Quantum Algorithms
$|f \otimes r\rangle$
One-Sided Crossing Minimization

Susanna Caroppo, Giordano Da Lozzo, Giuseppe Di Battista

Roma Tre University

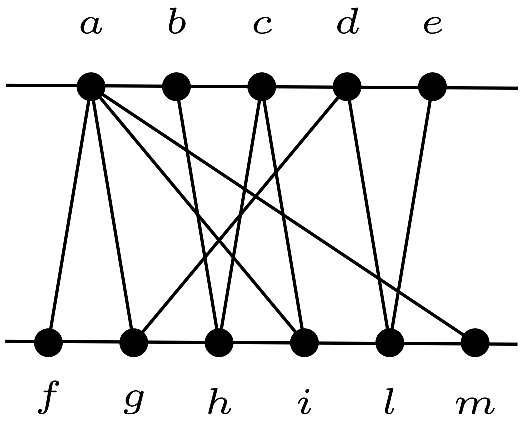
What is the One-Sided Crossing Minimization problem?

What is the One-Sided Crossing Minimization problem?

Input: a bipartite graph $G = (U, V, E)$ with a fixed linear ordering of U

What is the One-Sided Crossing Minimization problem?

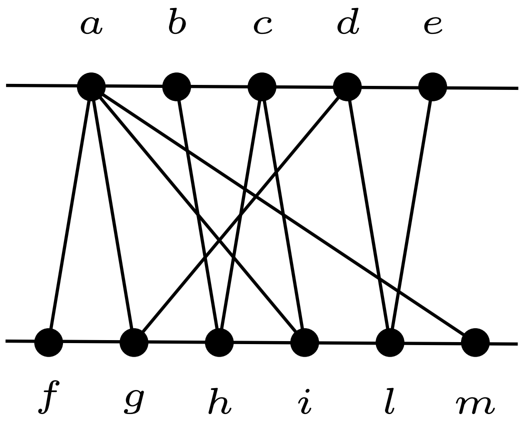
Input: a bipartite graph $G = (U, V, E)$ with a fixed linear ordering of U



What is the One-Sided Crossing Minimization problem?

Input: a bipartite graph $G = (U, V, E)$ with a fixed linear ordering of U

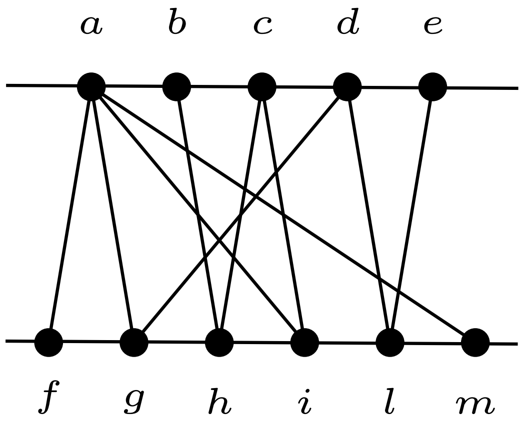
Output: a linear ordering of V such that the number of crossings is minimum

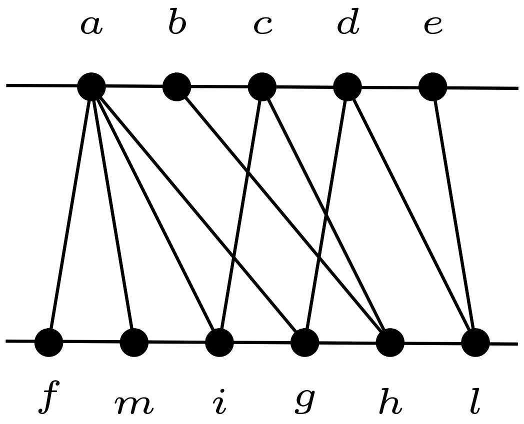


What is the One-Sided Crossing Minimization problem?

Input: a bipartite graph $G = (U, V, E)$ with a fixed linear ordering of U

Output: a linear ordering of V such that the number of crossings is minimum

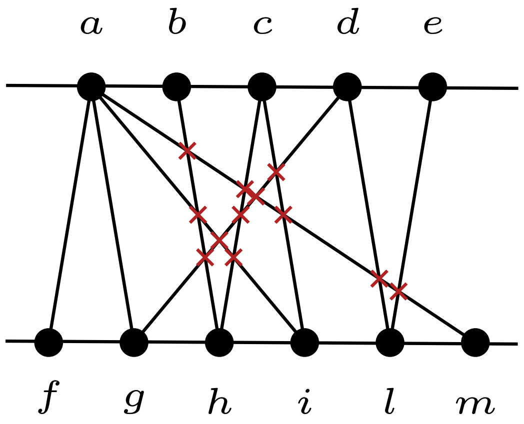




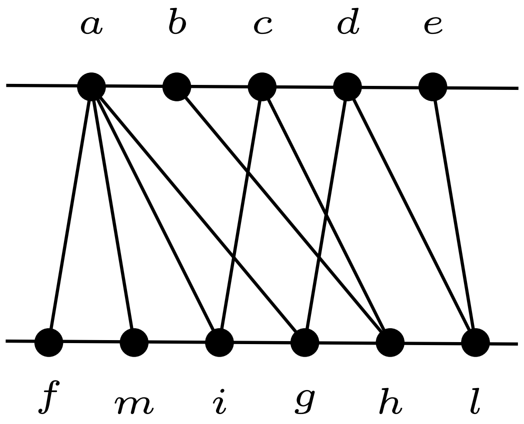
What is the One-Sided Crossing Minimization problem?

Input: a bipartite graph $G = (U, V, E)$ with a fixed linear ordering of U

Output: a linear ordering of V such that the number of crossings is minimum



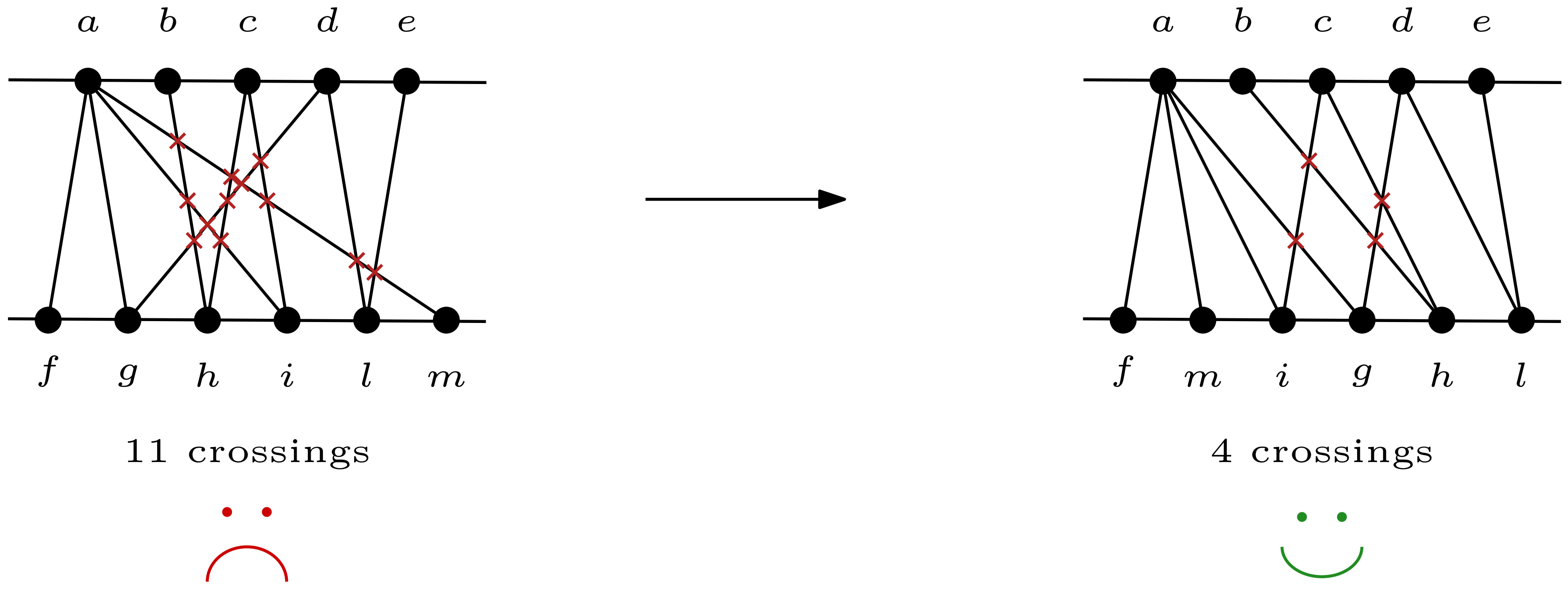
11 crossings



What is the One-Sided Crossing Minimization problem?

Input: a bipartite graph $G = (U, V, E)$ with a fixed linear ordering of U

Output: a linear ordering of V such that the number of crossings is minimum



State of the art

State of the art

- OSCM problem is $\mathcal{NP}$-complete [Eades et al. 1994]
  - Even for sparse graphs [Muñoz et al. 2001]

State of the art

- OSCM problem is $\mathcal{NP}$-complete [Eades et al. 1994]
  - Even for sparse graphs [Muñoz et al. 2001]
- Its importance in GD was first put in evidence by Sugiyama [Sugiyama et al. 1981]

State of the art

- OSCM problem is $\mathcal{NP}$-complete [Eades et al. 1994]
  - Even for sparse graphs [Muñoz et al. 2001]
- Its importance in GD was first put in evidence by Sugiyama [Sugiyama et al. 1981]
- Exact solutions for OSCM has been searched with branch-and-cut techniques [Jünger et al. 1995, Mutzel et al. 1998, Valls et al. 1996]

State of the art

- OSCM problem is $\mathcal{NP}$-complete [Eades et al. 1994]
  - Even for sparse graphs [Muñoz et al. 2001]
- Its importance in GD was first put in evidence by Sugiyama [Sugiyama et al. 1981]
- Exact solutions for OSCM has been searched with branch-and-cut techniques [Jünger et al. 1995, Mutzel et al. 1998, Valls et al. 1996]
- A $\mathcal{O}^*(2^n)$ time and space exact classic algorithm can be obtained [Bodlaender et. al 2009]
- A $\mathcal{O}^*(4^n)$ time and polynomial space exact classic algorithm can be obtained [Bodlaender et. al 2009]

State of the art

- OSCM problem is $\mathcal{NP}$-complete [Eades et al. 1994]
    - Even for sparse graphs [Muñoz et al. 2001]
- Its importance in GD was first put in evidence by Sugiyama [Sugiyama et al. 1981]
- Exact solutions for OSCM has been searched with branch-and-cut techniques [Jünger et al. 1995, Mutzel et al. 1998, Valls et al. 1996]
- A $\mathcal{O}^*(2^n)$ time and space exact classic algorithm can be obtained [Bodlaender et. al 2009]
- A $\mathcal{O}^*(4^n)$ time and polynomial space exact classic algorithm can be obtained [Bodlaender et. al 2009]
- The parameterized version of the problem has been widely investigated [Dujmovic et al. 2004,2008, Fernau et al. 2010, Kenyon-Mathieu et al. 2007, Alon et al. 2009]

State of the art

- OSCM problem is $\mathcal{NP}$-complete [Eades et al. 1994]
  - Even for sparse graphs [Muñoz et al. 2001]
- Its importance in GD was first put in evidence by Sugiyama [Sugiyama et al. 1981]
- Exact solutions for OSCM has been searched with branch-and-cut techniques [Jünger et al. 1995, Mutzel et al. 1998, Valls et al. 1996]
- A $\mathcal{O}^*(2^n)$ time and space exact classic algorithm can be obtained [Bodlaender et. al 2009]
- A $\mathcal{O}^*(4^n)$ time and polynomial space exact classic algorithm can be obtained [Bodlaender et. al 2009]
- The parameterized version of the problem has been widely investigated [Dujmovic et al. 2004,2008, Fernau et al. 2010, Kenyon-Mathieu et al. 2007, Alon et al. 2009]
  - Currently the best FPT results is $\mathcal{O}(k2^{\sqrt{2k}})$ [Kobayashi et al. 2015]

Our Contribution

Our Contribution

- We present singly-exponential quantum algorithms for the One-Sided Crossing Minimization (OSCM) problem

Our Contribution

- We present singly-exponential quantum algorithms for the One-Sided Crossing Minimization (OSCM) problem

Classic	
time	space
$\mathcal{O}^*(2^n)$	$\mathcal{O}^*(2^n)$
$\mathcal{O}^*(4^n)$	$\mathcal{O}^*(poly(n))$

Our Contribution

- We present singly-exponential quantum algorithms for the One-Sided Crossing Minimization (OSCM) problem

Classic		Quantum	
time	space	time	space
$\mathcal{O}^*(2^n)$	$\mathcal{O}^*(2^n)$	$\mathcal{O}^*(1.728^n)$	$\mathcal{O}^*(1.728^n)$
$\mathcal{O}^*(4^n)$	$\mathcal{O}^*(poly(n))$		

Our Contribution

- We present singly-exponential quantum algorithms for the One-Sided Crossing Minimization (OSCM) problem

Classic		Quantum	
time	space	time	space
$\mathcal{O}^*(2^n)$	$\mathcal{O}^*(2^n)$	$\mathcal{O}^*(1.728^n)$	$\mathcal{O}^*(1.728^n)$
$\mathcal{O}^*(4^n)$	$\mathcal{O}^*(poly(n))$		

Our Contribution

- We present singly-exponential quantum algorithms for the One-Sided Crossing Minimization (OSCM) problem

Classic		Quantum	
time	space	time	space
$\mathcal{O}^*(2^n)$	$\mathcal{O}^*(2^n)$	$\mathcal{O}^*(1.728^n)$	$\mathcal{O}^*(1.728^n)$
$\mathcal{O}^*(4^n)$	$\mathcal{O}^*(poly(n))$		

Our Contribution

- We present singly-exponential quantum algorithms for the One-Sided Crossing Minimization (OSCM) problem

Classic		Quantum	
time	space	time	space
$\mathcal{O}^*(2^n)$	$\mathcal{O}^*(2^n)$	$\mathcal{O}^*(1.728^n)$	$\mathcal{O}^*(1.728^n)$
$\mathcal{O}^*(4^n)$	$\mathcal{O}^*(poly(n))$	$\mathcal{O}^*(2^n)$	$\mathcal{O}^*(poly(n))$

Our Contribution

- We present singly-exponential quantum algorithms for the One-Sided Crossing Minimization (OSCM) problem

Classic		Quantum	
time	space	time	space
$\mathcal{O}^*(2^n)$	$\mathcal{O}^*(2^n)$	$\mathcal{O}^*(1.728^n)$	$\mathcal{O}^*(1.728^n)$
$\mathcal{O}^*(4^n)$	$\mathcal{O}^*(poly(n))$	$\mathcal{O}^*(2^n)$	$\mathcal{O}^*(poly(n))$

Quantum Preliminaries

Quantum Preliminaries

- Qubit

Quantum Preliminaries

- Qubit
  - $|\gamma\rangle$

Quantum Preliminaries

- Qubit
  - $|\gamma\rangle = \alpha_1 |0\rangle + \alpha_2 |1\rangle,\ \alpha_1, \alpha_2 \in \mathcal{C}$

Quantum Preliminaries

- Qubit
  - $|\gamma\rangle = \alpha_1 |0\rangle + \alpha_2 |1\rangle$, $\alpha_1, \alpha_2 \in \mathcal{C}$
- Superposition

Quantum Preliminaries

- Qubit
  - $|\gamma\rangle = \alpha_1 |0\rangle + \alpha_2 |1\rangle$, $\alpha_1, \alpha_2 \in \mathcal{C}$
- Superposition
  - $|\phi\rangle$

Quantum Preliminaries

- Qubit
  - $|\gamma\rangle = \alpha_1 |0\rangle + \alpha_2 |1\rangle,\ \alpha_1, \alpha_2 \in \mathcal{C}$
- Superposition
  - $|\phi\rangle = \alpha_1 |00\rangle + \alpha_2 |01\rangle + \alpha_3 |10\rangle + \alpha_4 |11\rangle$

Quantum Preliminaries

- Qubit
  - $|\gamma\rangle = \alpha_1 |0\rangle + \alpha_2 |1\rangle$, $\alpha_1, \alpha_2 \in \mathcal{C}$
- Superposition
  - $|\phi\rangle = \alpha_1 |00\rangle + \alpha_2 |01\rangle + \alpha_3 |10\rangle + \alpha_4 |11\rangle = \sum_{x \in \{0,1\}^2} \alpha_x |x\rangle$

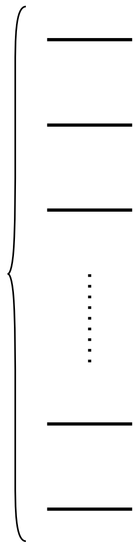
Quantum Preliminaries

- Qubit
  - $|\gamma\rangle = \alpha_1 |0\rangle + \alpha_2 |1\rangle$, $\alpha_1, \alpha_2 \in \mathcal{C}$
- Superposition
  - $|\phi\rangle = \alpha_1 |00\rangle + \alpha_2 |01\rangle + \alpha_3 |10\rangle + \alpha_4 |11\rangle = \sum_{x \in \{0,1\}^2} \alpha_x |x\rangle$
- Quantum computation

Quantum Preliminaries

- Qubit
  - $|\gamma\rangle = \alpha_1 |0\rangle + \alpha_2 |1\rangle,\ \alpha_1, \alpha_2 \in \mathcal{C}$
- Superposition
  - $|\phi\rangle = \alpha_1 |00\rangle + \alpha_2 |01\rangle + \alpha_3 |10\rangle + \alpha_4 |11\rangle = \sum_{x\in\{0,1\}^2} \alpha_x |x\rangle$
- Quantum computation

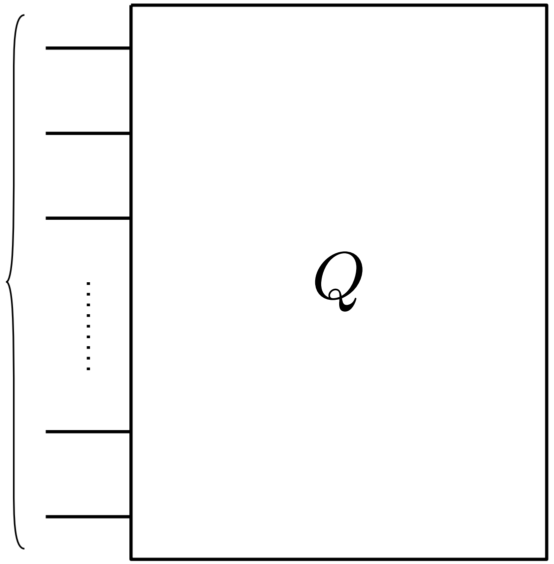
$$\sum_{x\in\{0,1\}^n} \alpha_x |x\rangle$$



Quantum Preliminaries

- Qubit
  - $|\gamma\rangle = \alpha_1 |0\rangle + \alpha_2 |1\rangle,\ \alpha_1, \alpha_2 \in \mathcal{C}$
- Superposition
  - $|\phi\rangle = \alpha_1 |00\rangle + \alpha_2 |01\rangle + \alpha_3 |10\rangle + \alpha_4 |11\rangle = \sum_{x \in \{0,1\}^2} \alpha_x |x\rangle$
- Quantum computation

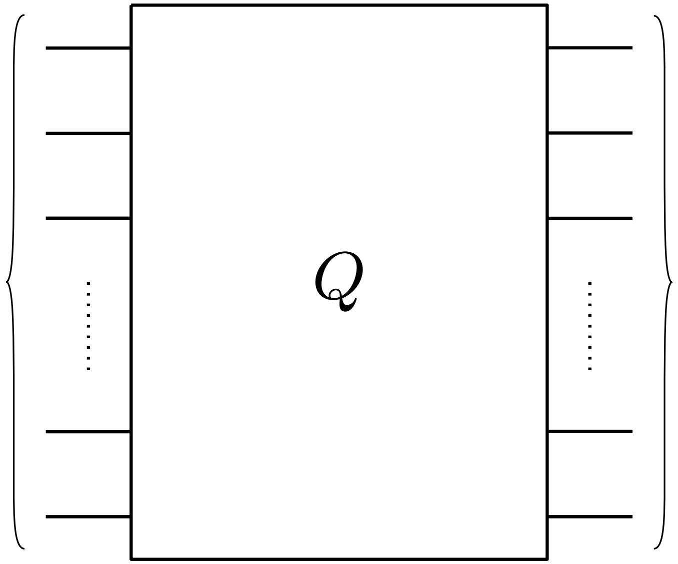
$$\sum_{x \in \{0,1\}^n} \alpha_x |x\rangle$$



Quantum Preliminaries

- Qubit
  - $|\gamma\rangle = \alpha_1 |0\rangle + \alpha_2 |1\rangle,\ \alpha_1, \alpha_2 \in \mathcal{C}$
- Superposition
  - $|\phi\rangle = \alpha_1 |00\rangle + \alpha_2 |01\rangle + \alpha_3 |10\rangle + \alpha_4 |11\rangle = \sum_{x \in \{0,1\}^2} \alpha_x |x\rangle$
- Quantum computation

$\sum_{x \in \{0,1\}^n} \alpha_x |x\rangle$



$\sum_{x \in \{0,1\}^n} \beta_x |x\rangle$

Quantum tools

Quantum tools

- Quantum Random Access Memory (QRAM)

Quantum tools

- Quantum Random Access Memory (QRAM)
  - This model of computation enables us to use quantum search primitives that involve condition checking on data stored in a random access memory

Quantum tools

- Quantum Random Access Memory (QRAM)
  - This model of computation enables us to use quantum search primitives that involve condition checking on data stored in a random access memory

$\lvert x\rangle$ (address)	$\lvert d_x\rangle$ (data)
00	110
01	101
10	001
11	111

Quantum tools

- Quantum Random Access Memory (QRAM)
  - This model of computation enables us to use quantum search primitives that involve condition checking on data stored in a random access memory

- $\alpha_0 |00\rangle |000\rangle + \alpha_1 |01\rangle |000\rangle + \alpha_2 10 |000\rangle + \alpha_3 |11\rangle |000\rangle$

$\vert x\rangle$ (address)	$\vert d_x\rangle$ (data)
00	110
01	101
10	001
11	111

Quantum tools

- Quantum Random Access Memory (QRAM)
  - This model of computation enables us to use quantum search primitives that involve condition checking on data stored in a random access memory

- $\alpha_0 |00\rangle |000\rangle + \alpha_1 |01\rangle |000\rangle + \alpha_2 10 |000\rangle + \alpha_3 |11\rangle |000\rangle$

$\|x\rangle$ (address)	$\|d_x\rangle$ (data)
00	110
01	101
10	001
11	111

Quantum tools

- Quantum Random Access Memory (QRAM)
  - This model of computation enables us to use quantum search primitives that involve condition checking on data stored in a random access memory

- $\alpha_0 |00\rangle |000\rangle + \alpha_1 |01\rangle |000\rangle + \alpha_2 10 |000\rangle + \alpha_3 |11\rangle |000\rangle$

$\|x\rangle$ (address)	$\|d_x\rangle$ (data)
00	110
01	101
10	001
11	111

Quantum tools

- Quantum Random Access Memory (QRAM)
  - This model of computation enables us to use quantum search primitives that involve condition checking on data stored in a random access memory

- $\alpha_0 |00\rangle |000\rangle + \alpha_1 |01\rangle |000\rangle + \alpha_2 10 |000\rangle + \alpha_3 |11\rangle |000\rangle$

$\|x\rangle$ (address)	$\|d_x\rangle$ (data)
00	110
01	101
10	001
11	111

$\alpha_0 |00\rangle |000\rangle$ ——

$\alpha_1 |01\rangle |000\rangle$ ——

$\alpha_2 |10\rangle |000\rangle$ ——

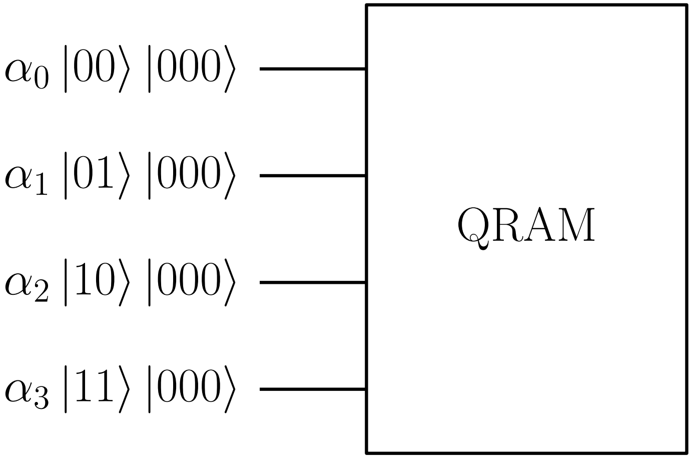
$\alpha_3 |11\rangle |000\rangle$ ——

Quantum tools

- Quantum Random Access Memory (QRAM)
  - This model of computation enables us to use quantum search primitives that involve condition checking on data stored in a random access memory

- $\alpha_0 |00\rangle |000\rangle + \alpha_1 |01\rangle |000\rangle + \alpha_2 10 |000\rangle + \alpha_3 |11\rangle |000\rangle$

$\lvert x\rangle$ (address)	$\lvert d_x\rangle$ (data)
00	110
01	101
10	001
11	111

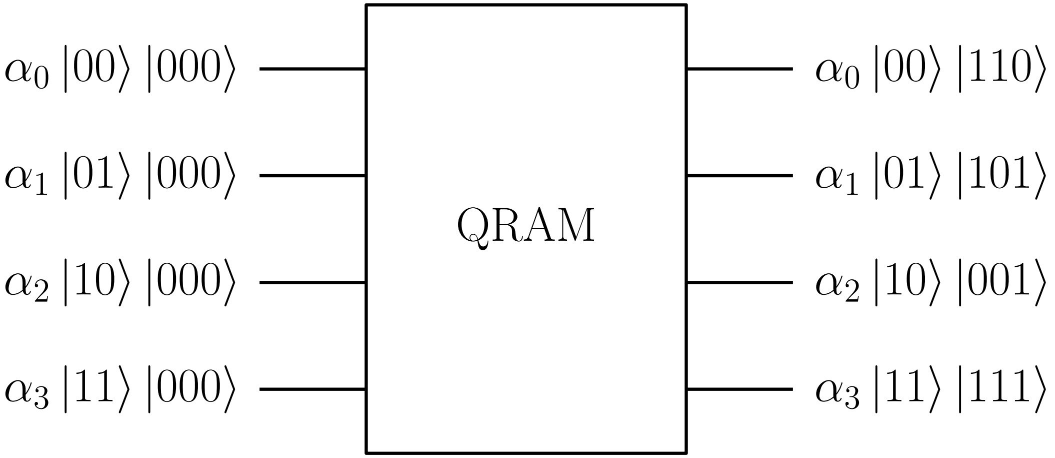


Quantum tools

- Quantum Random Access Memory (QRAM)
  - This model of computation enables us to use quantum search primitives that involve condition checking on data stored in a random access memory

- $\alpha_0 |00\rangle |000\rangle + \alpha_1 |01\rangle |000\rangle + \alpha_2 10 |000\rangle + \alpha_3 |11\rangle |000\rangle$

$\lvert x\rangle$ (address)	$\lvert d_x\rangle$ (data)
00	110
01	101
10	001
11	111

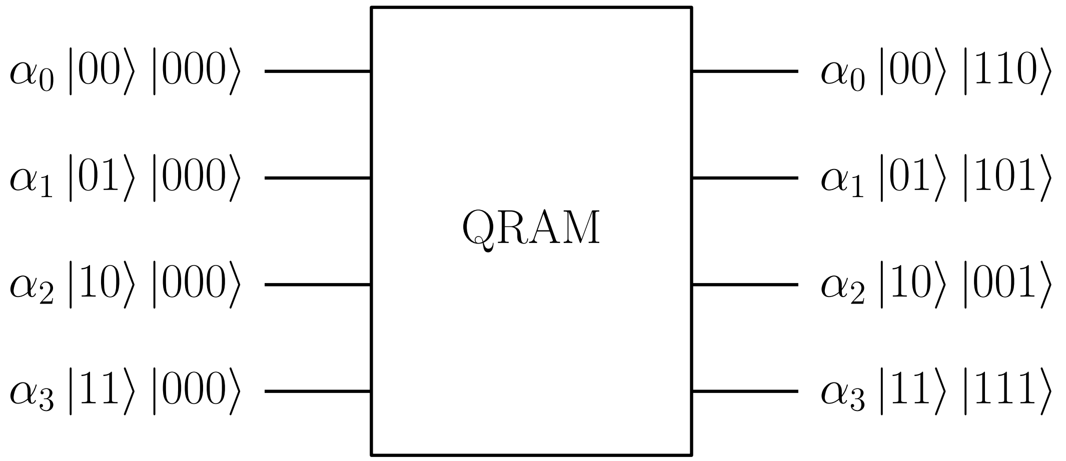


Quantum tools

- Quantum Random Access Memory (QRAM)
  - This model of computation enables us to use quantum search primitives that involve condition checking on data stored in a random access memory

- $\alpha_0 |00\rangle |000\rangle + \alpha_1 |01\rangle |000\rangle + \alpha_2 10 |000\rangle + \alpha_3 |11\rangle |000\rangle \rightarrow \alpha_0 |00\rangle |110\rangle + \alpha_1 |01\rangle |101\rangle + \alpha_2 10 |001\rangle + \alpha_3 |11\rangle |111\rangle$

$\lvert x\rangle$ (address)	$\lvert d_x\rangle$ (data)
00	110
01	101
10	001
11	111



Quantum tools

Quantum tools

- Quantum Minimum Finding (QMF) [Durr 1996]

Quantum tools

- Quantum Minimum Finding (QMF) [Durr 1996]
  - Given a table T of size N the algorithms finds the index y such that $T[y]$ is minimized in time $\mathcal{O}(\sqrt{N})$

Quantum Dynamic Programming for Set Problems

Quantum Dynamic Programming for Set Problems

- Set problem

Quantum Dynamic Programming for Set Problems

- Set problem
  - The solution for a set X can be determined by considering optimal solutions for all partitions $(S, X \setminus S)$ of X with $|S| = k$, for any fixed positive k, using polynomial time for each partition

Quantum Dynamic Programming for Set Problems

- Set problem
  - The solution for a set X can be determined by considering optimal solutions for all partitions $(S, X \setminus S)$ of X with $|S| = k$, for any fixed positive k, using polynomial time for each partition
- Ambainis et al. [SODA 2019] introduced a quantum framework designed to speedup several classic exponential-time and space algorithms

Quantum Dynamic Programming for Set Problems

- Set problem
  - The solution for a set X can be determined by considering optimal solutions for all partitions $(S, X \setminus S)$ of X with $|S| = k$, for any fixed positive k, using polynomial time for each partition
- Ambainis et al. [SODA 2019] introduced a quantum framework designed to speedup several classic exponential-time and space algorithms

Lemma *Let $\mathcal{P}$ be an optimization problem over a set X.*

Quantum Dynamic Programming for Set Problems

- Set problem
  - The solution for a set X can be determined by considering optimal solutions for all partitions $(S, X \setminus S)$ of X with $|S| = k$, for any fixed positive k, using polynomial time for each partition
- Ambainis et al. [SODA 2019] introduced a quantum framework designed to speedup several classic exponential-time and space algorithms

Lemma *Let $\mathcal{P}$ be an optimization problem over a set X. Let $|X| = n$ and let $OPT_{\mathcal{P}}(X)$ be the optimal value for $\mathcal{P}$ over X.*

Quantum Dynamic Programming for Set Problems

- Set problem
  - The solution for a set X can be determined by considering optimal solutions for all partitions $(S, X \setminus S)$ of X with $|S| = k$, for any fixed positive k, using polynomial time for each partition
- Ambainis et al. [SODA 2019] introduced a quantum framework designed to speedup several classic exponential-time and space algorithms

Lemma *Let $\mathcal{P}$ be an optimization problem over a set X. Let $|X| = n$ and let $OPT_{\mathcal{P}}(X)$ be the optimal value for $\mathcal{P}$ over X. Suppose that there exists a polynomial-time computable function $f_{\mathcal{P}} : 2^X \times 2^X \to \mathbb{R}$ such that, for any $S \subseteq X$, it holds that for any $k \in [|S| - 1]$:*

Quantum Dynamic Programming for Set Problems

- Set problem
  - The solution for a set X can be determined by considering optimal solutions for all partitions $(S, X \setminus S)$ of X with $|S| = k$, for any fixed positive k, using polynomial time for each partition
- Ambainis et al. [SODA 2019] introduced a quantum framework designed to speedup several classic exponential-time and space algorithms

Lemma *Let $\mathcal{P}$ be an optimization problem over a set X. Let $|X| = n$ and let $OPT_{\mathcal{P}}(X)$ be the optimal value for $\mathcal{P}$ over X. Suppose that there exists a polynomial-time computable function $f_{\mathcal{P}} : 2^X \times 2^X \to \mathbb{R}$ such that, for any $S \subseteq X$, it holds that for any $k \in [|S| - 1]$:*

$$OPT_{\mathcal{P}}(S) = \min_{W \subset S, |W| = k} \{OPT_{\mathcal{P}}(W) + OPT_{\mathcal{P}}(S \setminus W) + f_{\mathcal{P}}(W, S \setminus W)\}$$

Quantum Dynamic Programming for Set Problems

- Set problem
  - The solution for a set X can be determined by considering optimal solutions for all partitions $(S, X \setminus S)$ of X with $|S| = k$, for any fixed positive k, using polynomial time for each partition
- Ambainis et al. [SODA 2019] introduced a quantum framework designed to speedup several classic exponential-time and space algorithms

Lemma *Let $\mathcal{P}$ be an optimization problem over a set X. Let $|X| = n$ and let $OPT_{\mathcal{P}}(X)$ be the optimal value for $\mathcal{P}$ over X. Suppose that there exists a polynomial-time computable function $f_{\mathcal{P}} : 2^X \times 2^X \to \mathbb{R}$ such that, for any $S \subseteq X$, it holds that for any $k \in [|S| - 1]$:*

$$OPT_{\mathcal{P}}(S) = \min_{W \subset S, |W| = k} \{OPT_{\mathcal{P}}(W) + OPT_{\mathcal{P}}(S \setminus W) + f_{\mathcal{P}}(W, S \setminus W)\}$$

Quantum Dynamic Programming for Set Problems

- Set problem
  - The solution for a set X can be determined by considering optimal solutions for all partitions $(S, X \setminus S)$ of X with $|S| = k$, for any fixed positive k, using polynomial time for each partition
- Ambainis et al. [SODA 2019] introduced a quantum framework designed to speedup several classic exponential-time and space algorithms

Lemma *Let $\mathcal{P}$ be an optimization problem over a set X. Let $|X| = n$ and let $OPT_{\mathcal{P}}(X)$ be the optimal value for $\mathcal{P}$ over X. Suppose that there exists a polynomial-time computable function $f_{\mathcal{P}} : 2^X \times 2^X \to \mathbb{R}$ such that, for any $S \subseteq X$, it holds that for any $k \in [|S| - 1]$:*

$$OPT_{\mathcal{P}}(S) = \min_{W \subset S, |W| = k} \{OPT_{\mathcal{P}}(W) + OPT_{\mathcal{P}}(S \setminus W) + f_{\mathcal{P}}(W, S \setminus W)\}$$

Quantum Dynamic Programming for Set Problems

- Set problem
  - The solution for a set X can be determined by considering optimal solutions for all partitions $(S, X \setminus S)$ of X with $|S| = k$, for any fixed positive k, using polynomial time for each partition
- Ambainis et al. [SODA 2019] introduced a quantum framework designed to speedup several classic exponential-time and space algorithms

Lemma *Let $\mathcal{P}$ be an optimization problem over a set X. Let $|X| = n$ and let $OPT_{\mathcal{P}}(X)$ be the optimal value for $\mathcal{P}$ over X. Suppose that there exists a polynomial-time computable function $f_{\mathcal{P}} : 2^X \times 2^X \to \mathbb{R}$ such that, for any $S \subseteq X$, it holds that for any $k \in [|S| - 1]$:*

$$OPT_{\mathcal{P}}(S) = \min_{W \subset S, |W| = k} \{OPT_{\mathcal{P}}(W) + OPT_{\mathcal{P}}(S \setminus W) + f_{\mathcal{P}}(W, S \setminus W)\}$$

Quantum Dynamic Programming for Set Problems

- Set problem
  - The solution for a set X can be determined by considering optimal solutions for all partitions $(S, X \setminus S)$ of X with $|S| = k$, for any fixed positive k, using polynomial time for each partition
- Ambainis et al. [SODA 2019] introduced a quantum framework designed to speedup several classic exponential-time and space algorithms

Lemma *Let $\mathcal{P}$ be an optimization problem over a set X. Let $|X| = n$ and let $OPT_{\mathcal{P}}(X)$ be the optimal value for $\mathcal{P}$ over X. Suppose that there exists a polynomial-time computable function $f_{\mathcal{P}} : 2^X \times 2^X \to \mathbb{R}$ such that, for any $S \subseteq X$, it holds that for any $k \in [|S| - 1]$:*

$$OPT_{\mathcal{P}}(S) = \min_{W \subset S, |W| = k} \{OPT_{\mathcal{P}}(W) + OPT_{\mathcal{P}}(S \setminus W) + f_{\mathcal{P}}(W, S \setminus W)\}$$

Quantum Dynamic Programming for Set Problems

- Set problem
  - The solution for a set X can be determined by considering optimal solutions for all partitions $(S, X \setminus S)$ of X with $|S| = k$, for any fixed positive k, using polynomial time for each partition
- Ambainis et al. [SODA 2019] introduced a quantum framework designed to speedup several classic exponential-time and space algorithms

Lemma *Let $\mathcal{P}$ be an optimization problem over a set X. Let $|X| = n$ and let $OPT_{\mathcal{P}}(X)$ be the optimal value for $\mathcal{P}$ over X. Suppose that there exists a polynomial-time computable function $f_{\mathcal{P}} : 2^X \times 2^X \to \mathbb{R}$ such that, for any $S \subseteq X$, it holds that for any $k \in [|S| - 1]$:*

$$OPT_{\mathcal{P}}(S) = \min_{W \subset S, |W| = k} \{OPT_{\mathcal{P}}(W) + OPT_{\mathcal{P}}(S \setminus W) + f_{\mathcal{P}}(W, S \setminus W)\}$$

Then, $OPT_{\mathcal{P}}(X)$ can be computed by a quantum algorithm that uses QRAM in $\mathcal{O}^*(1.728^n)$ time and space.

Quantum Dynamic Programming for Set Problems

Quantum Dynamic Programming for Set Problems

- The main idea is to

Quantum Dynamic Programming for Set Problems

- The main idea is to
  - Precompute solutions (pre-processing) for smaller subsets using classic dynamic programming and save the results in the QRAM

Quantum Dynamic Programming for Set Problems

- The main idea is to
  - Precompute solutions (pre-processing) for smaller subsets using classic dynamic programming and save the results in the QRAM
  - Recombine the results of the precomputation step to obtain the optimal solution for the whole set (recursively) applying Quantum Minimum Finding (QMF)

Quantum Dynamic Programming for Set Problems

Quantum Dynamic Programming for Set Problems

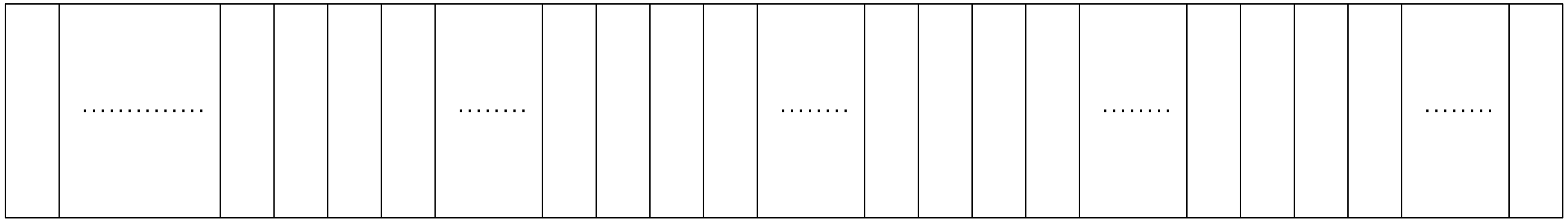
Quantum

Classic

Quantum Dynamic Programming for Set Problems

Quantum

Classic



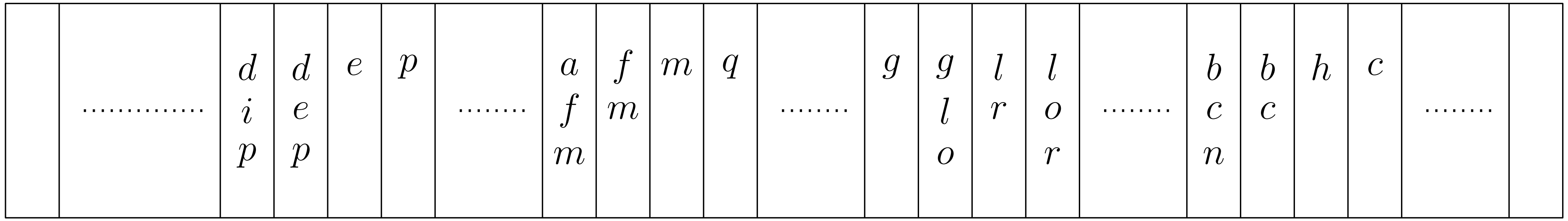
QRAM

pre-processing
classic precomputed
optimal solutions
are stored in QRAM

Quantum Dynamic Programming for Set Problems

Quantum

Classic



pre-processing classic precomputed optimal solutions are stored in QRAM

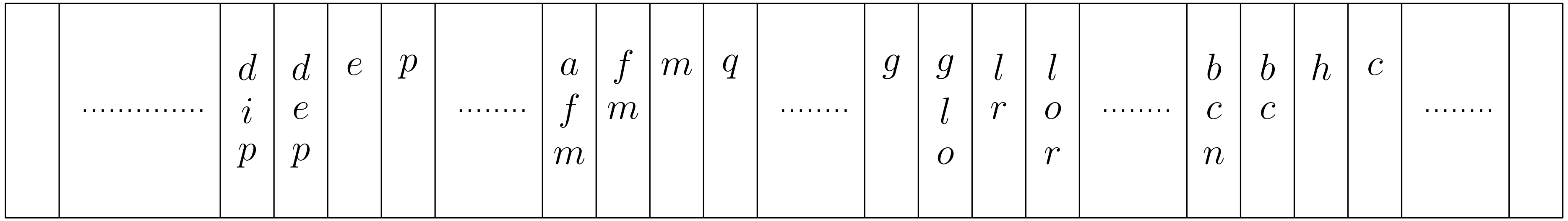
QRAM

Quantum Dynamic Programming for Set Problems

$a \quad b \quad c \quad d \quad e \quad f \quad g \quad h \quad i \quad l \quad m \quad n \quad o \quad p \quad q \quad r$

Quantum

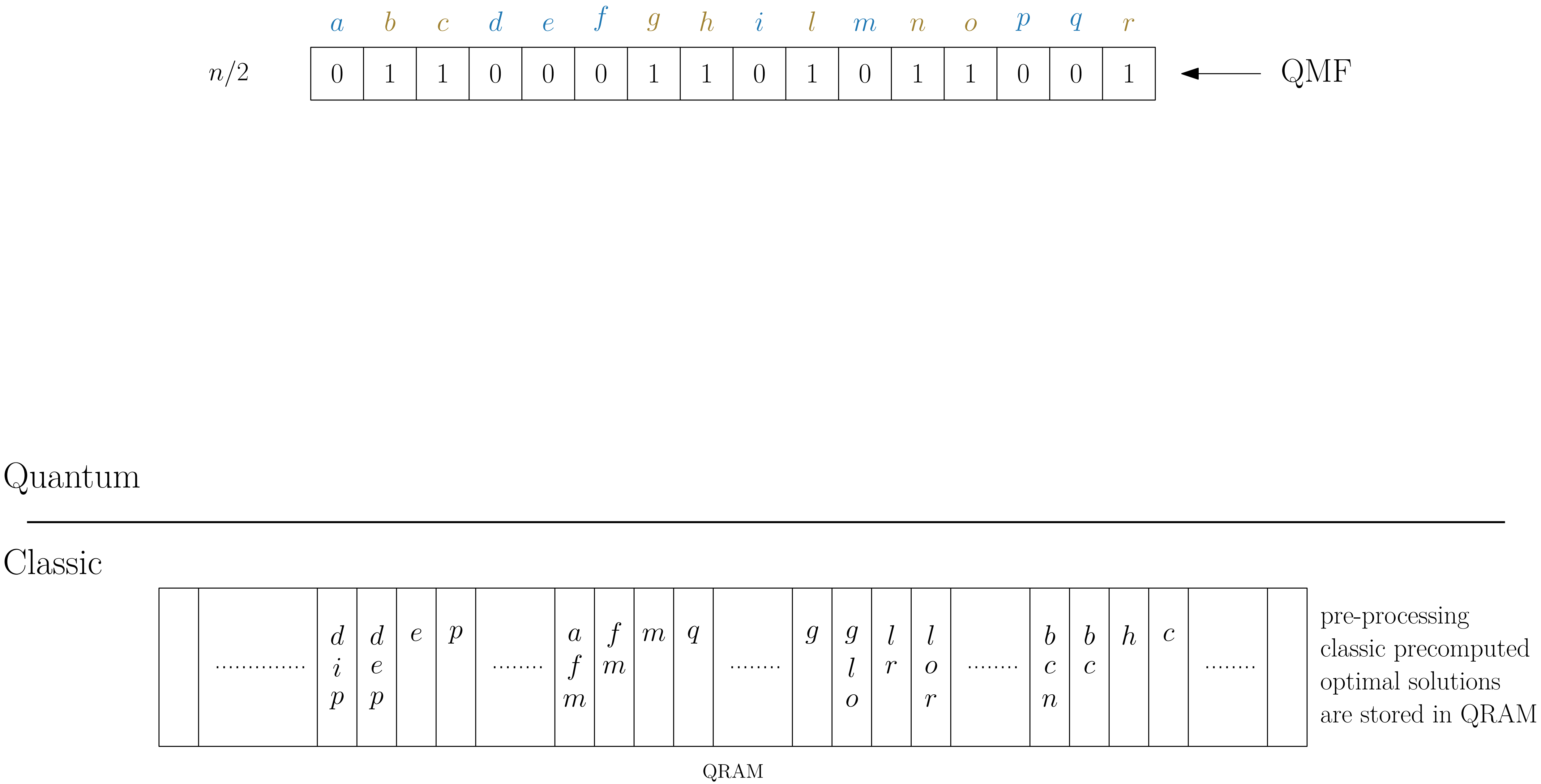
Classic



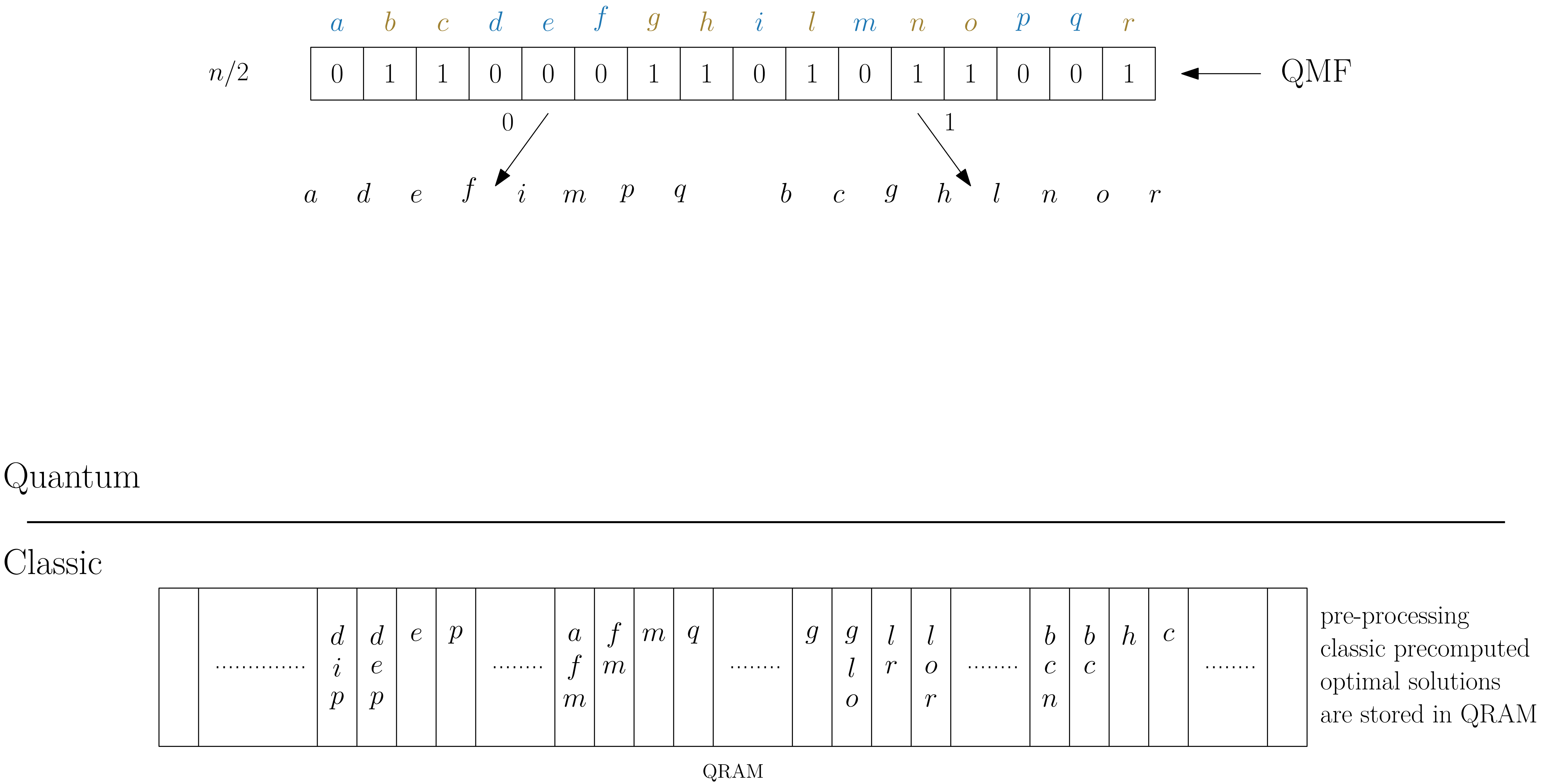
pre-processing classic precomputed optimal solutions are stored in QRAM

QRAM

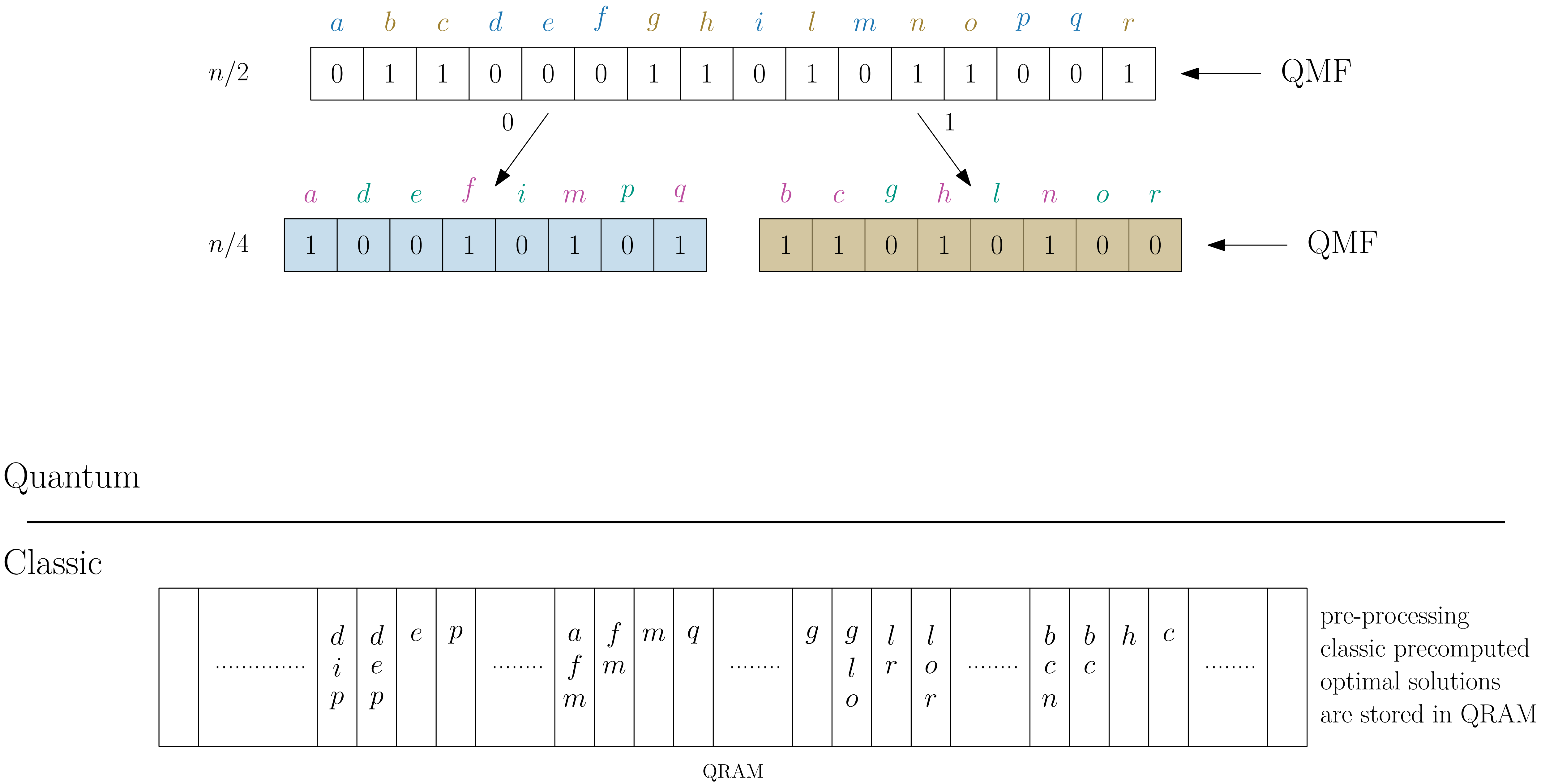
Quantum Dynamic Programming for Set Problems



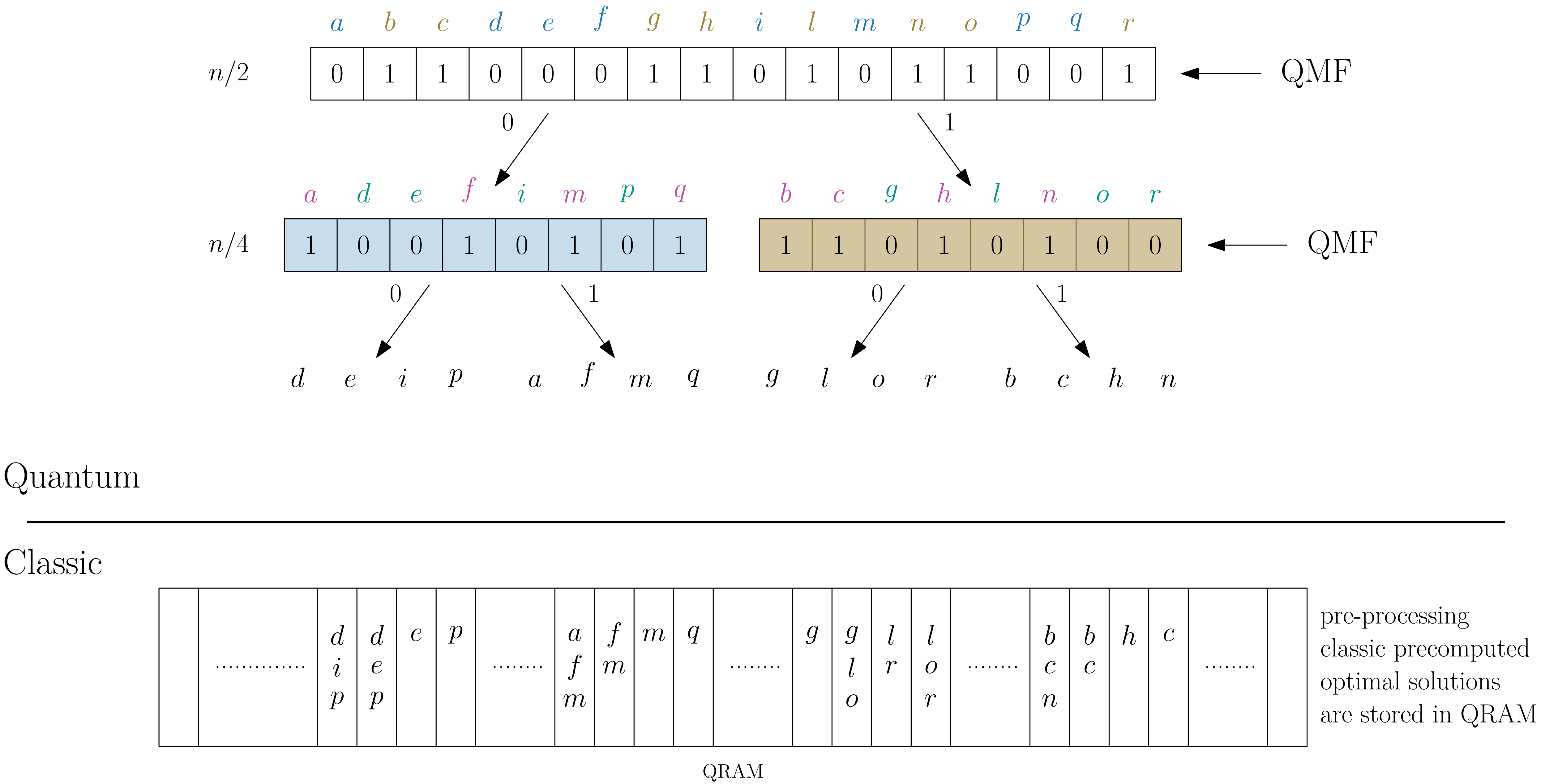
Quantum Dynamic Programming for Set Problems



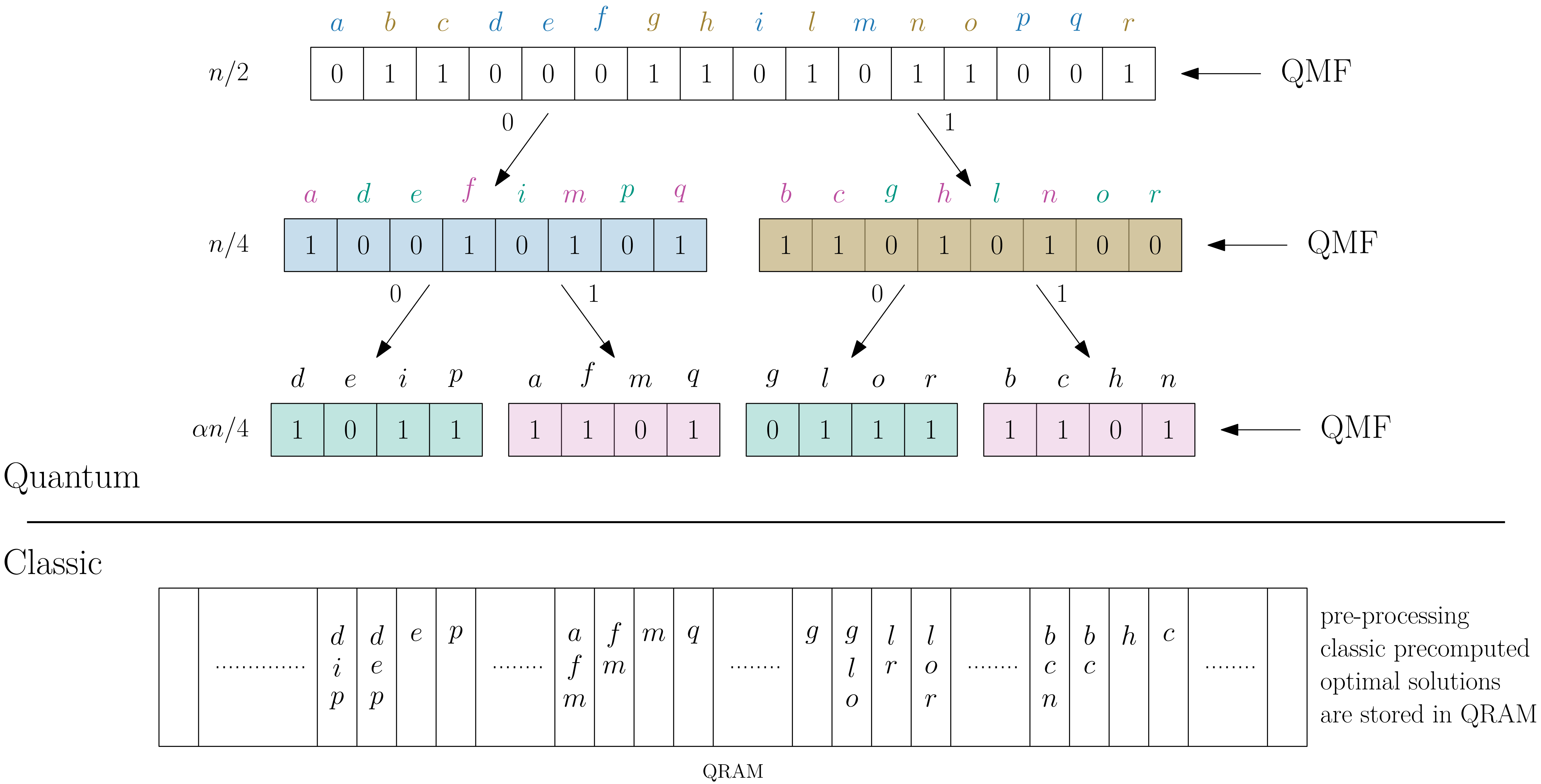
Quantum Dynamic Programming for Set Problems



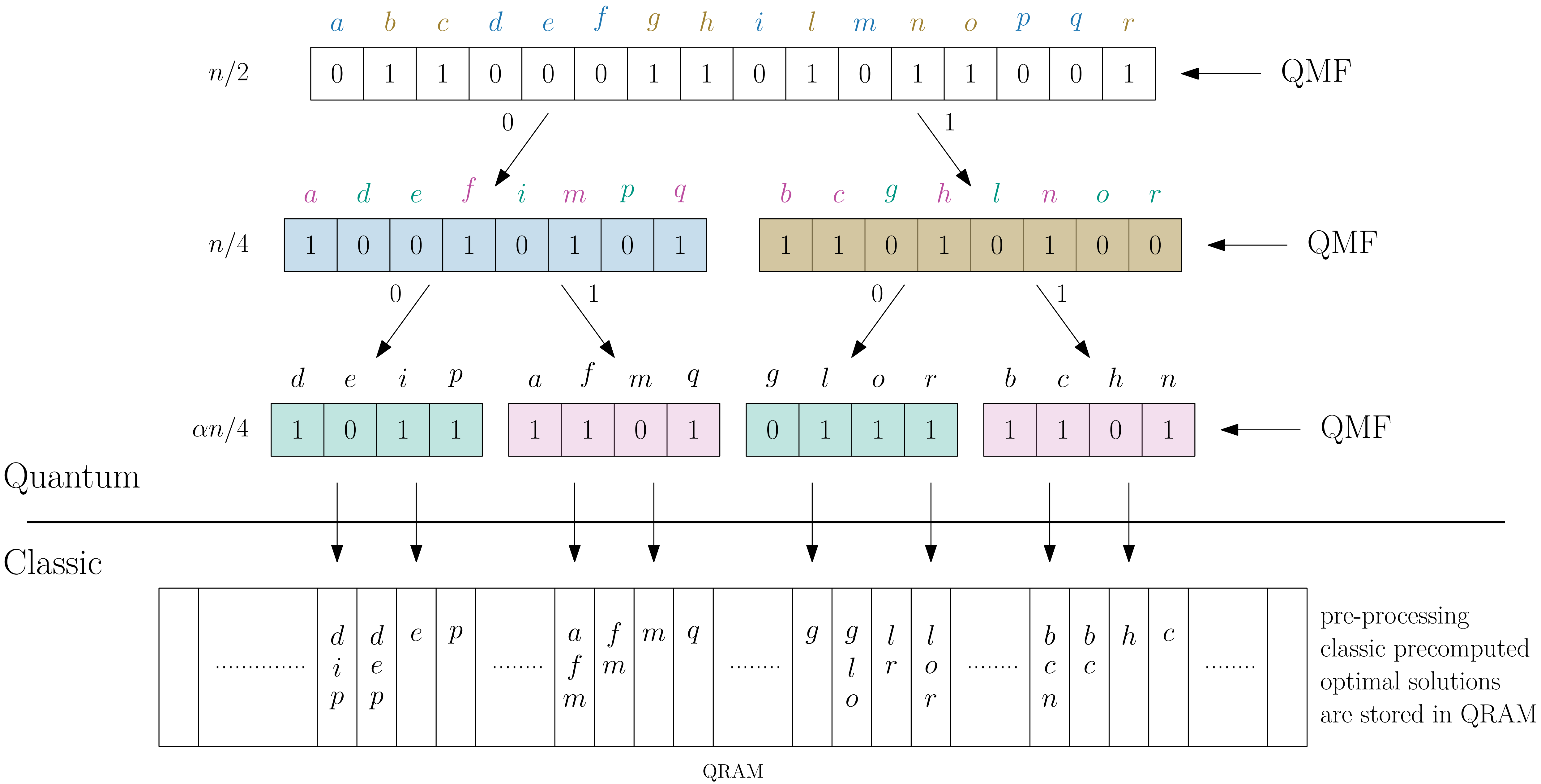
Quantum Dynamic Programming for Set Problems



Quantum Dynamic Programming for Set Problems



Quantum Dynamic Programming for Set Problems



Complexity Analysis

Complexity Analysis

- Time complexity of the classic pre-processing

Complexity Analysis

- Time complexity of the classic pre-processing
  - $\mathcal{O}^*\left(\binom{n}{\leq (1-\alpha)\frac{n}{4}}\right) = \mathcal{O}^*(1.728^n)$

Complexity Analysis

- Time complexity of the classic pre-processing
  - $\mathcal{O}^*\left(\binom{n}{\leq (1-\alpha)\frac{n}{4}}\right) = \mathcal{O}^*(1.728^n)$

Number of solutions calculated and stored during the pre-processing

Complexity Analysis

- Time complexity of the classic pre-processing
  - $\mathcal{O}^*\left(\binom{n}{\leq (1-\alpha)\frac{n}{4}}\right) = \mathcal{O}^*(1.728^n)$
- Time complexity of the quantum part

Complexity Analysis

- Time complexity of the classic pre-processing
  - $\mathcal{O}^*\left(\binom{n}{\leq (1-\alpha)\frac{n}{4}}\right) = \mathcal{O}^*(1.728^n)$
- Time complexity of the quantum part
  - $\mathcal{O}^*\left(\sqrt{\binom{n}{\frac{n}{2}}\binom{\frac{n}{2}}{\frac{n}{4}}\binom{n}{\frac{\alpha n}{4}}}\right) = \mathcal{O}^*\left(1.728^n\right)$

Complexity Analysis

- Time complexity of the classic pre-processing
  - $\mathcal{O}^*\left(\binom{n}{\leq(1-\alpha)\frac{n}{4}}\right) = \mathcal{O}^*(1.728^n)$
- Time complexity of the quantum part
  - $\mathcal{O}^*\left(\sqrt{\binom{n}{\frac{n}{2}}\binom{\frac{n}{2}}{\frac{n}{4}}\binom{n}{\frac{\alpha n}{4}}}\right) = \mathcal{O}^*\left(1.728^n\right)$

QMF over all subsets of size $n/2$

Complexity Analysis

- Time complexity of the classic pre-processing
  - $\mathcal{O}^*\left(\binom{n}{\leq(1-\alpha)\frac{n}{4}}\right) = \mathcal{O}^*(1.728^n)$
- Time complexity of the quantum part
  - $\mathcal{O}^*\left(\sqrt{\binom{n}{\frac{n}{2}}\binom{\frac{n}{2}}{\frac{n}{4}}\binom{n}{\frac{\alpha n}{4}}}\right) = \mathcal{O}^*\left(1.728^n\right)$

↓ QMF over all subsets of size $n/4$

Complexity Analysis

- Time complexity of the classic pre-processing
  - $\mathcal{O}^*\left(\binom{n}{\leq (1-\alpha)\frac{n}{4}}\right) = \mathcal{O}^*(1.728^n)$
- Time complexity of the quantum part
  - $\mathcal{O}^*\left(\sqrt{\binom{n}{\frac{n}{2}}\binom{\frac{n}{2}}{\frac{n}{4}}\binom{n}{\frac{\alpha n}{4}}}\right) = \mathcal{O}^*\left(1.728^n\right)$

QMF over all subsets of size $\alpha n/4$

Complexity Analysis

- Time complexity of the classic pre-processing
  - $\mathcal{O}^*\left(\binom{n}{\leq(1-\alpha)\frac{n}{4}}\right) = \mathcal{O}^*(1.728^n)$
- Time complexity of the quantum part
  - $\mathcal{O}^*\left(\sqrt{\binom{n}{\frac{n}{2}}\binom{\frac{n}{2}}{\frac{n}{4}}\binom{n}{\frac{\alpha n}{4}}}\right) = \mathcal{O}^*\left(1.728^n\right)$
- α is selected to balance quantum and classic complexities

Complexity Analysis

- Time complexity of the classic pre-processing
  - $\mathcal{O}^*\left(\binom{n}{\leq (1-\alpha)\frac{n}{4}}\right) = \mathcal{O}^*(1.728^n)$
- Time complexity of the quantum part
  - $\mathcal{O}^*\left(\sqrt{\binom{n}{\frac{n}{2}}\binom{\frac{n}{2}}{\frac{n}{4}}\binom{n}{\frac{\alpha n}{4}}}\right) = \mathcal{O}^*\left(1.728^n\right)$
- α is selected to balance quantum and classic complexities
- The resulting space and time complexity is $\mathcal{O}^*(1.728^n)$

Complexity Analysis

- Time complexity of the classic pre-processing
  - $\mathcal{O}^*\left(\binom{n}{\leq(1-\alpha)\frac{n}{4}}\right) = \mathcal{O}^*(1.728^n)$
- Time complexity of the quantum part
  - $\mathcal{O}^*\left(\sqrt{\binom{n}{\frac{n}{2}}\binom{\frac{n}{2}}{\frac{n}{4}}\binom{n}{\frac{\alpha n}{4}}}\right) = \mathcal{O}^*\left(1.728^n\right)$
- α is selected to balance quantum and classic complexities
- The resulting space and time complexity is $\mathcal{O}^*(1.728^n)$
  - The time and space complexity of the best classic algorithm is $\mathcal{O}^*(2^n)$

Quantum Dynamic Programming for OSCM

Quantum Dynamic Programming for OSCM

- OSCM is a set problem

Quantum Dynamic Programming for OSCM

- OSCM is a set problem
  - The optimal solution must respect the recurrence

Quantum Dynamic Programming for OSCM

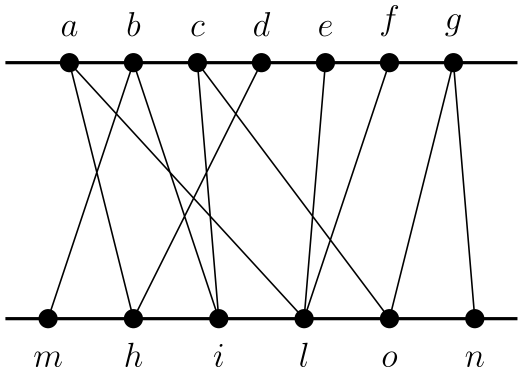
- OSCM is a set problem
  - The optimal solution must respect the recurrence

$$OPT_{\mathcal{P}}(S) = \min_{W \subset S, |W| = k} \{OPT_{\mathcal{P}}(W) + OPT_{\mathcal{P}}(S \setminus W) + f_{\mathcal{P}}(W, S \setminus W)\}$$

Quantum Dynamic Programming for OSCM

- OSCM is a set problem
  - The optimal solution must respect the recurrence

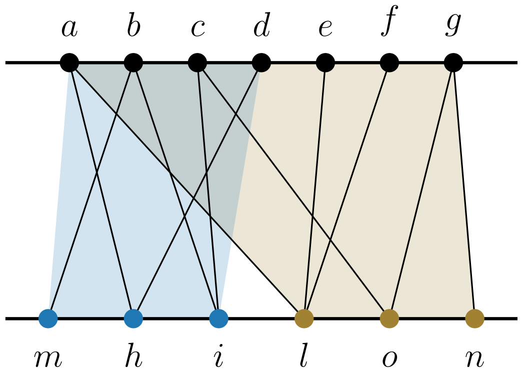
$$OPT_{\mathcal{P}}(S) = \min_{W \subset S, |W|=k} \{OPT_{\mathcal{P}}(W) + OPT_{\mathcal{P}}(S \setminus W) + f_{\mathcal{P}}(W, S \setminus W)\}$$



Quantum Dynamic Programming for OSCM

- OSCM is a set problem
  - The optimal solution must respect the recurrence

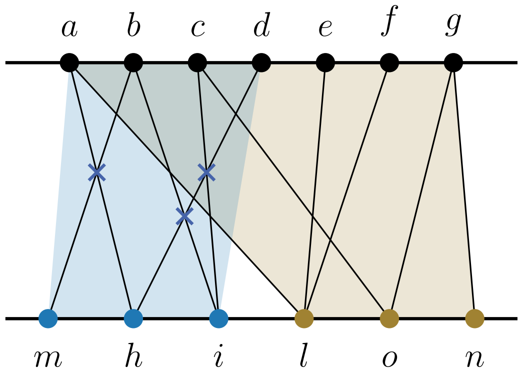
$$OPT_{\mathcal{P}}(S) = \min_{W \subset S, |W|=k} \{OPT_{\mathcal{P}}(W) + OPT_{\mathcal{P}}(S \setminus W) + f_{\mathcal{P}}(W, S \setminus W)\}$$



Quantum Dynamic Programming for OSCM

- OSCM is a set problem
  - The optimal solution must respect the recurrence

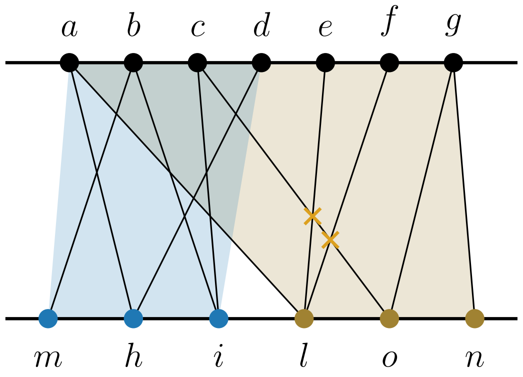
$$OPT_{\mathcal{P}}(S) = \min_{W \subset S, |W|=k} \{OPT_{\mathcal{P}}(W) + OPT_{\mathcal{P}}(S \setminus W) + f_{\mathcal{P}}(W, S \setminus W)\}$$



Quantum Dynamic Programming for OSCM

- OSCM is a set problem
  - The optimal solution must respect the recurrence

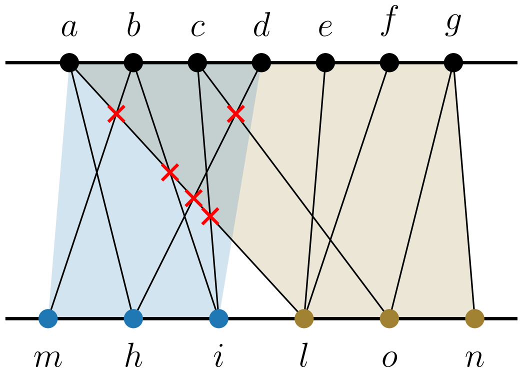
$$OPT_{\mathcal{P}}(S) = \min_{W \subset S, |W|=k} \{OPT_{\mathcal{P}}(W) + OPT_{\mathcal{P}}(S \setminus W) + f_{\mathcal{P}}(W, S \setminus W)\}$$



Quantum Dynamic Programming for OSCM

- OSCM is a set problem
  - The optimal solution must respect the recurrence

$$OPT_{\mathcal{P}}(S) = \min_{W \subset S, |W|=k} \{OPT_{\mathcal{P}}(W) + OPT_{\mathcal{P}}(S \setminus W) + f_{\mathcal{P}}(W, S \setminus W)\}$$



Quantum Dynamic Programming for OSCM

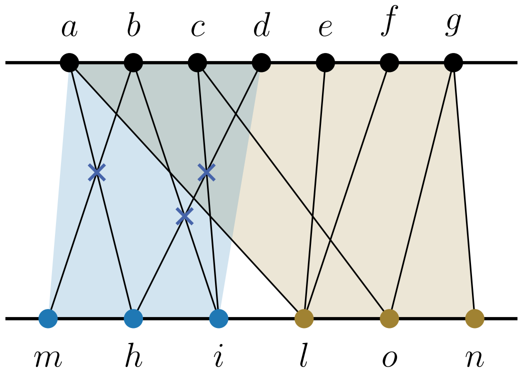
- OSCM is a set problem
  - The optimal solution must respect the recurrence

$$OPT_{\mathcal{P}}(S) = \min_{W \subset S, |W|=k} \{OPT_{\mathcal{P}}(W) + OPT_{\mathcal{P}}(S \setminus W) + f_{\mathcal{P}}(W, S \setminus W)\}$$

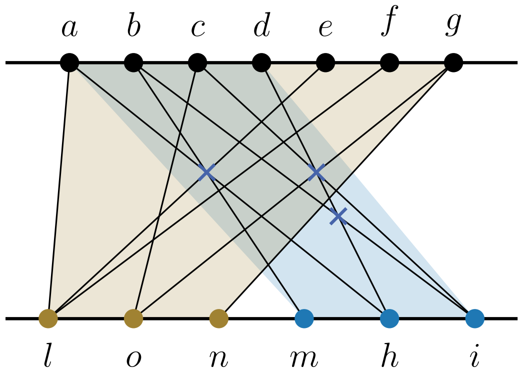
Quantum Dynamic Programming for OSCM

- OSCM is a set problem
  - The optimal solution must respect the recurrence

$$OPT_{\mathcal{P}}(S) = \min_{W \subset S, |W|=k} \{OPT_{\mathcal{P}}(W) + OPT_{\mathcal{P}}(S \setminus W) + f_{\mathcal{P}}(W, S \setminus W)\}$$



3 crossings

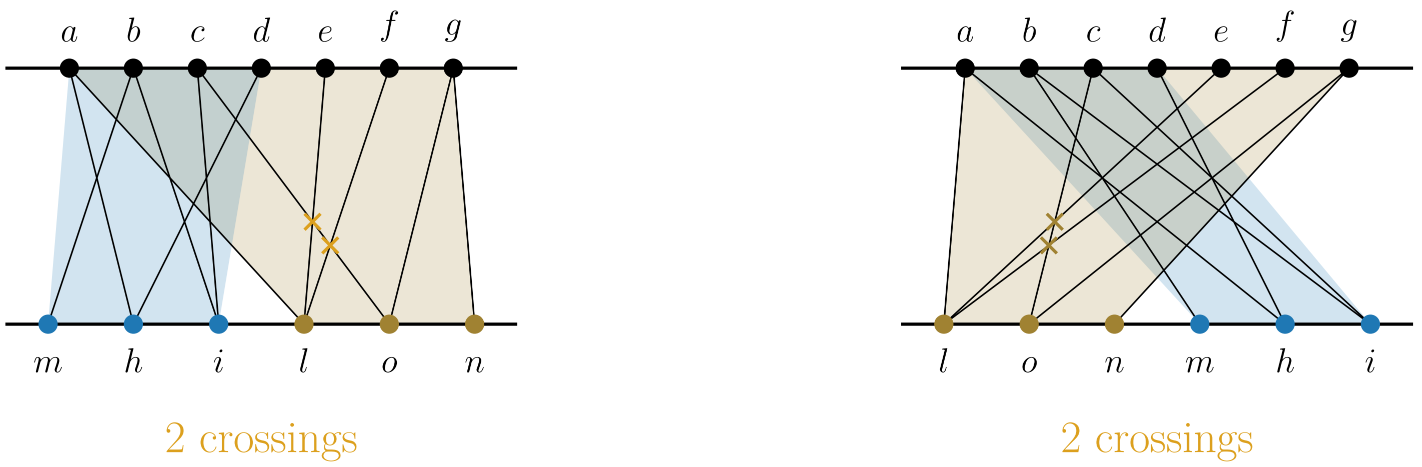


3 crossings

Quantum Dynamic Programming for OSCM

- OSCM is a set problem
  - The optimal solution must respect the recurrence

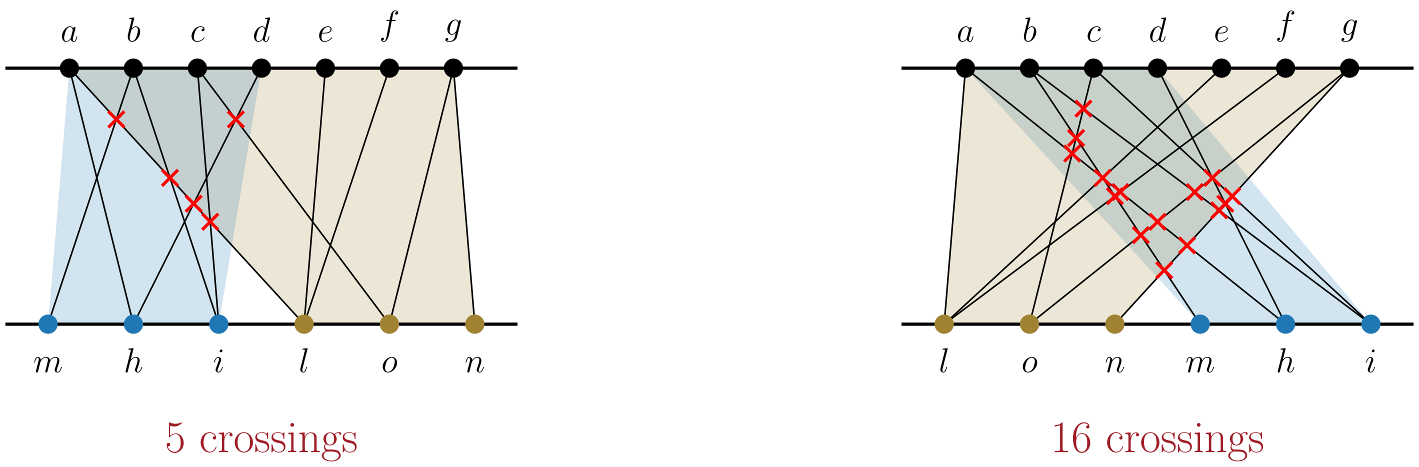
$$OPT_{\mathcal{P}}(S) = \min_{W \subset S, |W|=k} \{OPT_{\mathcal{P}}(W) + OPT_{\mathcal{P}}(S \setminus W) + f_{\mathcal{P}}(W, S \setminus W)\}$$



Quantum Dynamic Programming for OSCM

- OSCM is a set problem
  - The optimal solution must respect the recurrence

$$OPT_{\mathcal{P}}(S) = \min_{W \subset S, |W|=k} \{OPT_{\mathcal{P}}(W) + OPT_{\mathcal{P}}(S \setminus W) + f_{\mathcal{P}}(W, S \setminus W)\}$$



Quantum Dynamic Programming for OSCM

- OSCM is a set problem
  - The optimal solution must respect the recurrence

$$OPT_{\mathcal{P}}(S) = \min_{W \subset S, |W|=k} \{OPT_{\mathcal{P}}(W) + OPT_{\mathcal{P}}(S \setminus W) + f_{\mathcal{P}}(W, S \setminus W)\}$$

  - The number of crossings between the two partitions can be computed in polynomial time

Quantum Dynamic Programming for OSCM

- OSCM is a set problem
  - The optimal solution must respect the recurrence

$$OPT_{\mathcal{P}}(S) = \min_{W \subset S, |W|=k} \{OPT_{\mathcal{P}}(W) + OPT_{\mathcal{P}}(S \setminus W) + f_{\mathcal{P}}(W, S \setminus W)\}$$

  - The number of crossings between the two partitions can be computed in polynomial time
- There exists a quantum algorithm that solves OSCM using $\mathcal{O}^*(1.728^n)$ time and space

Quantum Divide and Conquer for Set Problems

Quantum Divide and Conquer for Set Problems

Lemma *Let $\mathcal{P}$ be an optimization problem over a set X. Let $|X| = n$ and let $OPT_{\mathcal{P}}(X)$ be the optimal value for $\mathcal{P}$ over X. Suppose that there exists a polynomial-time computable function $f_{\mathcal{P}} : 2^X \times 2^X \to \mathbb{R}$ and a constant $c_{\mathcal{P}}$ such that, for any $S \subseteq X$, it holds that:*

- *If $|S| \leq c_{\mathcal{P}}$, then $OPT_{\mathcal{P}}(S) = f_{\mathcal{P}}(S, \emptyset)$.*
- *If $|S| > c_{\mathcal{P}}$, then*

$$OPT_{\mathcal{P}}(S) = \min_{W \subset S, |W| = \frac{|S|}{2}} \{OPT_{\mathcal{P}}(W) + OPT_{\mathcal{P}}(S \setminus W) + f_{\mathcal{P}}(W, S \setminus W)\}$$

We have that, $OPT_{\mathcal{P}}(X)$ can be computed by a quantum algorithm without using QRAM in $\mathcal{O}^*(2^n)$ time and polynomial space.

Quantum Divide and Conquer for Set Problems

Lemma *Let $\mathcal{P}$ be an optimization problem over a set X. Let $|X| = n$ and let $OPT_{\mathcal{P}}(X)$ be the optimal value for $\mathcal{P}$ over X. Suppose that there exists a polynomial-time computable function $f_{\mathcal{P}} : 2^X \times 2^X \to \mathbb{R}$ and a constant $c_{\mathcal{P}}$ such that, for any $S \subseteq X$, it holds that:*

- *If $|S| \leq c_{\mathcal{P}}$, then $OPT_{\mathcal{P}}(S) = f_{\mathcal{P}}(S, \emptyset)$.*
- *If $|S| > c_{\mathcal{P}}$, then*

$$OPT_{\mathcal{P}}(S) = \min_{W \subset S, |W| = \frac{|S|}{2}} \{OPT_{\mathcal{P}}(W) + OPT_{\mathcal{P}}(S \setminus W) + f_{\mathcal{P}}(W, S \setminus W)\}$$

We have that, $OPT_{\mathcal{P}}(X)$ can be computed by a quantum algorithm without using QRAM in $\mathcal{O}^*(2^n)$ time and polynomial space.

- It does not interrupt the recursion

Quantum Divide and Conquer for Set Problems

Lemma *Let $\mathcal{P}$ be an optimization problem over a set X. Let $|X| = n$ and let $OPT_{\mathcal{P}}(X)$ be the optimal value for $\mathcal{P}$ over X. Suppose that there exists a polynomial-time computable function $f_{\mathcal{P}} : 2^X \times 2^X \to \mathbb{R}$ and a constant $c_{\mathcal{P}}$ such that, for any $S \subseteq X$, it holds that:*

- *If $|S| \leq c_{\mathcal{P}}$, then $OPT_{\mathcal{P}}(S) = f_{\mathcal{P}}(S, \emptyset)$.*
- *If $|S| > c_{\mathcal{P}}$, then*

$$OPT_{\mathcal{P}}(S) = \min_{W \subset S, |W| = \frac{|S|}{2}} \{OPT_{\mathcal{P}}(W) + OPT_{\mathcal{P}}(S \setminus W) + f_{\mathcal{P}}(W, S \setminus W)\}$$

We have that, $OPT_{\mathcal{P}}(X)$ can be computed by a quantum algorithm without using QRAM in $\mathcal{O}^*(2^n)$ time and polynomial space.

- It does not interrupt the recursion

Quantum Divide and Conquer for Set Problems

Lemma *Let $\mathcal{P}$ be an optimization problem over a set X. Let $|X| = n$ and let $OPT_{\mathcal{P}}(X)$ be the optimal value for $\mathcal{P}$ over X. Suppose that there exists a polynomial-time computable function $f_{\mathcal{P}} : 2^X \times 2^X \to \mathbb{R}$ and a constant $c_{\mathcal{P}}$ such that, for any $S \subseteq X$, it holds that:*

- *If $|S| \leq c_{\mathcal{P}}$, then $OPT_{\mathcal{P}}(S) = f_{\mathcal{P}}(S, \emptyset)$.*
- *If $|S| > c_{\mathcal{P}}$, then*

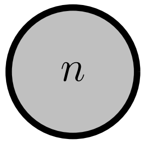
$$OPT_{\mathcal{P}}(S) = \min_{W \subset S, |W| = \frac{|S|}{2}} \{OPT_{\mathcal{P}}(W) + OPT_{\mathcal{P}}(S \setminus W) + f_{\mathcal{P}}(W, S \setminus W)\}$$

We have that, $OPT_{\mathcal{P}}(X)$ can be computed by a quantum algorithm without using QRAM in $\mathcal{O}^*(2^n)$ time and polynomial space.

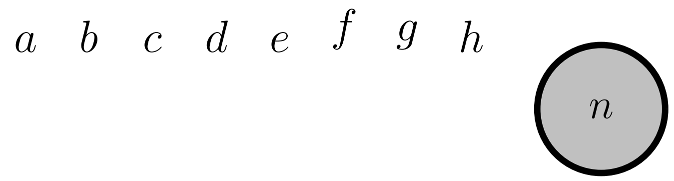
- It does not interrupt the recursion
- It does not use any QRAM

Quantum Divide and Conquer for Set Problems

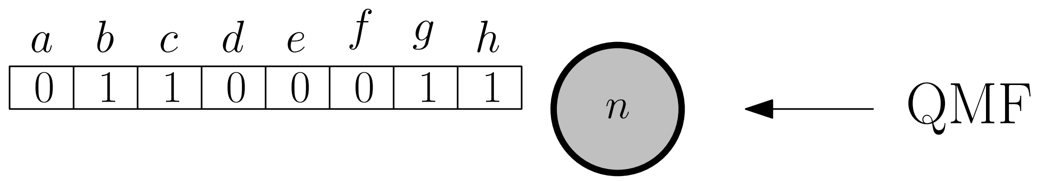
Quantum Divide and Conquer for Set Problems



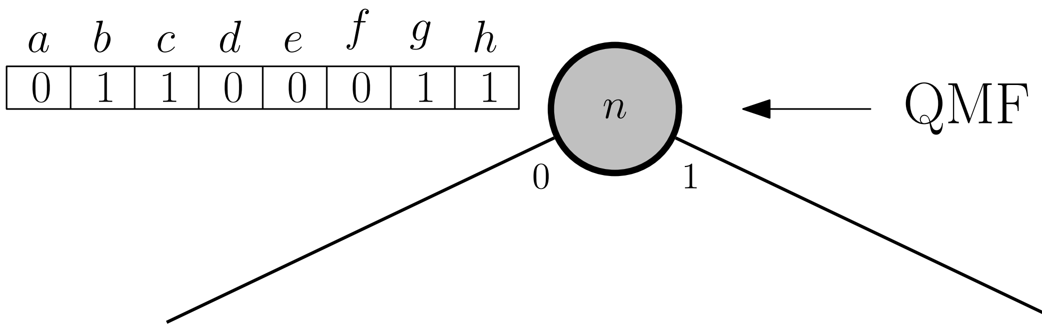
Quantum Divide and Conquer for Set Problems



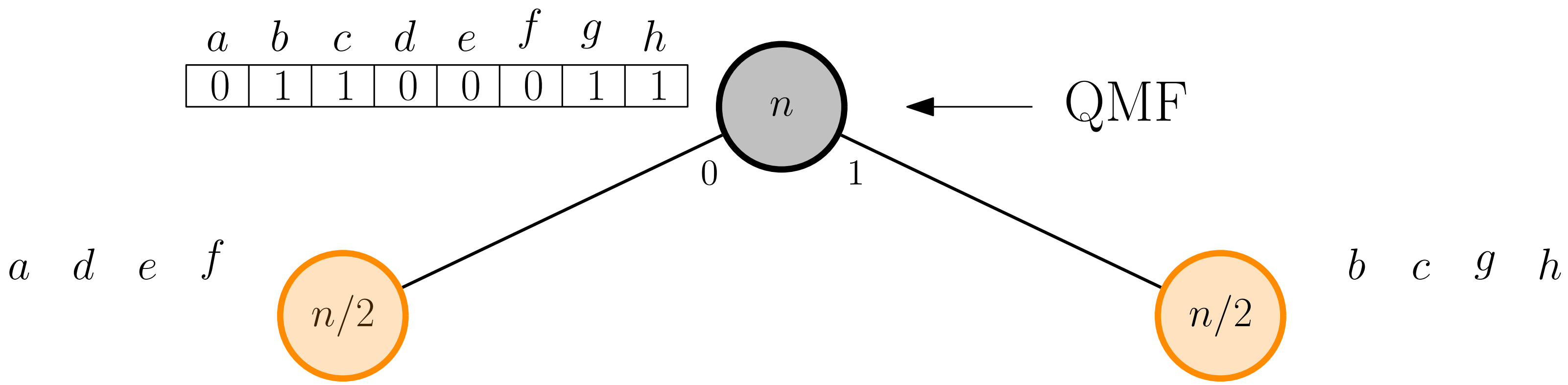
Quantum Divide and Conquer for Set Problems



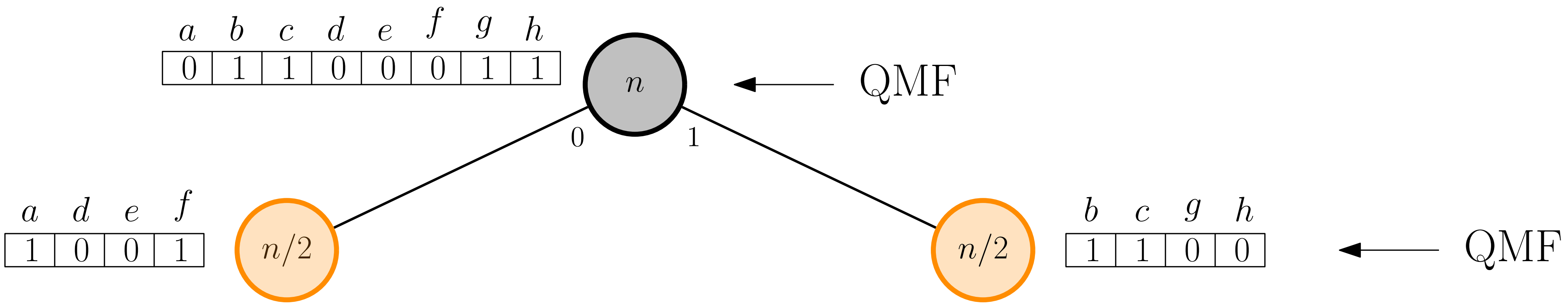
Quantum Divide and Conquer for Set Problems



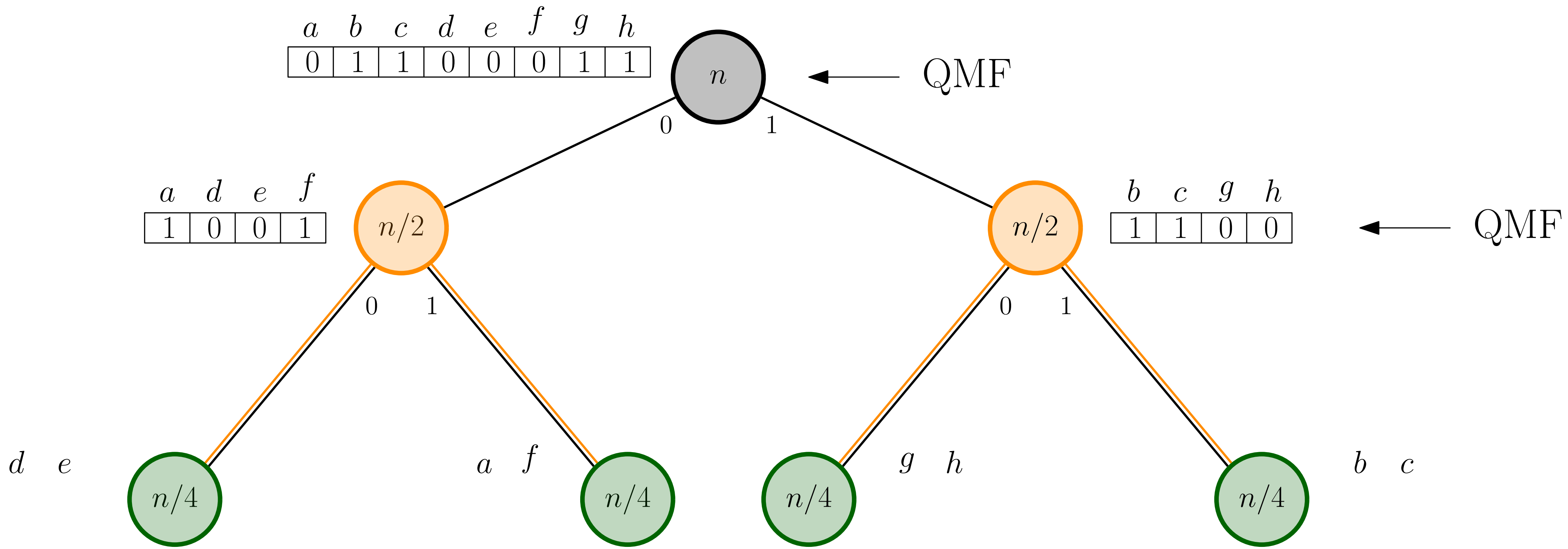
Quantum Divide and Conquer for Set Problems



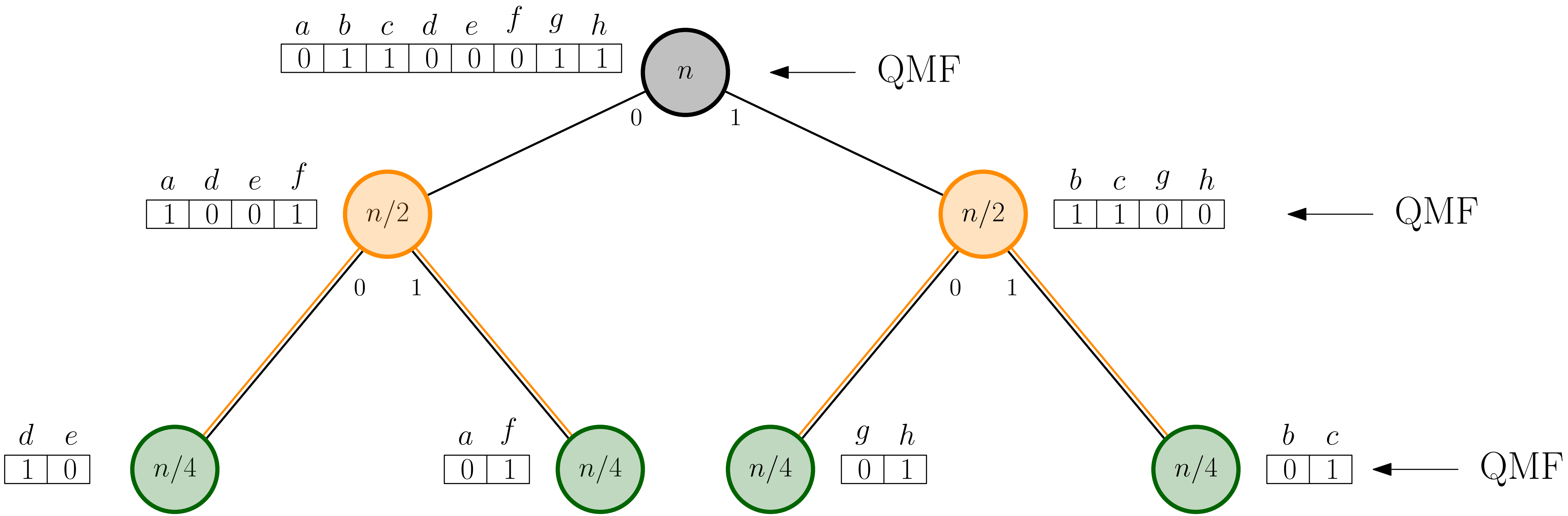
Quantum Divide and Conquer for Set Problems



Quantum Divide and Conquer for Set Problems



Quantum Divide and Conquer for Set Problems



Complexity Analysis

Complexity Analysis

- The running time of the algorithm obeys the recurrence

Complexity Analysis

- The running time of the algorithm obeys the recurrence
  - $Q(n) \leq \sqrt{\mathcal{O}\left(\binom{n}{n/2}\right)}\left(Q(\lfloor n/2 \rfloor) + Q(n/2) + poly(n)\right)$

Complexity Analysis

- The running time of the algorithm obeys the recurrence
  - $Q(n) \leq \sqrt{\mathcal{O}\left(\binom{n}{n/2}\right)}\left(Q(\lfloor n/2 \rfloor) + Q(n/2) + poly(n)\right)$
- The total running time is bounded by $\mathcal{O}^*(2^n)$

Complexity Analysis

- The running time of the algorithm obeys the recurrence
  - $Q(n) \leq \sqrt{\mathcal{O}\left(\binom{n}{n/2}\right)}\left(Q(\lfloor n/2 \rfloor) + Q(n/2) + poly(n)\right)$
- The total running time is bounded by $\mathcal{O}^*(2^n)$
- We can bound the space complexity in terms of qubits used during the computation

Complexity Analysis

- The running time of the algorithm obeys the recurrence
  - $Q(n) \leq \sqrt{\mathcal{O}\left(\binom{n}{n/2}\right)}\left(Q(\lfloor n/2 \rfloor) + Q(n/2) + poly(n)\right)$
- The total running time is bounded by $\mathcal{O}^*(2^n)$
- We can bound the space complexity in terms of qubits used during the computation
  - At height i of the computation tree we use $\frac{n}{2^i}$ qubits

Complexity Analysis

- The running time of the algorithm obeys the recurrence
  - $Q(n) \leq \sqrt{\mathcal{O}\left(\binom{n}{n/2}\right)}\left(Q(\lfloor n/2 \rfloor) + Q(n/2) + poly(n)\right)$
- The total running time is bounded by $\mathcal{O}^*(2^n)$
- We can bound the space complexity in terms of qubits used during the computation
  - At height i of the computation tree we use $\frac{n}{2^i}$ qubits
  - The computation tree has height $\log n$

Complexity Analysis

- The running time of the algorithm obeys the recurrence
  - $Q(n) \leq \sqrt{\mathcal{O}\left(\binom{n}{n/2}\right)}\left(Q(\lfloor n/2 \rfloor) + Q(n/2) + poly(n)\right)$
- The total running time is bounded by $\mathcal{O}^*(2^n)$
- We can bound the space complexity in terms of qubits used during the computation
  - At height i of the computation tree we use $\frac{n}{2^i}$ qubits
  - The computation tree has height $\log n$
- The number of qubits is at most $\sum_{i=0}^{\log n} \frac{n}{2^i} = 2n$

Complexity Analysis

- The running time of the algorithm obeys the recurrence
  - $Q(n) \leq \sqrt{\mathcal{O}\left(\binom{n}{n/2}\right)}\left(Q(\lfloor n/2 \rfloor) + Q(n/2) + poly(n)\right)$
- The total running time is bounded by $\mathcal{O}^*(2^n)$
- We can bound the space complexity in terms of qubits used during the computation
  - At height i of the computation tree we use $\frac{n}{2^i}$ qubits
  - The computation tree has height $\log n$
- The number of qubits is at most $\sum_{i=0}^{\log n} \frac{n}{2^i} = 2n$
  - Recall that the time complexity of the best classic algorithm using polynomial space is $\mathcal{O}^*(4^n)$

Take away

Take away

- Quantum computing can be used to improve classic bounds related to hard graph drawing problems

Take away

- Quantum computing can be used to improve classic bounds related to hard graph drawing problems
- Quantum dynamic programming, quantum minimum finding, and quantum divide and conquer are powerful tools for tackling several set-based problems

Take away

- Quantum computing can be used to improve classic bounds related to hard graph drawing problems
- Quantum dynamic programming, quantum minimum finding, and quantum divide and conquer are powerful tools for tackling several set-based problems
- A different perspecitve:

Take away

- Quantum computing can be used to improve classic bounds related to hard graph drawing problems
- Quantum dynamic programming, quantum minimum finding, and quantum divide and conquer are powerful tools for tackling several set-based problems
- A different perspecitve: Are there polynomial time solvable graph drawing problems whose current complexity bounds can be improved using quantum dynamic programming?

Thank you for your attention!