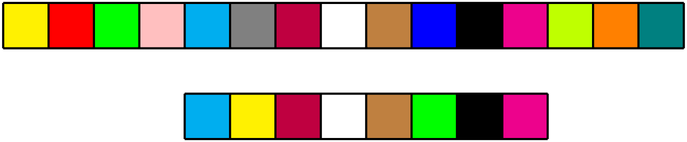
Dealing with errors

Note that the dihedral HSP algorithm might incorrectly claim a match if the pattern does not match the text, but almost matches at some offset:



Dealing with errors

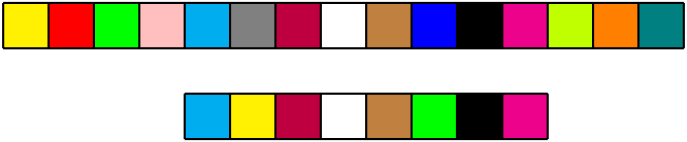
Note that the dihedral HSP algorithm might incorrectly claim a match if the pattern does not match the text, but almost matches at some offset:



We deal with this by checking any claimed match using Grover's algorithm.

Dealing with errors

Note that the dihedral HSP algorithm might incorrectly claim a match if the pattern does not match the text, but almost matches at some offset:

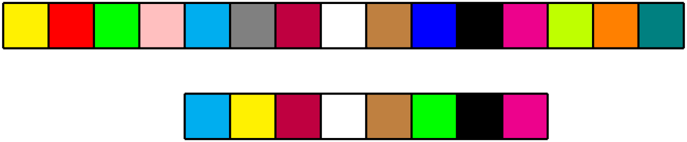


We deal with this by checking any claimed match using Grover's algorithm.

- This gives us an $O(1/\sqrt{\gamma})$ term in the runtime, where γ is the minimal fraction of positions where a non-matching pattern differs from the text.

Dealing with errors

Note that the dihedral HSP algorithm might incorrectly claim a match if the pattern does not match the text, but almost matches at some offset:

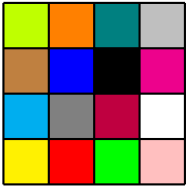


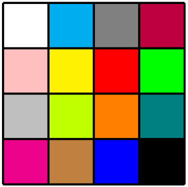
We deal with this by checking any claimed match using Grover's algorithm.

- This gives us an $O(1/\sqrt{\gamma})$ term in the runtime, where γ is the minimal fraction of positions where a non-matching pattern differs from the text.
- For most random patterns and texts, $\gamma = \Omega(1)$.

Higher dimensions

The dihedral HSP algorithm has allowed us to solve the case $d = 1$. Can we generalise this to higher d?

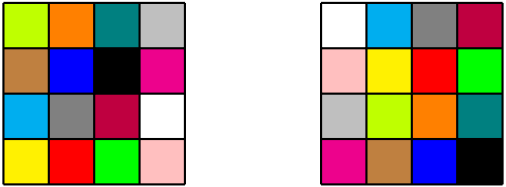




Now a hidden shift s becomes a d-tuple $(s_1, \ldots, s_d)$.

Higher dimensions

The dihedral HSP algorithm has allowed us to solve the case $d = 1$. Can we generalise this to higher d?



Now a hidden shift s becomes a d-tuple $(s_1, \ldots, s_d)$. When the input size is a power of 2, we have:

Theorem

Let $f, g : \mathbb{Z}_{2^n}^d \to X$ be injective functions such that $g(x) = f(x+s)$ for all $x \in \mathbb{Z}_{2^n}^d$. There is a quantum algorithm which outputs s with bounded error using $O(n2^{1.781\ldots\sqrt{dn}})$ queries.

Higher dimensions

The plan is to generalise an idea from [Kuperberg '05]:

Higher dimensions

The plan is to generalise an idea from [Kuperberg '05]:

1. Generate a large pool of states

$$|\psi_r\rangle = \frac{1}{\sqrt{2}}\left(|0\rangle + \omega^{rs}|1\rangle\right),$$

where $\omega := e^{\pi i/2^{n-1}}$, for random $r \in \mathbb{Z}_{2^n}$ (can be done by querying f and g in superposition and using the QFT).

Higher dimensions

The plan is to generalise an idea from [Kuperberg '05]:

1. Generate a large pool of states

$$|\psi_r\rangle = \frac{1}{\sqrt{2}}\left(|0\rangle + \omega^{rs}|1\rangle\right),$$

where $\omega := e^{\pi i/2^{n-1}}$, for random $r \in \mathbb{Z}_{2^n}$ (can be done by querying f and g in superposition and using the QFT).

2. Attempt to produce the state

$$|\psi_{2^{n-1}}\rangle = \frac{1}{\sqrt{2}}\left(|0\rangle + (-1)^s|1\rangle\right),$$

from which the low-order bit s_n can be determined.

Higher dimensions

The plan is to generalise an idea from [Kuperberg '05]:

1. Generate a large pool of states

$$|\psi_r\rangle = \frac{1}{\sqrt{2}}\left(|0\rangle + \omega^{rs}|1\rangle\right),$$

where $\omega := e^{\pi i/2^{n-1}}$, for random $r \in \mathbb{Z}_{2^n}$ (can be done by querying f and g in superposition and using the QFT).

2. Attempt to produce the state

$$|\psi_{2^{n-1}}\rangle = \frac{1}{\sqrt{2}}\left(|0\rangle + (-1)^s|1\rangle\right),$$

from which the low-order bit s_n can be determined.

Once we know s_n, we can apply this idea to new functions f', g' to learn the other bits of s.

Producing better states

Step 2 uses a combination operation:

$$(|\psi_r\rangle, |\psi_t\rangle) \mapsto \begin{cases} |\psi_{r+t}\rangle \text{ (bad)} \\ |\psi_{r-t}\rangle \text{ (good)} \end{cases}$$

with equal probability of each.

Producing better states

Step 2 uses a combination operation:

$$(|\psi_r\rangle, |\psi_t\rangle) \mapsto \begin{cases} |\psi_{r+t}\rangle \text{ (bad)} \\ |\psi_{r-t}\rangle \text{ (good)} \end{cases}$$

with equal probability of each.

Split the $n-1$ low-order bits into equal-sized blocks of $O(\sqrt{n})$ bits each:

- If r and t's low-order bits were equal in some block, in the good case $(r-t)$'s bits will be zero in that block.

Producing better states

Step 2 uses a combination operation:

$$(|\psi_r\rangle, |\psi_t\rangle) \mapsto \begin{cases} |\psi_{r+t}\rangle \text{ (bad)} \\ |\psi_{r-t}\rangle \text{ (good)} \end{cases}$$

with equal probability of each.

Split the $n-1$ low-order bits into equal-sized blocks of $O(\sqrt{n})$ bits each:

- If r and t's low-order bits were equal in some block, in the good case $(r-t)$'s bits will be zero in that block.
- In the bad case, we discard the output state $|\psi_{r+t}\rangle$.

Producing better states

Step 2 uses a combination operation:

$$(|\psi_r\rangle, |\psi_t\rangle) \mapsto \begin{cases} |\psi_{r+t}\rangle \text{ (bad)} \\ |\psi_{r-t}\rangle \text{ (good)} \end{cases}$$

with equal probability of each.

Split the $n-1$ low-order bits into equal-sized blocks of $O(\sqrt{n})$ bits each:

- If r and t's low-order bits were equal in some block, in the good case $(r-t)$'s bits will be zero in that block.
- In the bad case, we discard the output state $|\psi_{r+t}\rangle$.
- If we start with a pool of $2^{O(\sqrt{n})}$ states $|\psi_r\rangle$, for each block there are many states whose bits are equal, so we have a good chance of producing $|\psi_{2^{n-1}}\rangle$ at the end.

Producing better states

Step 2 uses a combination operation:

$$(|\psi_r\rangle, |\psi_t\rangle) \mapsto \begin{cases} |\psi_{r+t}\rangle \text{ (bad)} \\ |\psi_{r-t}\rangle \text{ (good)} \end{cases}$$

with equal probability of each.

Split the $n-1$ low-order bits into equal-sized blocks of $O(\sqrt{n})$ bits each:

- If r and t's low-order bits were equal in some block, in the good case $(r-t)$'s bits will be zero in that block.
- In the bad case, we discard the output state $|\psi_{r+t}\rangle$.
- If we start with a pool of $2^{O(\sqrt{n})}$ states $|\psi_r\rangle$, for each block there are many states whose bits are equal, so we have a good chance of producing $|\psi_{2^{n-1}}\rangle$ at the end.

Everything turns out to go through for $d > 1\ldots$

Generalising this idea

1. Now $s \in \mathbb{Z}_{2^n}^d$ and the pool of states is of the form

$$|\psi_r\rangle = \frac{1}{\sqrt{2}}\left(|0\rangle + \omega^{r\cdot s}|1\rangle\right),$$

for random $r \in \mathbb{Z}_{2^n}^d$ (produced using the QFT over $\mathbb{Z}_{2^n}^d$).

Generalising this idea

1. Now $s \in \mathbb{Z}_{2^n}^d$ and the pool of states is of the form

$$|\psi_r\rangle = \frac{1}{\sqrt{2}} \left(|0\rangle + \omega^{r \cdot s} |1\rangle \right),$$

   for random $r \in \mathbb{Z}_{2^n}^d$ (produced using the QFT over $\mathbb{Z}_{2^n}^d$).

2. The combination operation works the same way as before.

Generalising this idea

1. Now $s \in \mathbb{Z}_{2^n}^d$ and the pool of states is of the form

$$|\psi_r\rangle = \frac{1}{\sqrt{2}} \left(|0\rangle + \omega^{r \cdot s} |1\rangle \right),$$

   for random $r \in \mathbb{Z}_{2^n}^d$ (produced using the QFT over $\mathbb{Z}_{2^n}^d$).
2. The combination operation works the same way as before.
3. We end up producing states $|\psi_r\rangle$ for random $r \in \{0, 2^{n-1}\}^d$, from which the d low-order bits of s can be found.

Generalising this idea

1. Now $s \in \mathbb{Z}_{2^n}^d$ and the pool of states is of the form

$$|\psi_r\rangle = \frac{1}{\sqrt{2}}\left(|0\rangle + \omega^{r\cdot s}|1\rangle\right),$$

   for random $r \in \mathbb{Z}_{2^n}^d$ (produced using the QFT over $\mathbb{Z}_{2^n}^d$).
2. The combination operation works the same way as before.
3. We end up producing states $|\psi_r\rangle$ for random $r \in \{0, 2^{n-1}\}^d$, from which the d low-order bits of s can be found.

We can improve the runtime of the algorithm of [Kuperberg '05]:

- Adjusting the block size as the algorithm progresses
- Reusing "bad" states, rather than just discarding them

Generalising this idea

1. Now $s \in \mathbb{Z}_{2^n}^d$ and the pool of states is of the form

$$|\psi_r\rangle = \frac{1}{\sqrt{2}}\left(|0\rangle + \omega^{r \cdot s}|1\rangle\right),$$

   for random $r \in \mathbb{Z}_{2^n}^d$ (produced using the QFT over $\mathbb{Z}_{2^n}^d$).
2. The combination operation works the same way as before.
3. We end up producing states $|\psi_r\rangle$ for random $r \in \{0, 2^{n-1}\}^d$, from which the d low-order bits of s can be found.

We can improve the runtime of the algorithm of [Kuperberg '05]:

- Adjusting the block size as the algorithm progresses
- Reusing "bad" states, rather than just discarding them

In the case $d = 1$ we get $\widetilde{O}(2^{1.781\ldots\sqrt{n}})$ rather than $O(2^{3\sqrt{n}})$, matching a more complicated algorithm in [Kuperberg '05].

Summary

- There is a quantum algorithm for the d-dimensional pattern matching problem which is super-polynomially faster than classical for most (long) patterns and texts:

$$\widetilde{O}\big((n/m)^{d/2}\, 2^{O(d^{3/2}\sqrt{\log m})}\big) \quad \text{vs.} \quad \widetilde{\Omega}\big((n/m)^d + n^{d/2}\big).$$

Summary

- There is a quantum algorithm for the d-dimensional pattern matching problem which is super-polynomially faster than classical for most (long) patterns and texts:

$$\widetilde{O}\left((n/m)^{d/2}\, 2^{O\left(d^{3/2}\sqrt{\log m}\right)}\right) \quad \text{vs.} \quad \widetilde{\Omega}\left((n/m)^d + n^{d/2}\right).$$

- For some inputs, the algorithm might fail (claim a match when there is no match)… but when it does, we at least know that the pattern was close to matching at that offset.

Summary

- There is a quantum algorithm for the d-dimensional pattern matching problem which is super-polynomially faster than classical for most (long) patterns and texts:

$$\widetilde{O}\left((n/m)^{d/2}\, 2^{O\left(d^{3/2}\sqrt{\log m}\right)}\right) \quad \text{vs.} \quad \widetilde{\Omega}\left((n/m)^d + n^{d/2}\right).$$

- For some inputs, the algorithm might fail (claim a match when there is no match)… but when it does, we at least know that the pattern was close to matching at that offset.

Interesting open question: Can we find an improved quantum algorithm for the dihedral HSP?

Thanks!

Further reading: arXiv:1408.1816

Advert

- Two postdoc positions available at Bristol to work on the theory of quantum computation.
- Application deadline 25 January; start date flexible.
- Talk to me if you're interested!