Нежное введение в криптоанализ систем на евклидовых решётках

Елена Киршанова

SIBECRYPT'19
Томский государственный университет, Россия
10 сентября 2019 г.

Балтийский федеральный университет имени Иммануила Канта

План

- Евклидовы решётки
- Задача обучения с ошибками
- Алгоритмы просеивания для нахождения короткого вектора
- Квантовые ускорения и открытые вопросы

Часть I

Евклидовы решётки

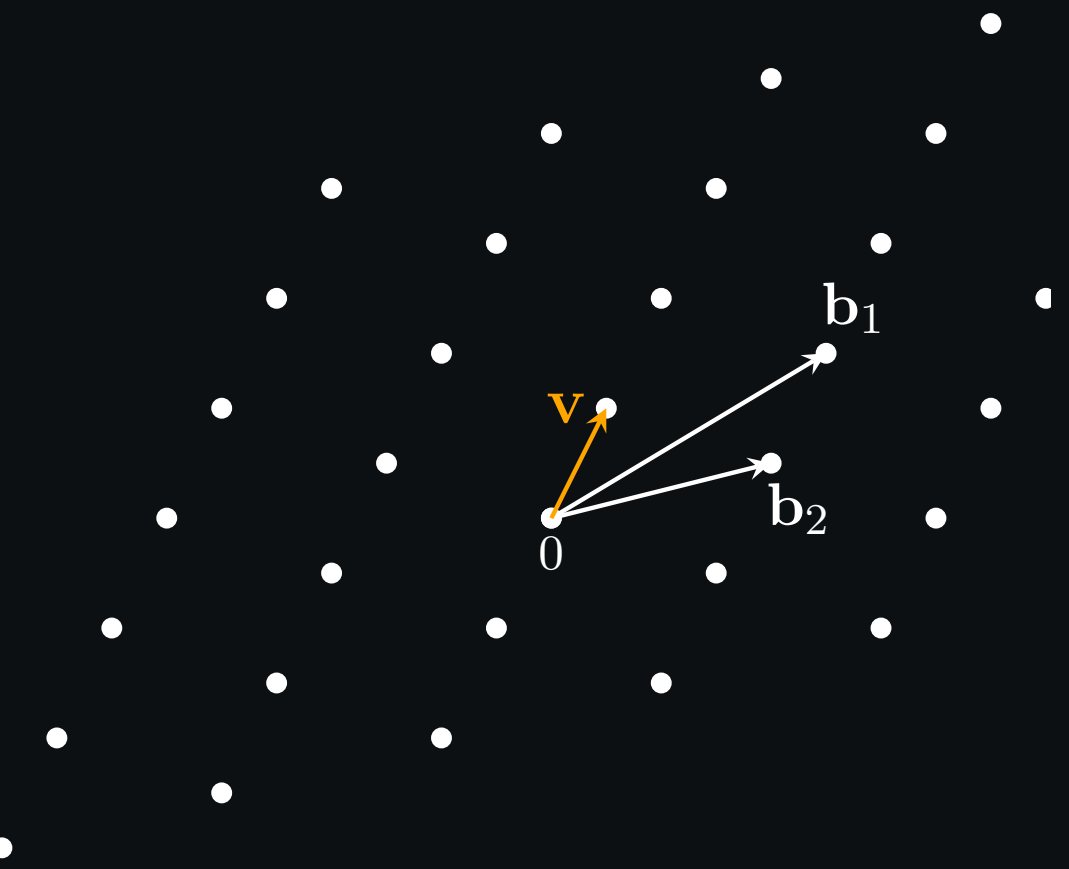
Определения

Решётка – это множество $\mathcal{L} = \{\sum_{i \le n} x_i \mathbf{b}_i \ : \ x_i \in \mathbb{Z}\}$ для лин. независимых $\mathbf{b}_i \in \mathbb{R}^n$

$\{\mathbf{b}_i\}_i$– базис $\mathcal{L}$

Минимум

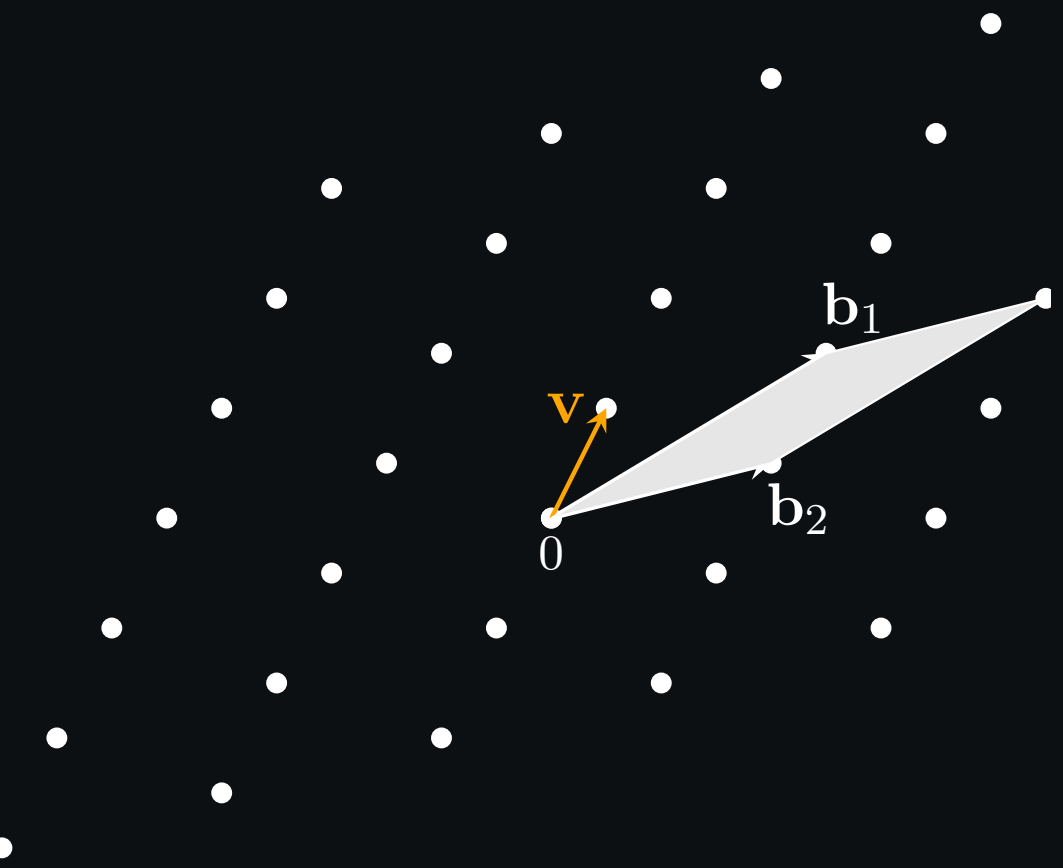
$$\lambda_1(\mathcal{L}) = \min_{\mathbf{v} \in \mathcal{L} \setminus \mathbf{0}} \|\mathbf{v}\|$$



Решётка – это множество $\mathcal{L} = \{\sum_{i \le n} x_i \mathbf{b}_i \ : \ x_i \in \mathbb{Z}\}$ для лин. независимых $\mathbf{b}_i \in \mathbb{R}^n$

$\{\mathbf{b}_i\}_i$– базис $\mathcal{L}$

Определения



Минимум

$$\lambda_1(\mathcal{L}) = \min_{\mathbf{v} \in \mathcal{L} \setminus \mathbf{0}} \|\mathbf{v}\|$$

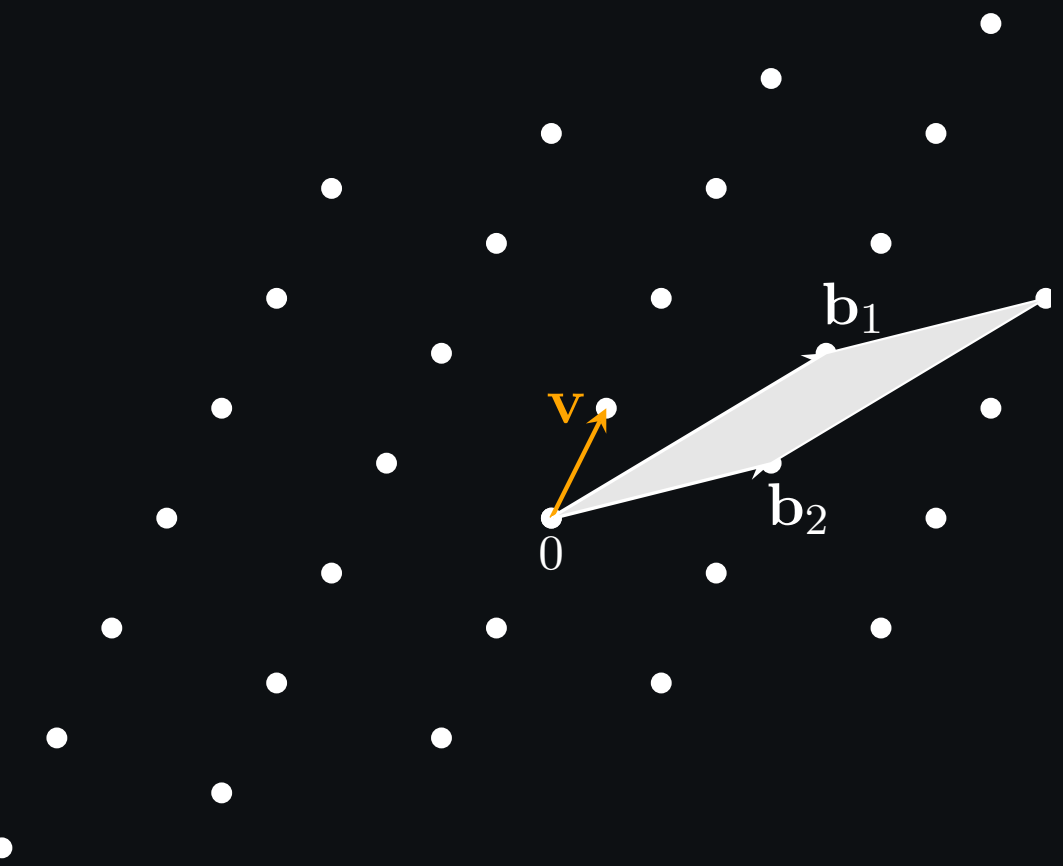
Определитель

$$\det(\mathcal{L}) = |\det(\mathbf{b}_i)_i|$$

Решётка – это множество $\mathcal{L} = \{\sum_{i \le n} x_i \mathbf{b}_i \ : \ x_i \in \mathbb{Z}\}$ для лин. независимых $\mathbf{b}_i \in \mathbb{R}^n$

$\{\mathbf{b}_i\}_i$– базис $\mathcal{L}$

Определения



Минимум

$$\lambda_1(\mathcal{L}) = \min_{\mathbf{v} \in \mathcal{L} \setminus \mathbf{0}} \|\mathbf{v}\|$$

Определитель

$$\det(\mathcal{L}) = |\det(\mathbf{b}_i)_i|$$

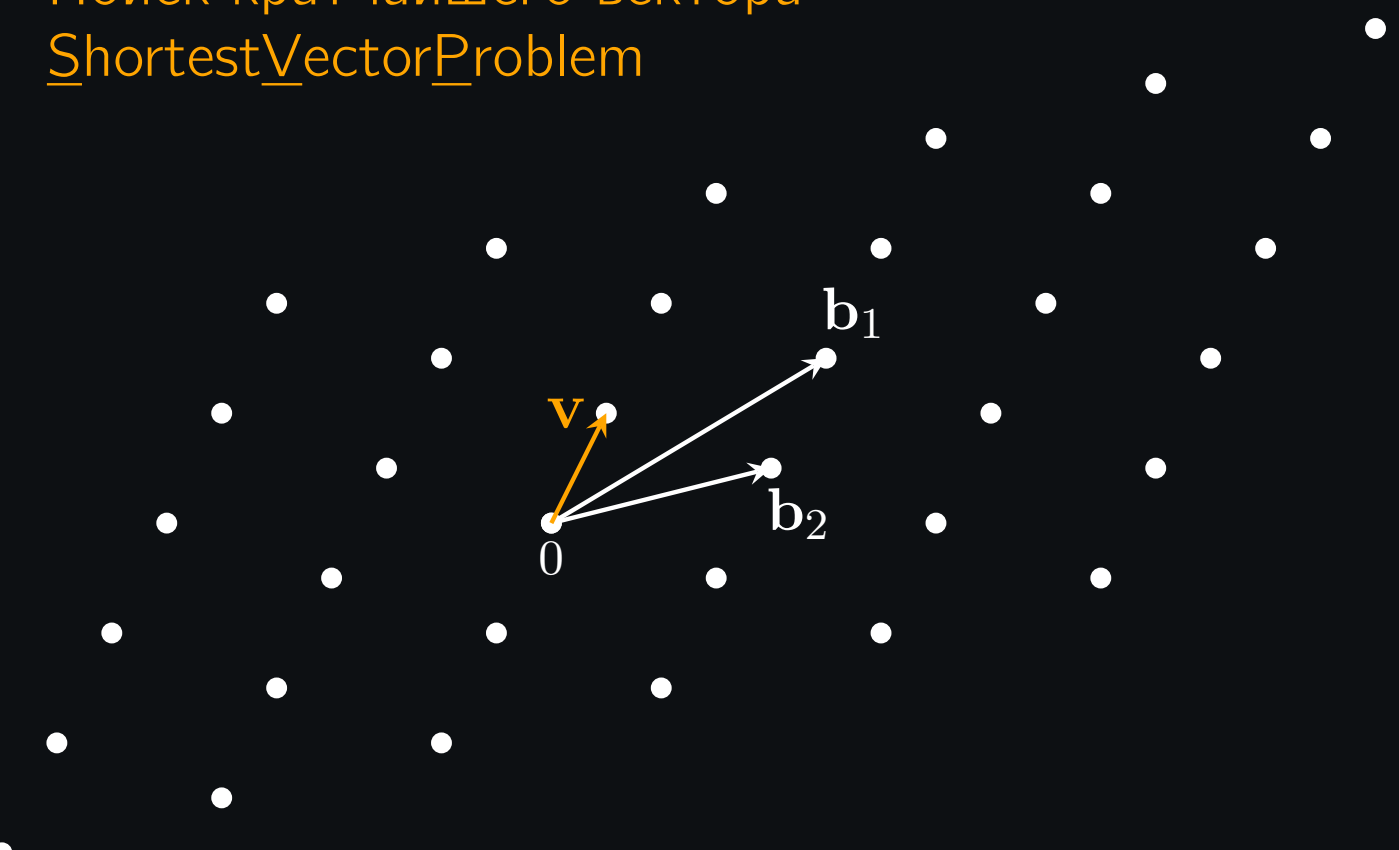
Граница Минковского

$$\lambda_1(\mathcal{L}) \leq \sqrt{n} \cdot \det(\mathcal{L})^{\frac{1}{n}}$$

Решётка – это множество $\mathcal{L} = \{\sum_{i \leq n} x_i \mathbf{b}_i \ : \ x_i \in \mathbb{Z}\}$ для лин. независимых $\mathbf{b}_i \in \mathbb{R}^n$

$\{\mathbf{b}_i\}_i$– базис $\mathcal{L}$

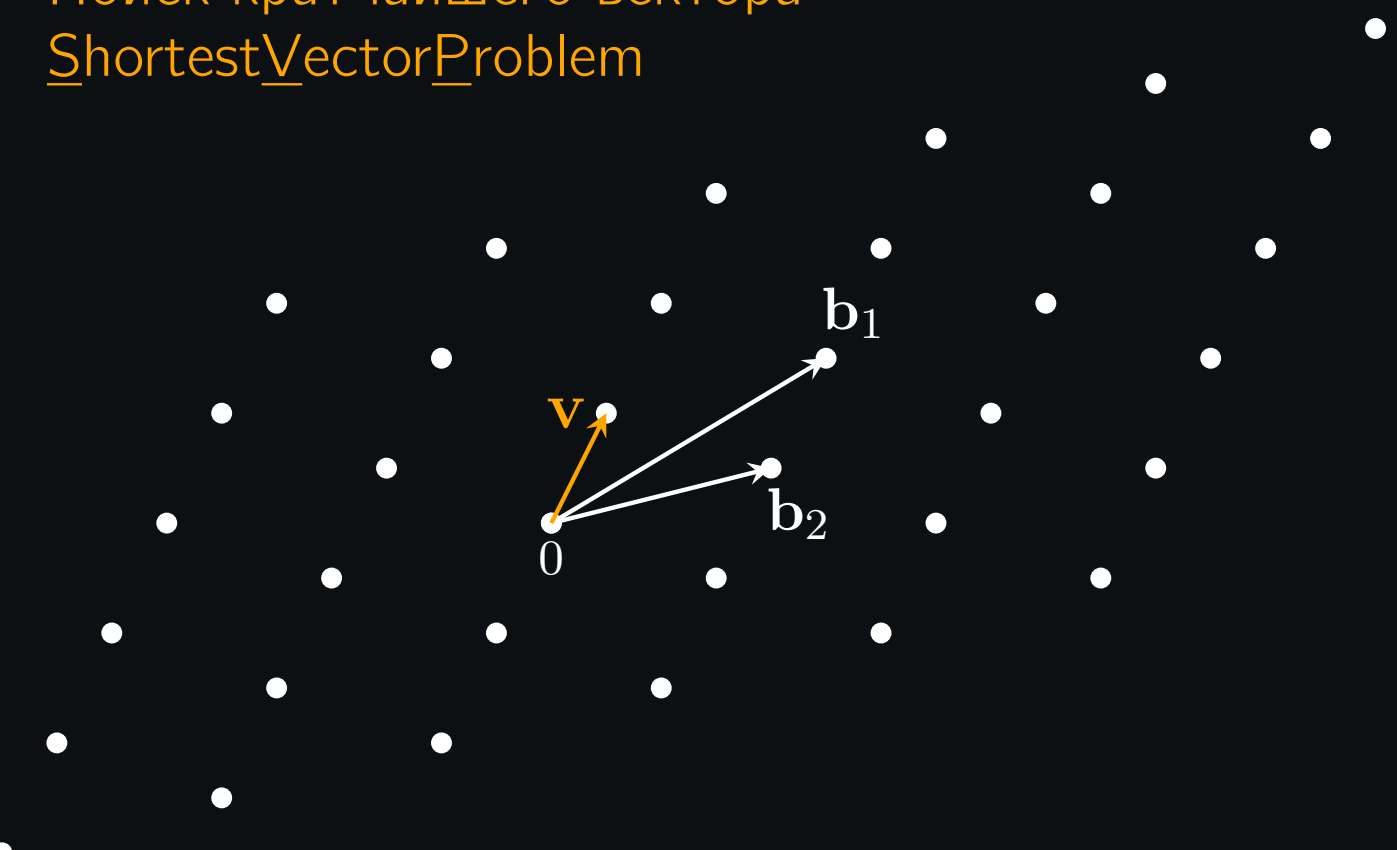
Поиск кратчайшего вектора
ShortestVectorProblem



В задаче поиска кратчайшего вектора (SVP) требуется найти $\mathbf{v}_{\text{shortest}} \in \mathcal{L}$:

$$\|\mathbf{v}_{\text{shortest}}\| = \lambda_1(\mathcal{L})$$

Поиск кратчайшего вектора
ShortestVectorProblem



В задаче поиска кратчайшего вектора (SVP) требуется найти $\mathbf{v}_{\text{shortest}} \in \mathcal{L}$:

$$\|\mathbf{v}_{\text{shortest}}\| = \lambda_1(\mathcal{L})$$

Упрощение: поиск аппроксимации (γ-SVP) к $\mathbf{v}_{\text{shortest}}$:

$$\|\mathbf{v}_{\text{short}}\| \leq \gamma \cdot \lambda_1(\mathcal{L})$$

Поиск ближайшего вектора CVP / BDD

В задаче поиска ближайшего вектора (CVP) для $\mathbf{t} \notin \mathcal{L}$ требуется найти $\mathbf{v} \in \mathcal{L}$:

$$\|\mathbf{v} - \mathbf{t}\| \text{ минимально для всех } \mathbf{v} \in \mathcal{L}$$

Поиск ближайшего вектора CVP / BDD

В задаче поиска ближайшего вектора (CVP) для $\mathbf{t} \notin \mathcal{L}$ требуется найти $\mathbf{v} \in \mathcal{L}$:

$$\|\mathbf{v} - \mathbf{t}\| \text{ минимально для всех } \mathbf{v} \in \mathcal{L}$$

Часто: $\text{dist}(t, \mathcal{L}) \leq \frac{1}{\gamma}\lambda_1(\mathcal{L})$. Получаем задачу Декодирования с Ограниченным расстоянием, γ-BDD

Поиск ближайшего вектора CVP / BDD

Для решения BDD в $\mathcal{L}$, вызываем оракул для approx-SVP в “связанной” решетке р-ти+ 1.

В задаче поиска ближайшего вектора (CVP) для $\mathbf{t} \notin \mathcal{L}$ требуется найти $\mathbf{v} \in \mathcal{L}$:

$$\|\mathbf{v} - \mathbf{t}\| \text{ минимально для всех } \mathbf{v} \in \mathcal{L}$$

Часто: $\text{dist}(t, \mathcal{L}) \leq \frac{1}{\gamma}\lambda_1(\mathcal{L})$. Получаем задачу Декодирования с Ограниченным расстоянием, γ-BDD

От задачи BDD к approxSVP: вложение Каннана

Для задачи BDD $(\mathcal{L}, \mathbf{t})$, где $\mathcal{L}$ имеет базиз B, рассмотрим для константы c

$$B' = \begin{bmatrix} B & \mathbf{t} \\ \mathbf{0} & \mathrm{c} \end{bmatrix}.$$

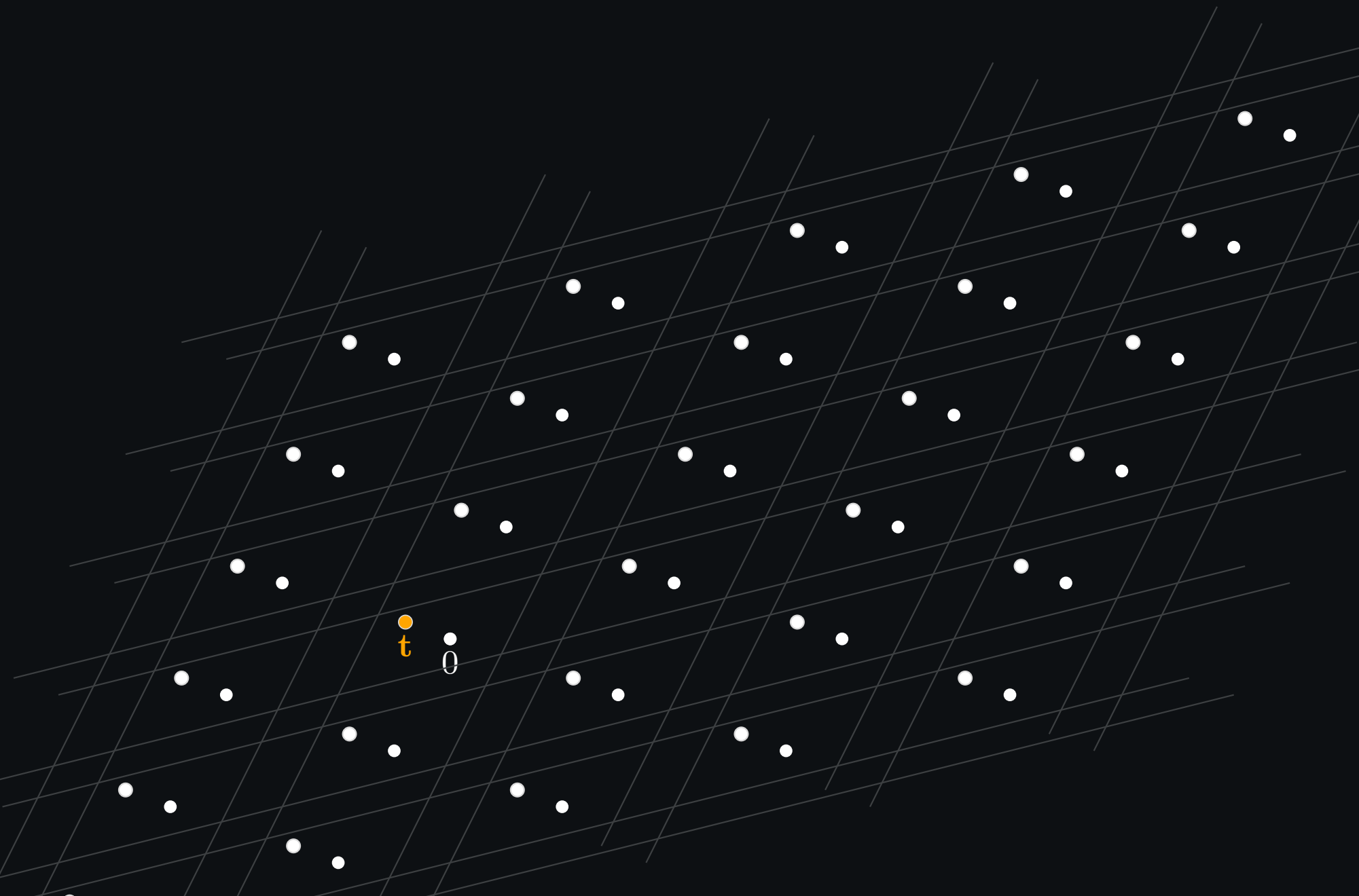
- столбцы B' лин. независимы
- Для "грамотно" выбранной c и $\mathbf{t}$ - достаточно близкого к $\mathcal{L}$,

$$\begin{bmatrix} B & \mathbf{t} \\ \mathbf{0} & \mathrm{c} \end{bmatrix} \cdot \begin{bmatrix} \mathbf{x} \\ -1 \end{bmatrix} = \begin{bmatrix} B\mathbf{x} - \mathbf{t} \\ \mathrm{c} \end{bmatrix}$$

– короткий вектора в $\mathcal{L}(B')$ (намного короче, чем любой $\mathbf{v} \in \mathcal{L}(B')$ не параллельный ему).

От задачи BDD к approxSVP: вложение Каннана

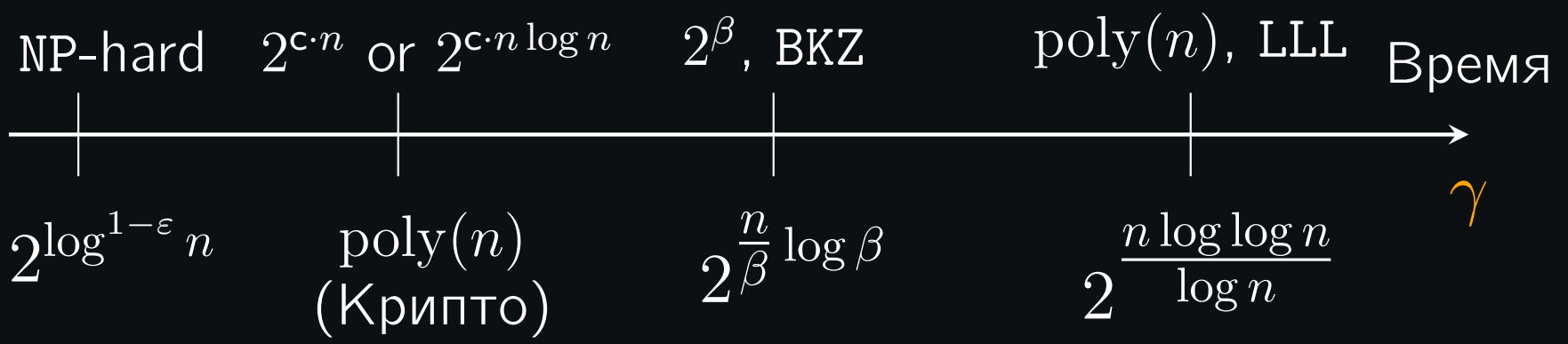
От задачи BDD к approxSVP: вложение Каннана



Асимптотическая сложность SVP ($o()$ опущены)

$$||\mathbf{v}_{\text{shortest}}|| \leq \sqrt{n} \cdot \det(\mathcal{L})^{1/n}$$

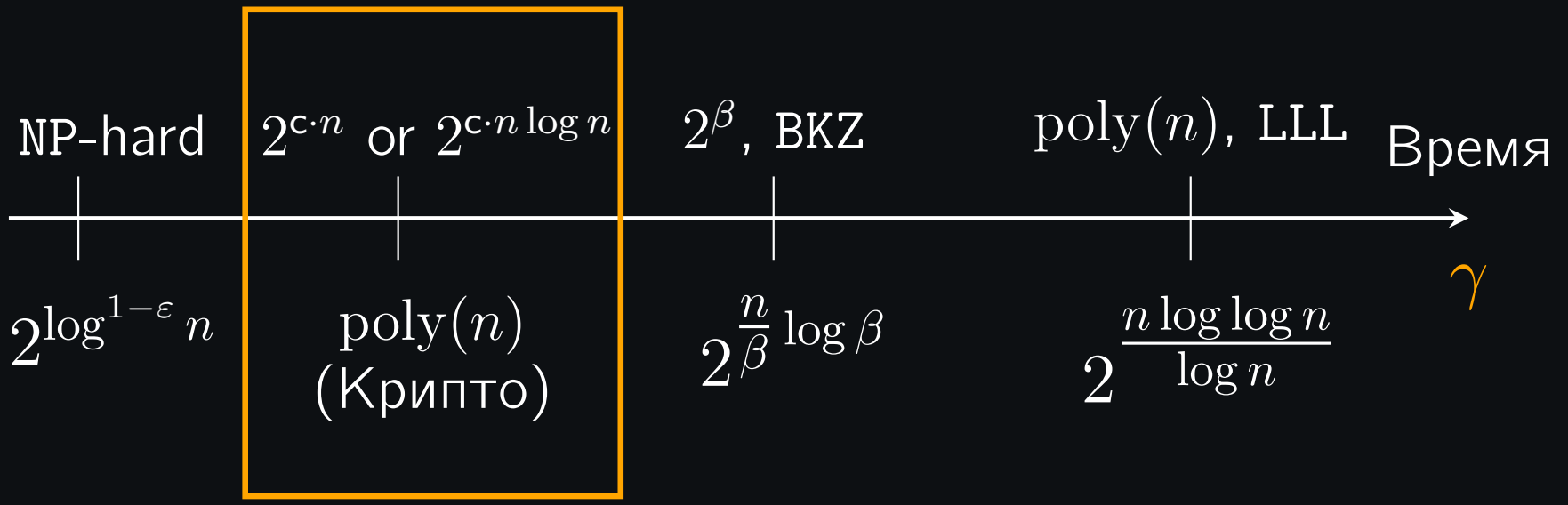
$$||\mathbf{v}_{\text{short}}|| \leq \gamma \cdot ||\mathbf{v}_{\text{shortest}}||$$



Асимптотическая сложность SVP ($o()$ опущены)

$$||\mathbf{v}_{\text{shortest}}|| \leq \sqrt{n} \cdot \det(\mathcal{L})^{1/n}$$

$$||\mathbf{v}_{\text{short}}|| \leq \gamma \cdot ||\mathbf{v}_{\text{shortest}}||$$

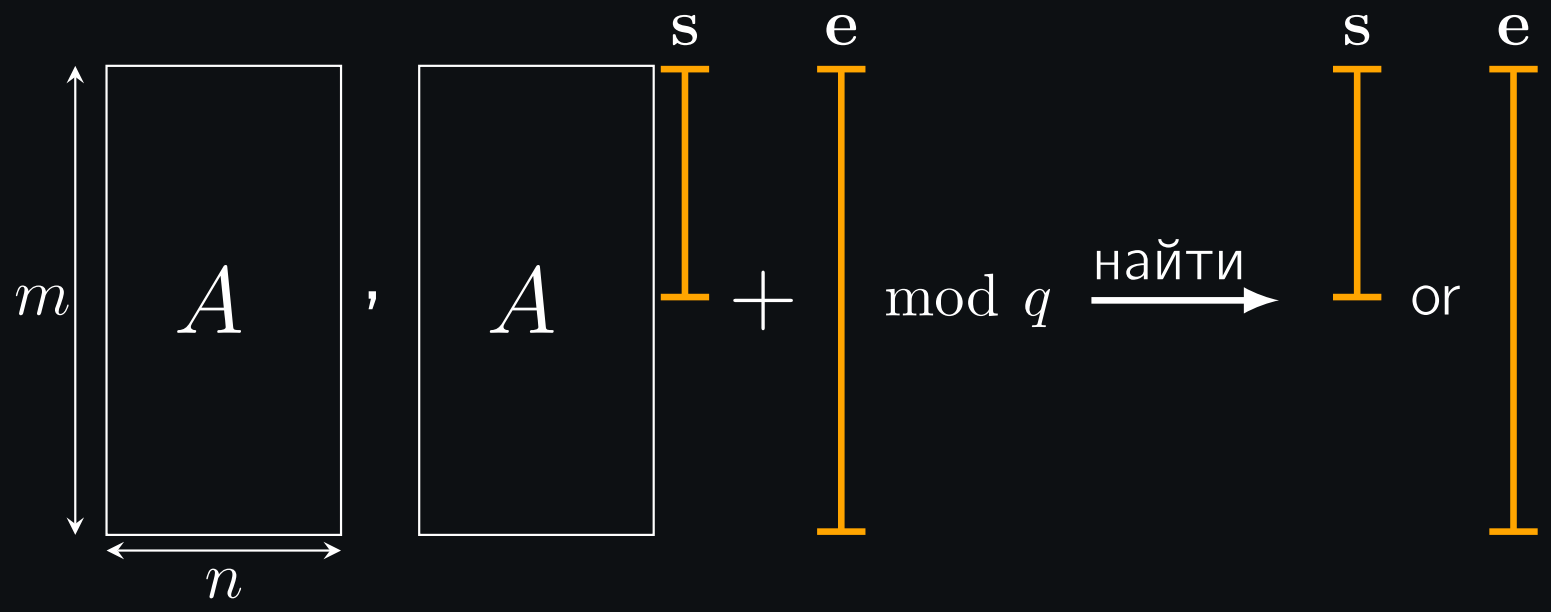


- Просеивание (эвристический):
  Время(exactSVP) $= 2^{0.292n}$ Память $= 2^{0.2075n}$
- Перечисление:
  Время(exactSVP) $= 2^{(1/2e)n \log n}$ Память $= \text{poly}(n)$

Часть II

Задача обучения с ошибками (The Learning with Errors problem, LWE)

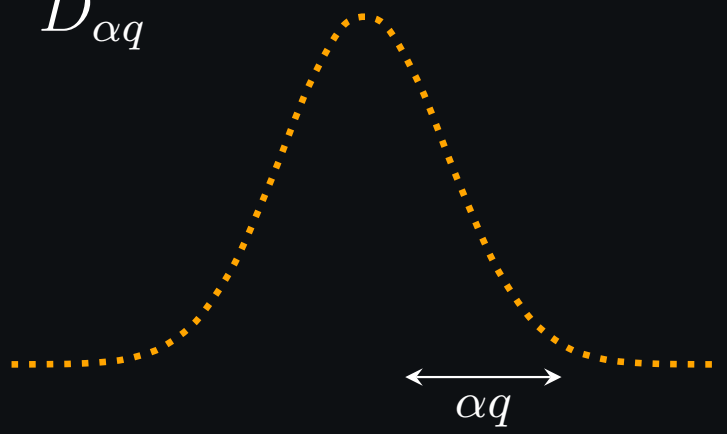
LWE (Regev'05)



$$A \xleftarrow{\$} \mathbb{Z}_q^{m \times n}$$

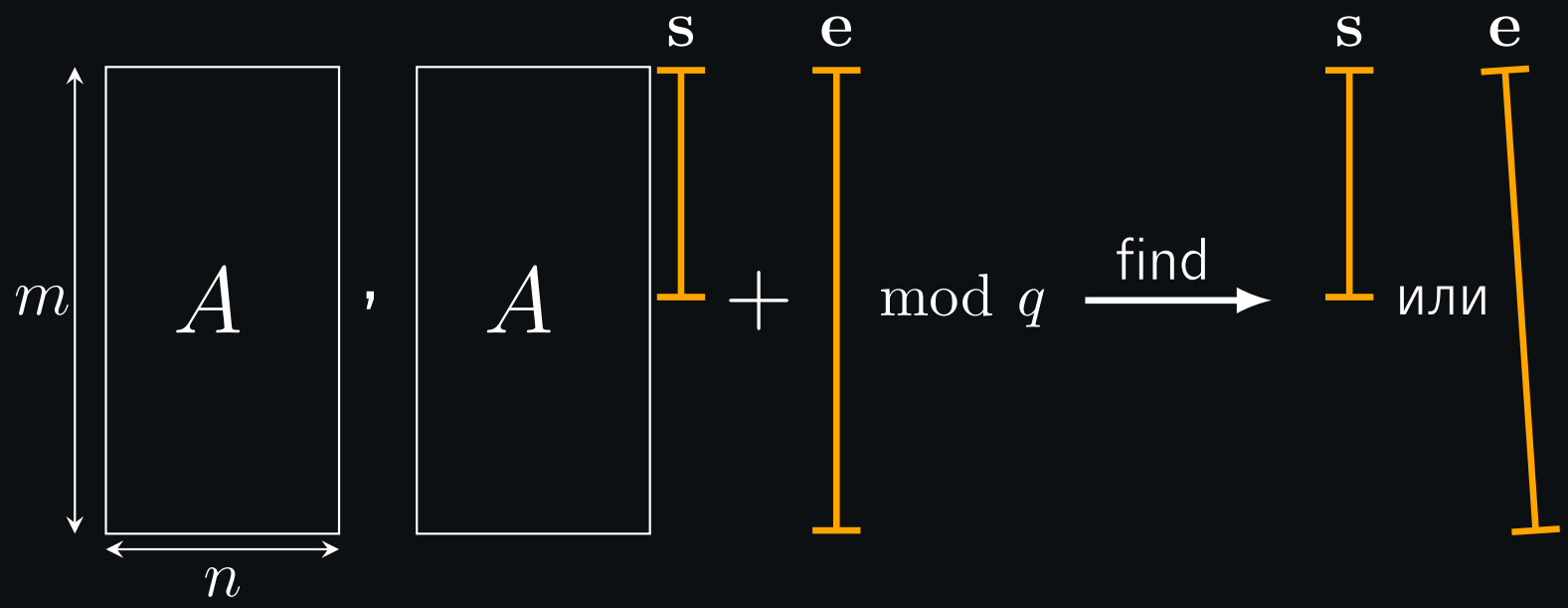
$$\mathbf{s} \xleftarrow{\$} \mathbb{Z}_q^{n}$$

$$\mathbf{e} \leftarrow D_{\alpha q}^{m}$$



Часто: $n = \Theta(\text{bit security})$, $q = n^{\Theta(1)}$, $m = \Theta(n \log q)$, $\alpha = \sqrt{n}/q$.

LWE is BDD

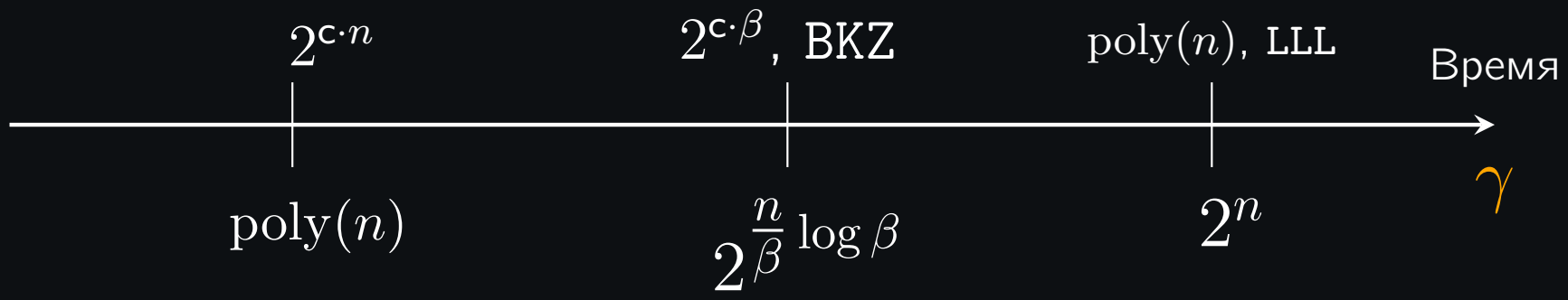


- A задает решётку A-Конструкции (Construction-A)

$$\mathcal{L}_q(A) = A\mathbb{Z}_q^n + q\mathbb{Z}^m$$

- $\mathcal{L}_q(A)$ ранга m и $\det(\mathcal{L}_q(A)) = q^{m-n}$.
- $As + e \bmod q$ – вектор, на расстоянии $\Theta(\sqrt{m}\alpha q)$ от $\mathcal{L}_q(A)$
- $(A, As + e)$ – пример BDD задачи для $\mathcal{L}_q(A)$ с $\gamma = \frac{q^{1-n/m}}{\alpha q}$

Сложность LWE при атаках на решётках ($o()$ опущены), [AGVW17,HKM17]



Для параметров LWE (n, m, q, α), $\gamma = \frac{q^{1-n/m}}{\alpha q}$

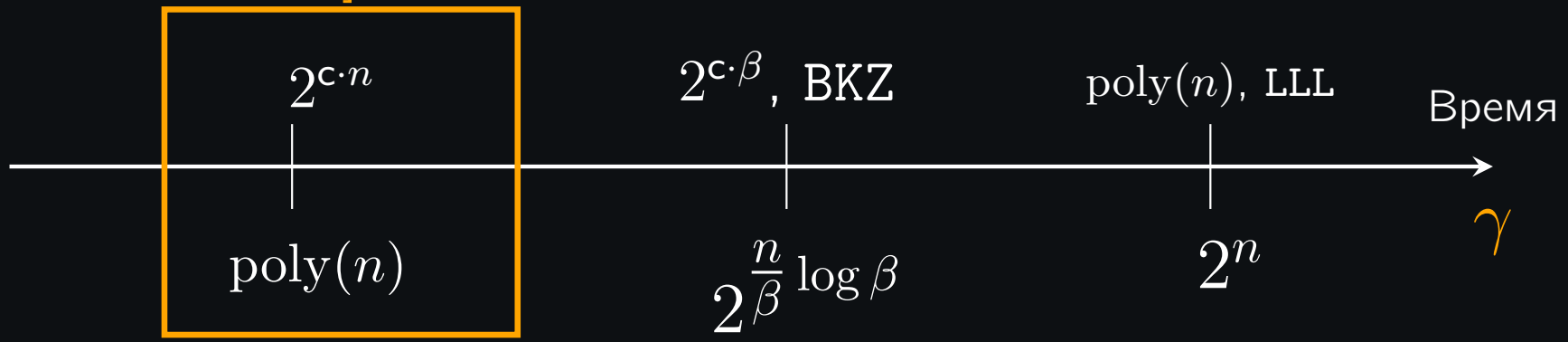
$$T(\mathrm{LWE}) = \exp\left(c \cdot \frac{\lg q}{\lg^2 \alpha} \lg\left(\frac{n \lg q}{\lg^2 \alpha}\right) \cdot n\right)$$

Это выражение получено решением для β

$$2^{\frac{m}{\beta} \log \beta} = \frac{q^{1-n/m}}{\alpha q}$$

и выбором $m = \Omega(n)$, минимизирующем решение.

Сложность LWE при атаках на решётках ($o()$ опущены), [AGVW17,HKM17]



Для параметров LWE (n, m, q, α), $\gamma = \frac{q^{1-n/m}}{\alpha q}$

$$T(\mathrm{LWE}) = \exp\left(c \cdot \frac{\lg q}{\lg^2 \alpha} \lg\left(\frac{n \lg q}{\lg^2 \alpha}\right) \cdot n\right)$$

Это выражение получено решением для β

$$2^{\frac{m}{\beta} \log \beta} = \frac{q^{1-n/m}}{\alpha q}$$

и выбором $m = \Omega(n)$, минимизирующем решение.

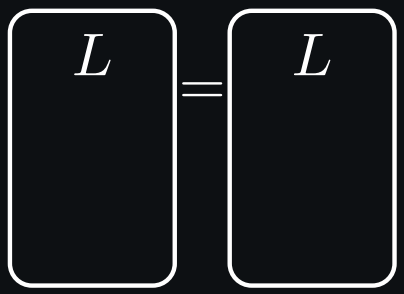
Часть III

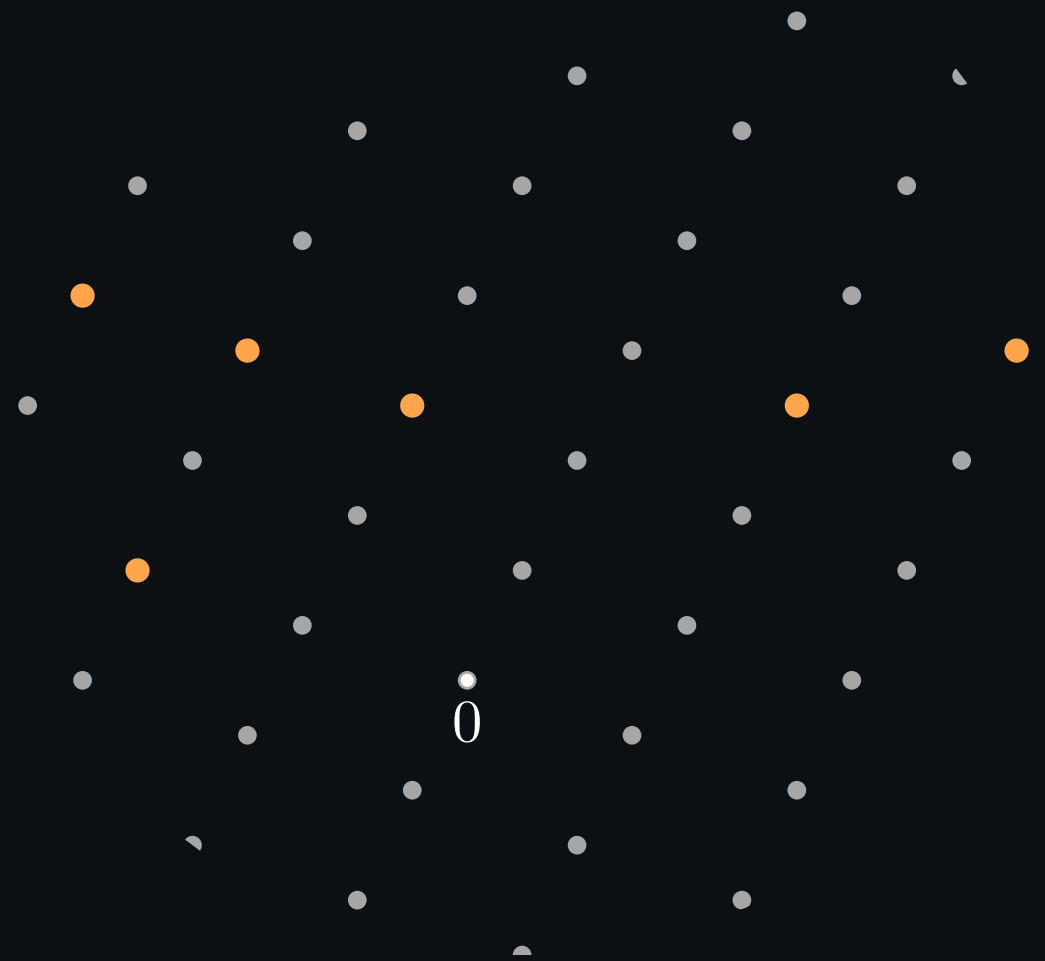
Алгоритмы просеивания

https://sites.google.com/a/x.bestledlights.cf/a223/1-32772214555

2-Просеивание (Nguyen-Vidick)

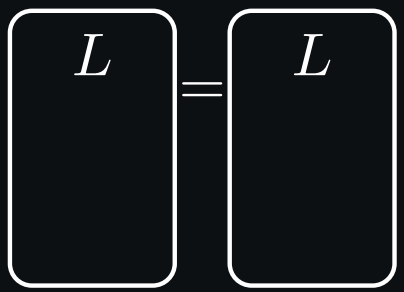
Идея: насыщать пространство векторами решётки так, чтобы суммы их пар давали короткие вектора

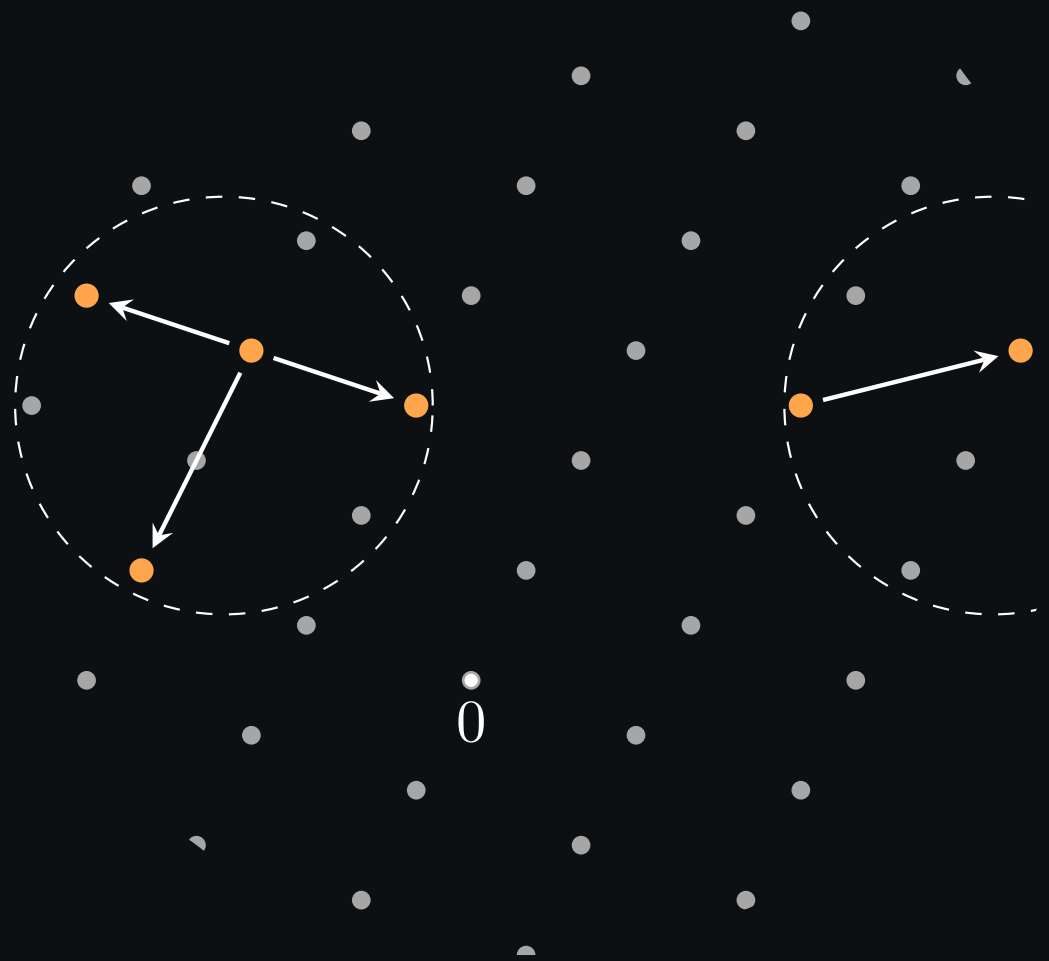




2-Просеивание (Nguyen-Vidick)

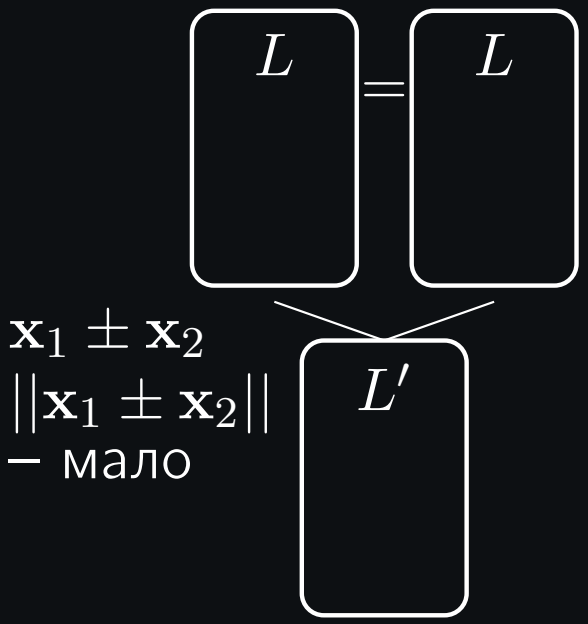
Идея: насыщать пространство векторами решётки так, чтобы суммы их пар давали короткие вектора

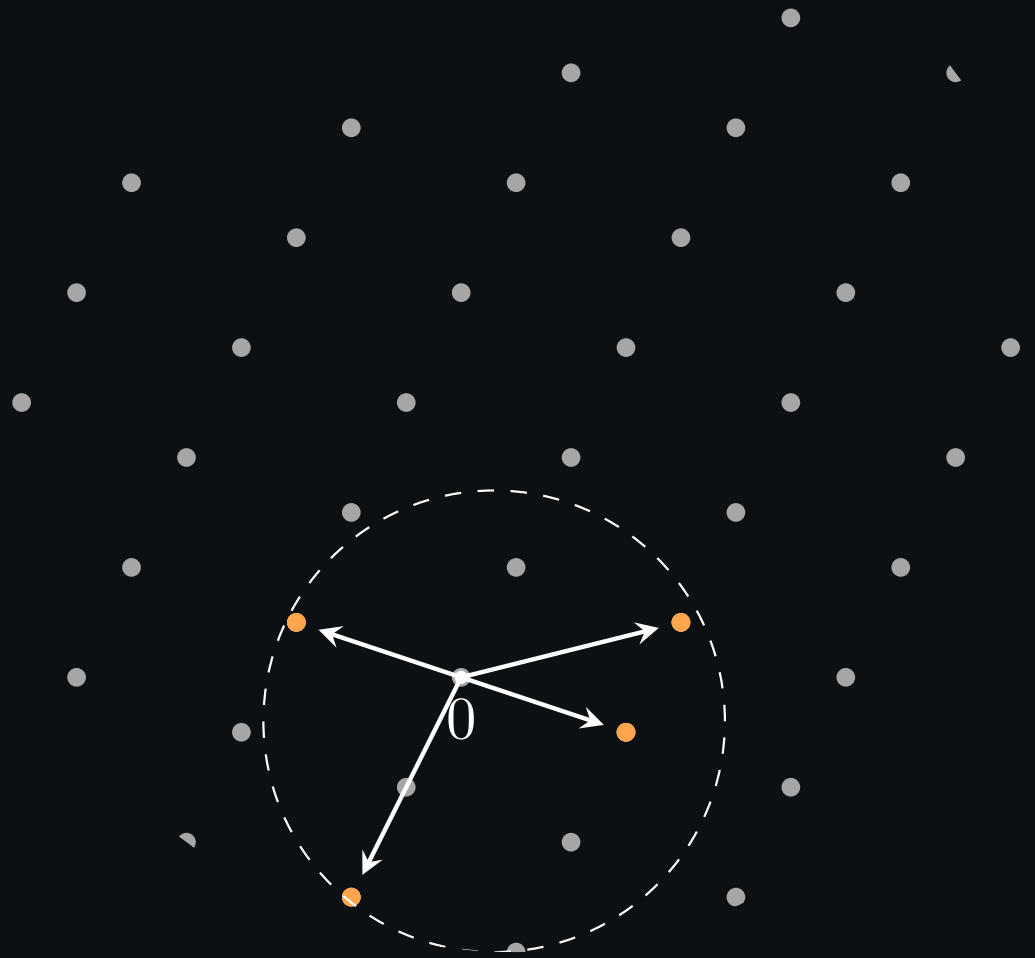




2-Просеивание (Nguyen-Vidick)

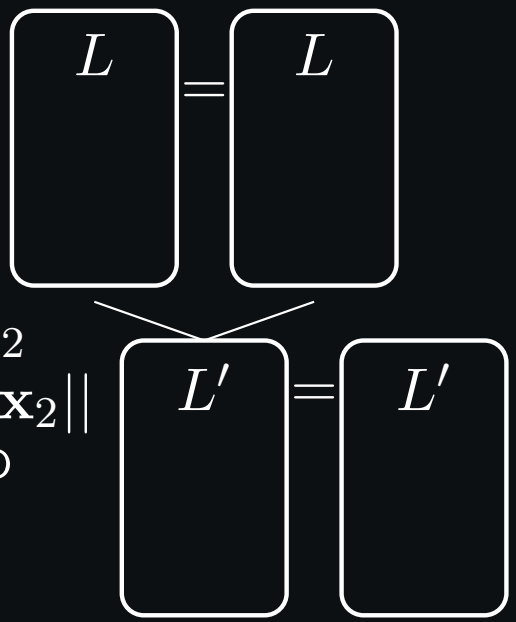
Идея: насыщать пространство векторами решётки так, чтобы суммы их пар давали короткие вектора



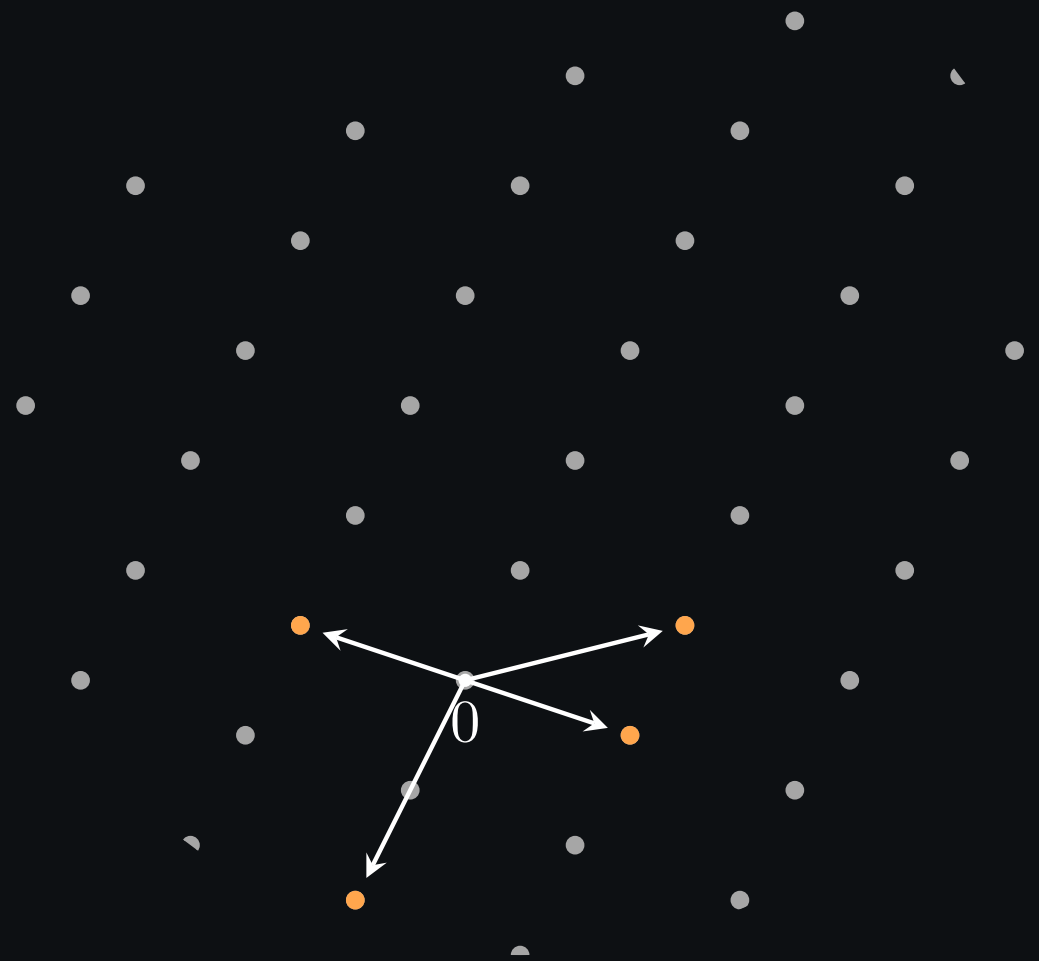


2-Просеивание (Nguyen-Vidick)

Идея: насыщать пространство векторами решётки так, чтобы суммы их пар давали короткие вектора

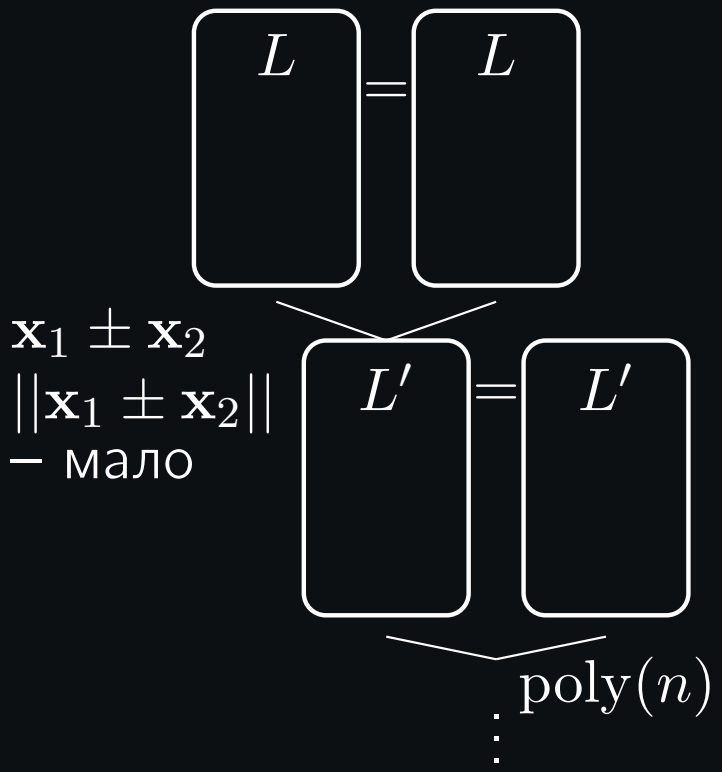


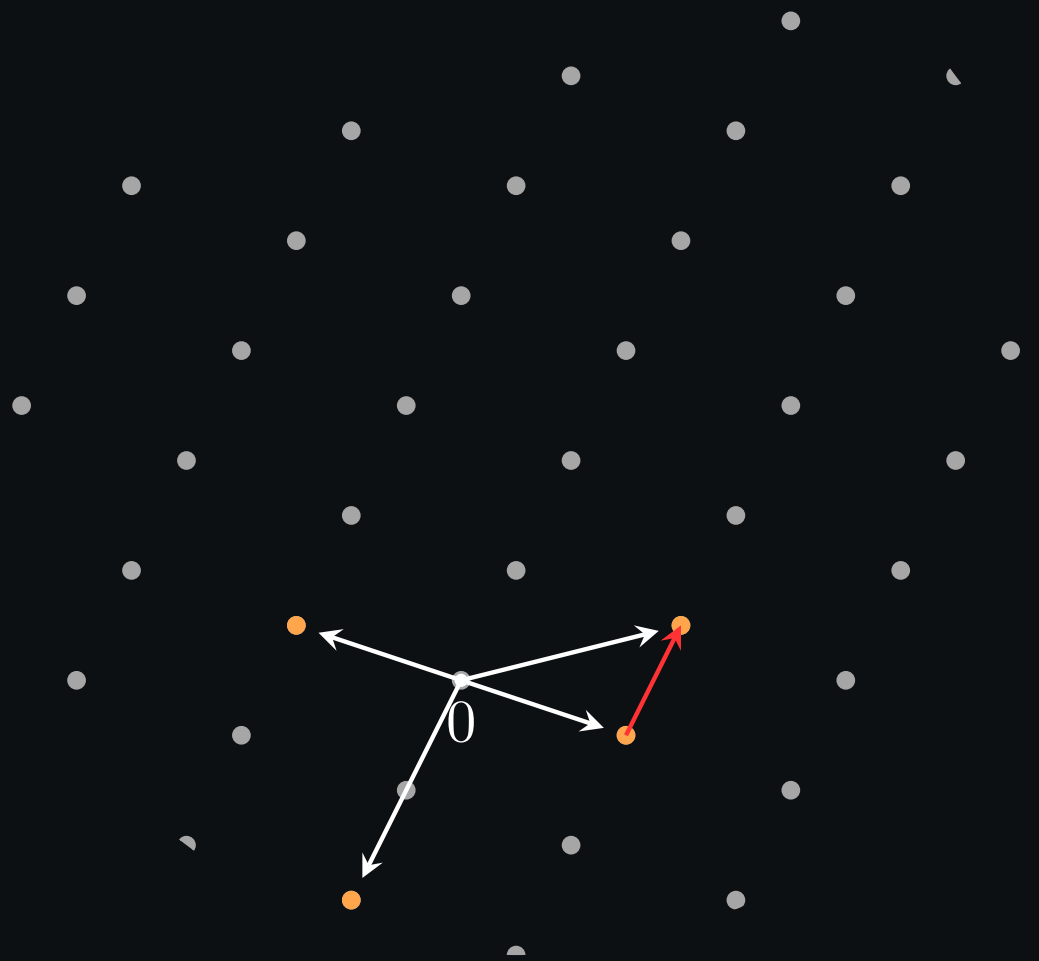
$\mathbf{x}_1 \pm \mathbf{x}_2$
$||\mathbf{x}_1 \pm \mathbf{x}_2||$
– мало



2-Просеивание (Nguyen-Vidick)

Идея: насыщать пространство векторами решётки так, чтобы суммы их пар давали короткие вектора

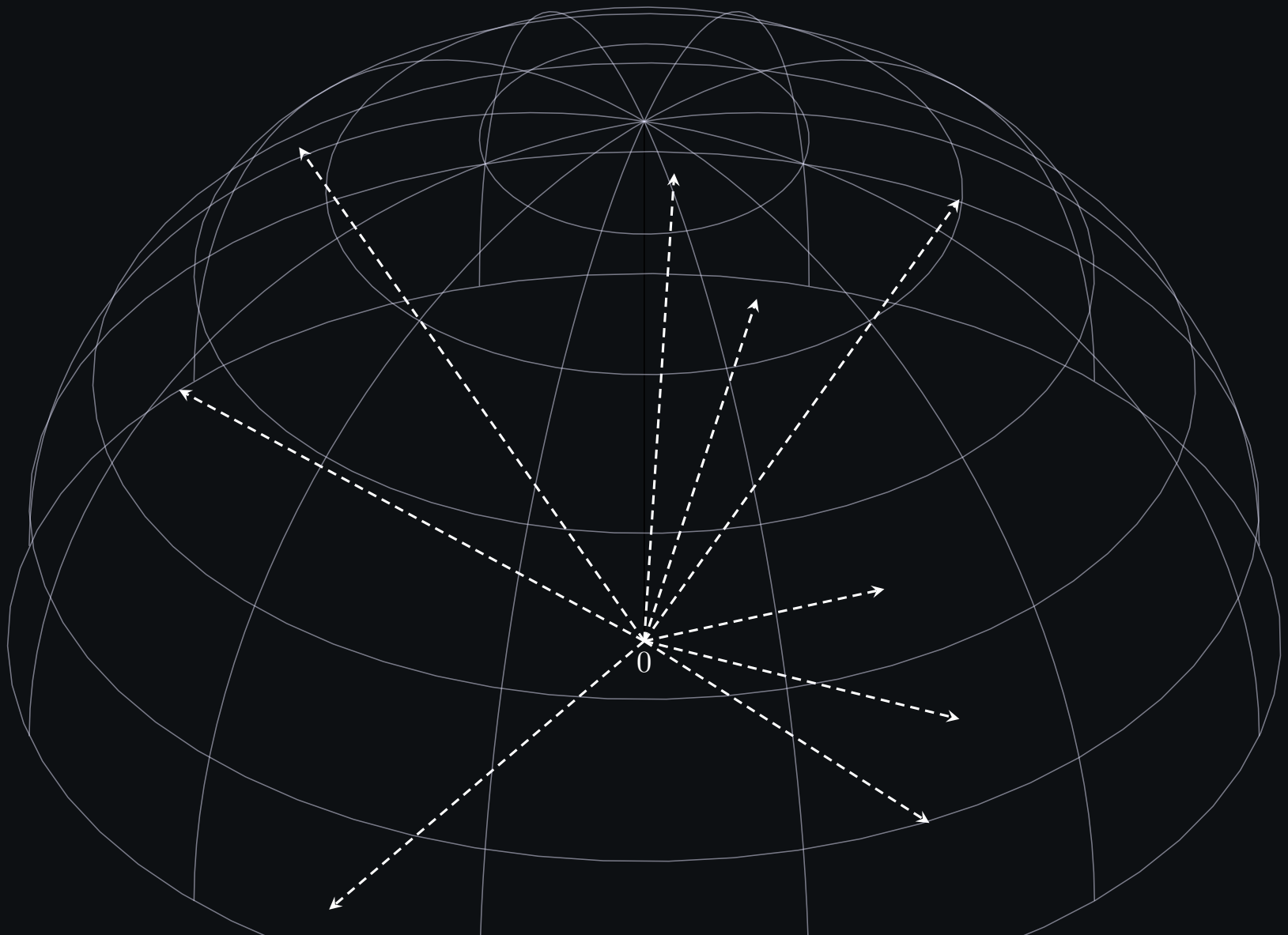




Насколько большим должно быть $|L|$?

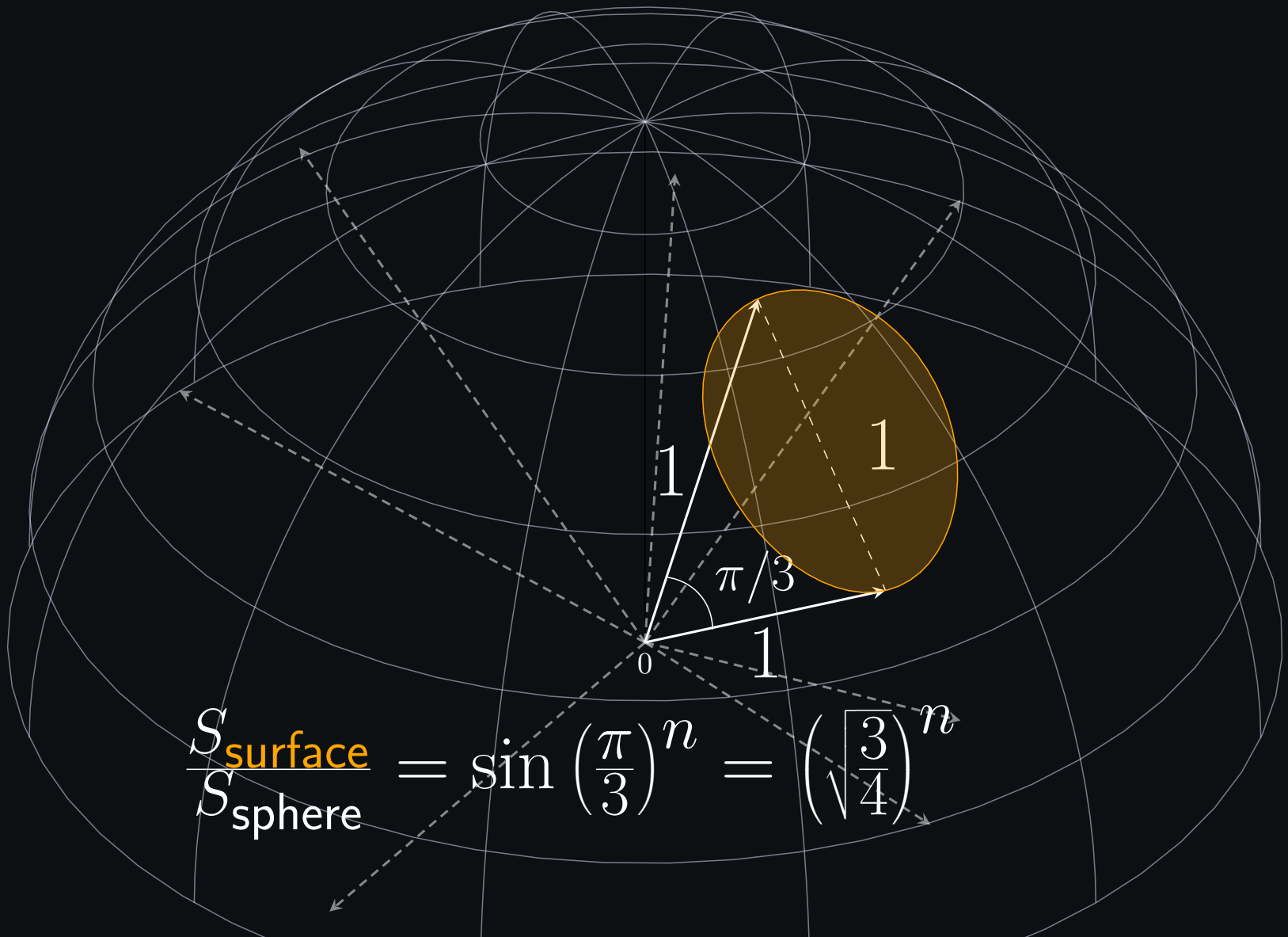
Насколько большим должно быть $|L|$?

Предположение: вектора (нормализованные) в $|L|$ равномерно независимо распределены в S^{n-1}.



Насколько большим должно быть $|L|$?

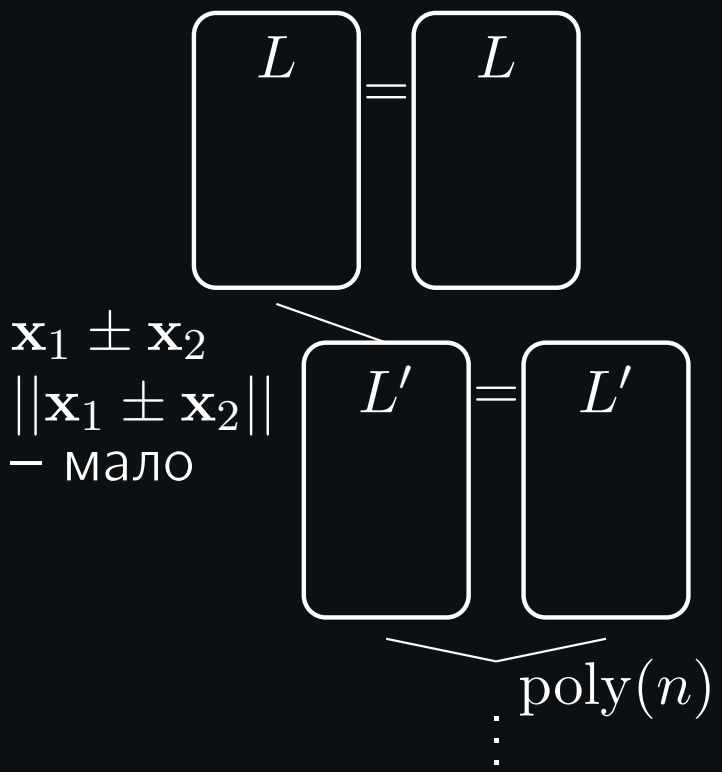
Предположение: вектора (нормализованные) в $|L|$ равномерно независимо распределены в S^{n-1}.



$$\frac{S_{\text{surface}}}{S_{\text{sphere}}} = \sin\left(\frac{\pi}{3}\right)^n = \left(\sqrt{\frac{3}{4}}\right)^n$$

2-Просеивание (Nguyen-Vidick)

Идея: насыщать пространство векторами решётки так, чтобы суммы их пар давали короткие вектора



$$|L| = \left(\sqrt{\frac{3}{4}}\right)^{-n} = 2^{0.2075n}$$

$$T(\text{2-Просеивания}) = |L|^2 = 2^{0.415n}$$

o(n) опушены

Улучшенное просеивание с хэшированием

Как достичь $T = 2^{0.292n+o(n)}$?

Использовать метод "Ближнего соседа"(Near Neighbour)!
(aka Locality-Sensitive techniques)

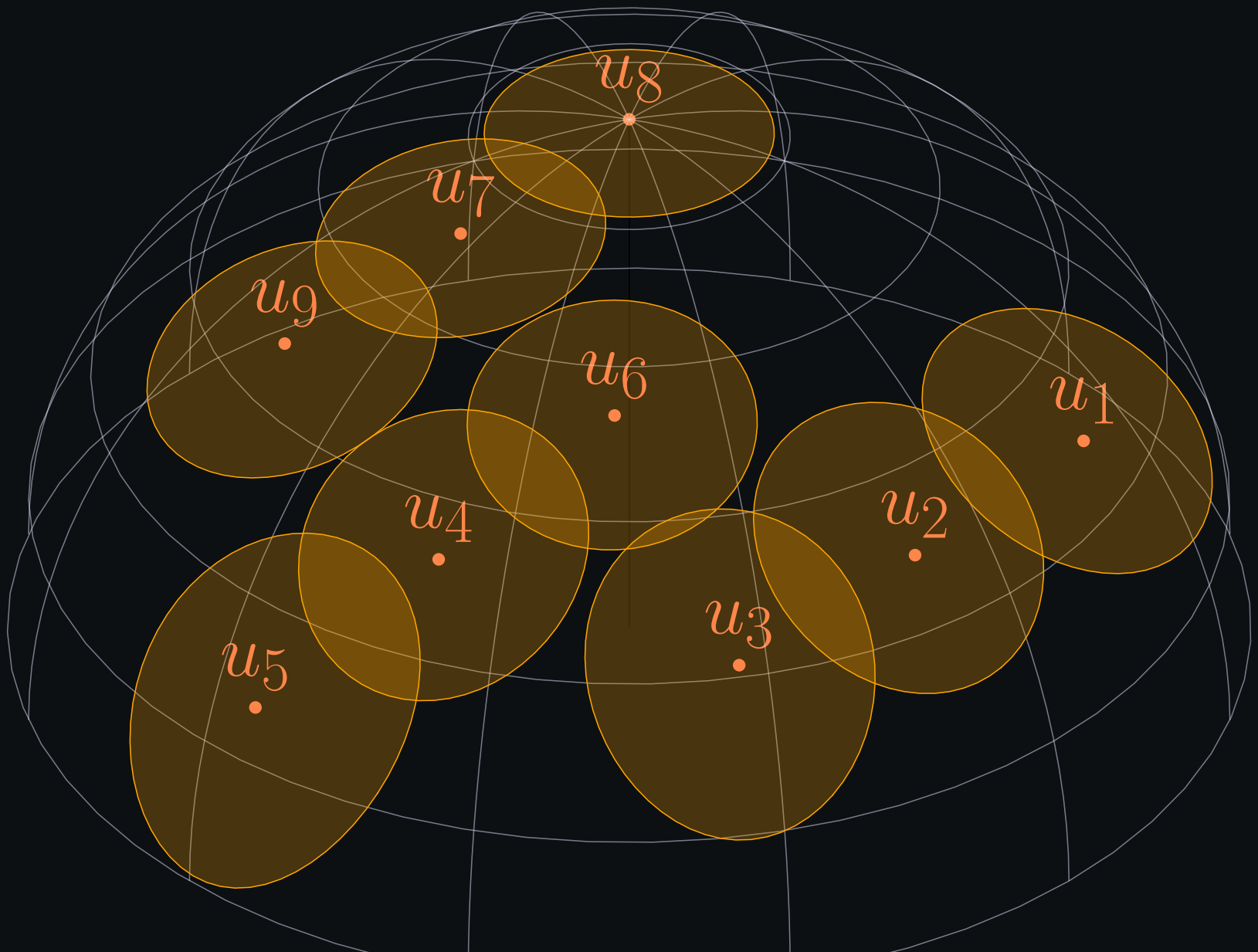
Улучшенное просеивание с хэшированием

Как достичь $T = 2^{0.292n+o(n)}$?

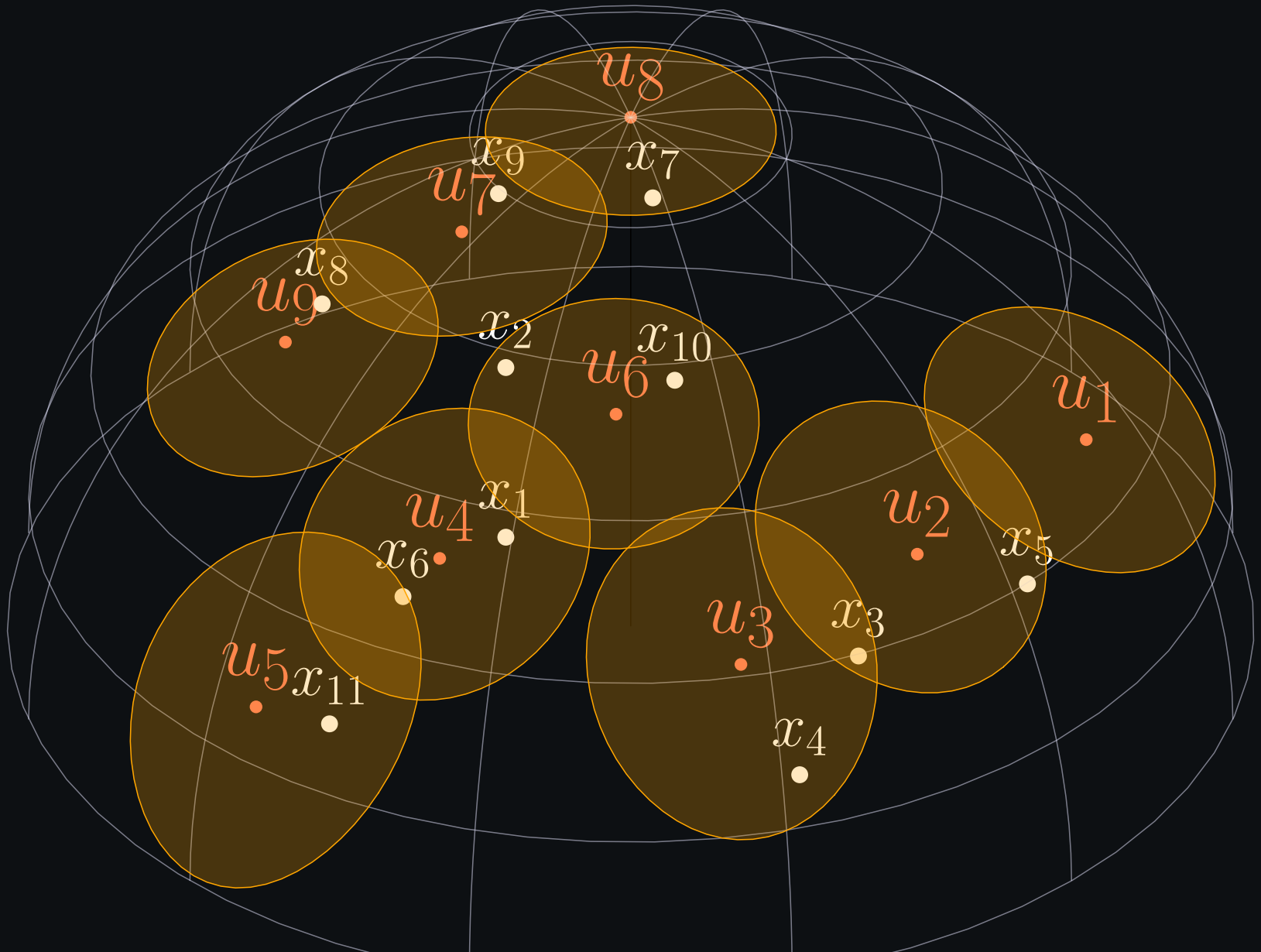
Использовать метод "Ближнего соседа"(Near Neighbour)!
(aka Locality-Sensitive techniques)

$$
\begin{aligned}
||\mathbf{x}_1 - \mathbf{x}_2|| < ||\mathbf{x}_1|| &\iff \\
||\mathbf{x}_1 - \mathbf{x}_2||^2 < ||\mathbf{x}_1||^2 &\iff \\
||\mathbf{x}_1||^2 - 2\langle \mathbf{x}_1, \mathbf{x}_2 \rangle + ||\mathbf{x}_2||^2 < ||\mathbf{x}_1||^2 &\iff \\
\langle \mathbf{x}_1, \mathbf{x}_2 \rangle > \frac{1}{2}||\mathbf{x}_2||^2 &
\end{aligned}
$$

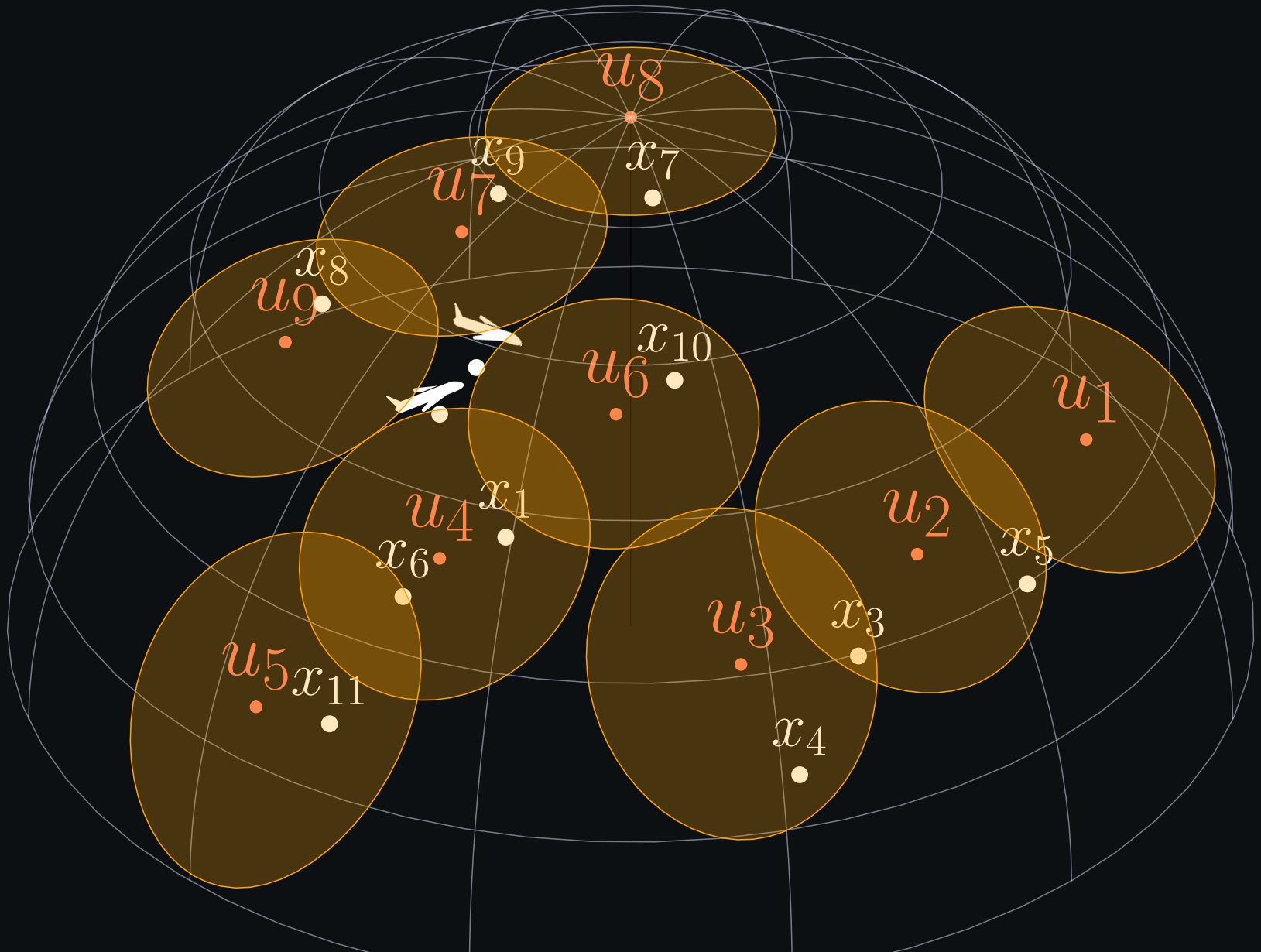
Locality-sensitive filtering (Фильтрация по расстоянию) [BGJ15, BDGL16]



Locality-sensitive filtering (Фильтрация по расстоянию)[BGJ15, BDGL16]



Locality-sensitive filtering (Фильтрация по расстоянию)[BGJ15, BDGL16]



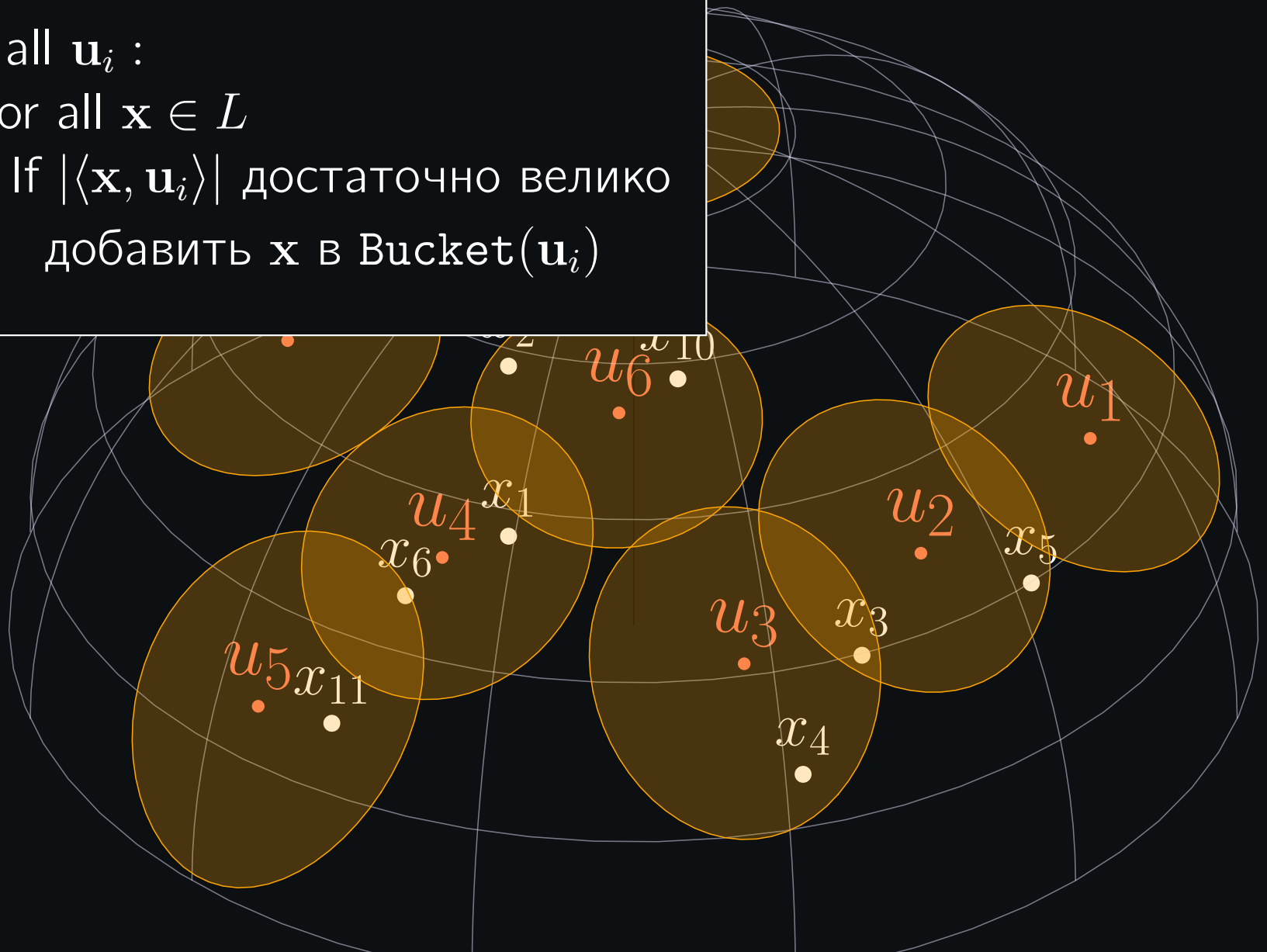
Locality-sensitive filtering (Фильтрация по расстоянию)[BGJ15, BDGL16]

u_i – центры "корзин"

$x_i \in L$

Locality-sensitive filtering (Фильтрация по расстоянию)[BGJ15, BDGL16]

For all $\mathbf{u}_i$:
  For all $\mathbf{x} \in L$
    If $|\langle \mathbf{x}, \mathbf{u}_i \rangle|$ достаточно велико
      добавить $\mathbf{x}$ в $\texttt{Bucket}(\mathbf{u}_i)$



Locality-sensitive filtering (Фильтрация по расстоянию)[BGJ15, BDGL16]

For all $\mathbf{u}_i$:
  For all $\mathbf{x} \in L$
    If $|\langle \mathbf{x}, \mathbf{u}_i \rangle|$ достаточно велико
      добавить $\mathbf{x}$ в $\mathtt{Bucket}(\mathbf{u}_i)$

For all $\mathbf{u}_i$:
  For all $(\mathbf{x}, \mathbf{x}') \in \mathtt{Bucket}(\mathbf{u}_i)$
    Проверить если $||\mathbf{x} \pm \mathbf{x}'||$ – мало

Locality-sensitive filtering (Фильтрация по расстоянию)[BGJ15, BDGL16]

For all $\mathbf{u}_i$:
  For all $\mathbf{x} \in L$
    If $|\langle \mathbf{x}, \mathbf{u}_i \rangle|$ достаточно велико
      добавить $\mathbf{x}$ в $\texttt{Bucket}(\mathbf{u}_i)$

For all $\mathbf{u}_i$:
  For all $(\mathbf{x}, \mathbf{x}') \in \texttt{Bucket}(\mathbf{u}_i)$
    Проверить если $||\mathbf{x} \pm \mathbf{x}'||$ – мало

Для $2^{(0.142+o(1))n}$ $\mathbf{u}_i$:

$$T = 2^{(0.349+o(1))n}$$

Если $\mathbf{u}_i$ имеют особую форму

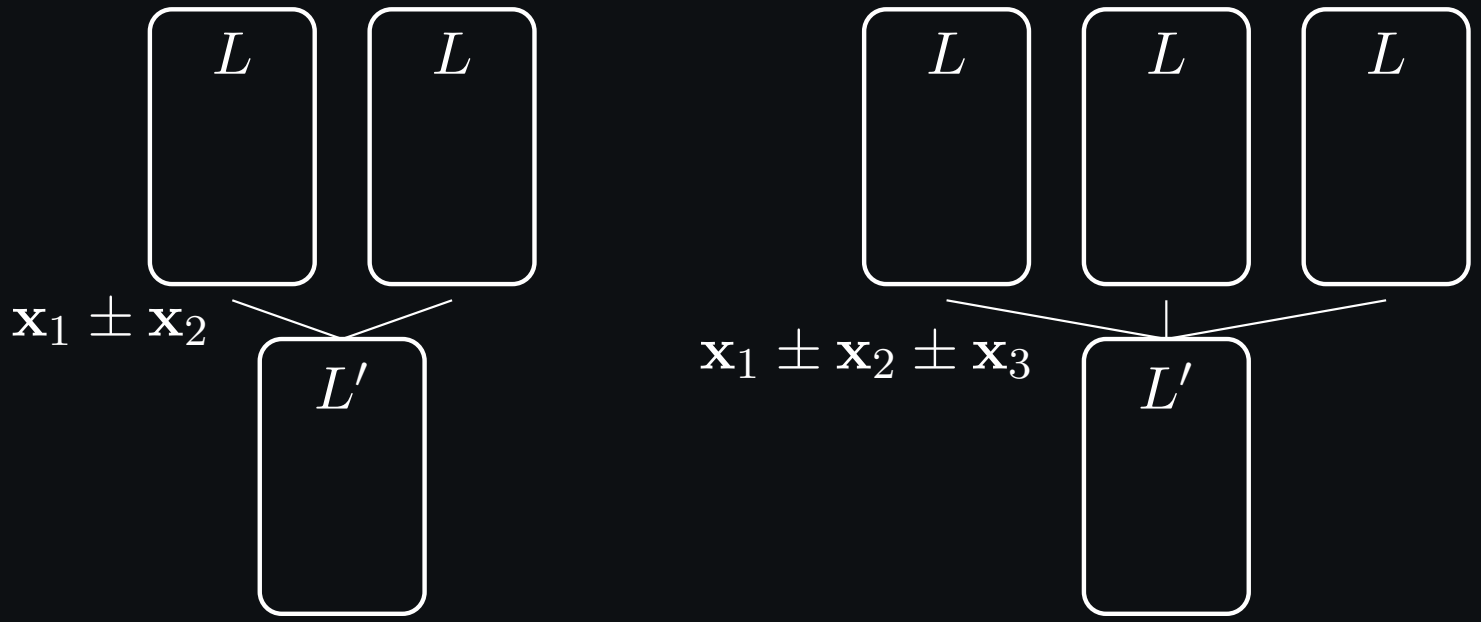
$$T = 2^{(0.292+o(1))n}$$

3-Просеивание [BLS16, HK17, HKL18]: уменьшение затрат памяти

Идея: насыщать пространство, пока тройки векторов не станут давать короткие суммы

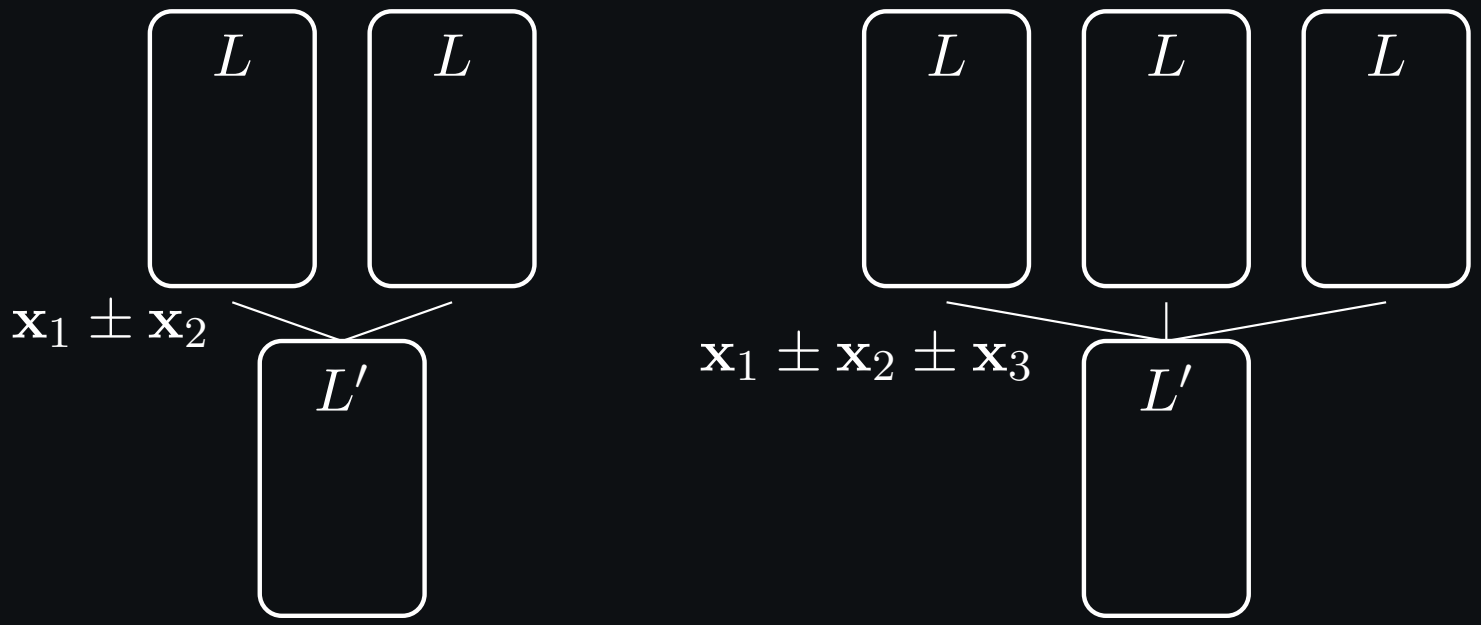
3-Просеивание [BLS16, HK17, HKL18]: уменьшение затрат памяти

Идея: насыщать пространство, пока тройки векторов не станут давать короткие суммы



3-Просеивание [BLS16, HK17, HKL18]: уменьшение затрат памяти

Идея: насыщать пространство, пока тройки векторов не станут давать короткие суммы



- Размер списка

$$|L| = |L|^3 \cdot \Pr[\ ||\mathbf{x}_1 + \mathbf{x}_2 + \mathbf{x}_3|| \text{ small }\ ]$$

- Для 3-Просеивания имеем

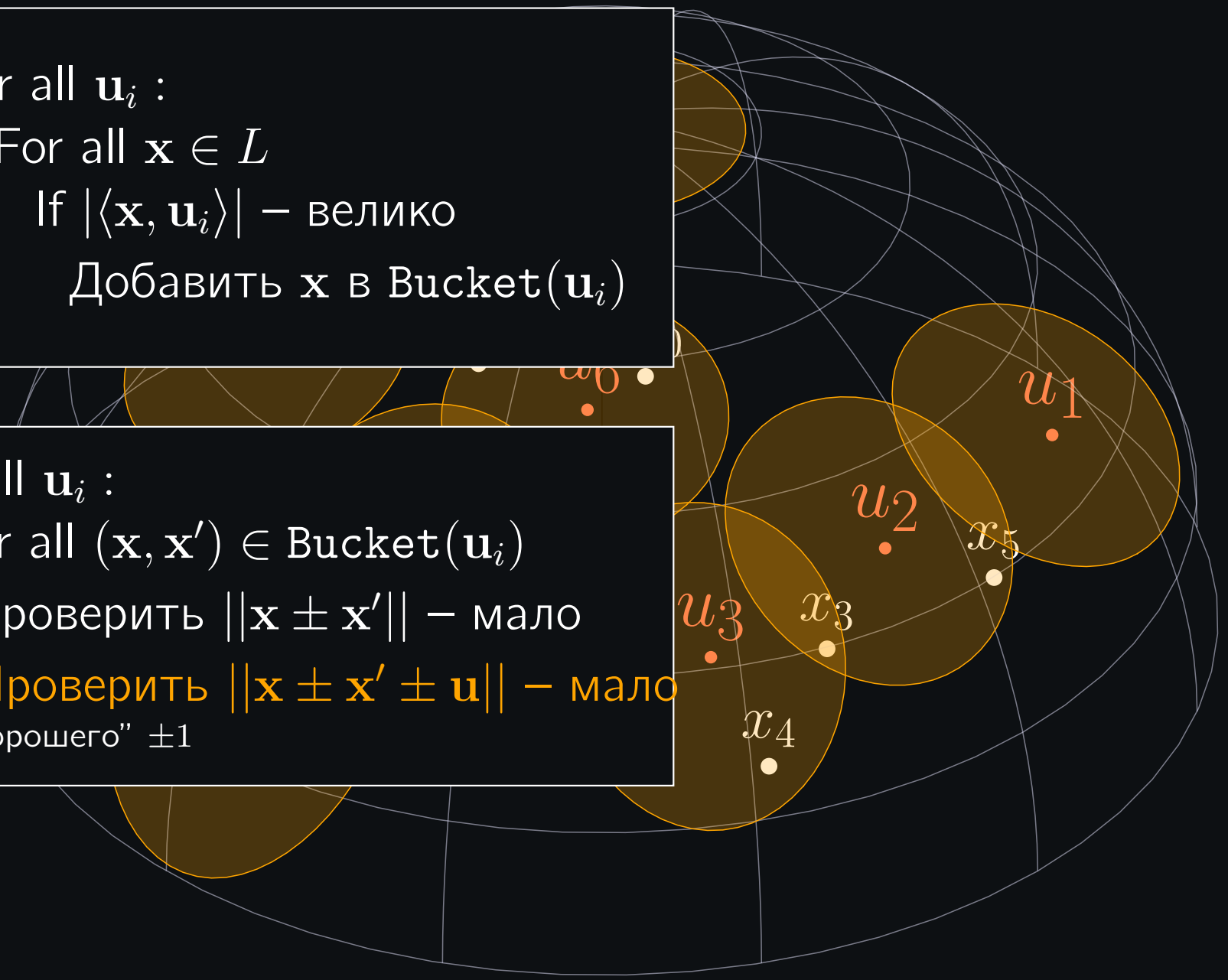
$$|L| = 2^{0.179n}, \quad \text{cf. } 2^{0.208n} \text{ for 2-Sieve}$$
$$T = 2^{0.359n}, \quad \text{cf. } 2^{0.292n} \text{ for 2-Sieve}$$

Улучшенное 3-просеивание с хэшированием

For all $\mathbf{u}_i$:
  For all $\mathbf{x} \in L$
    If $|\langle \mathbf{x}, \mathbf{u}_i \rangle|$ – велико
      Добавить $\mathbf{x}$ в $\texttt{Bucket}(\mathbf{u}_i)$

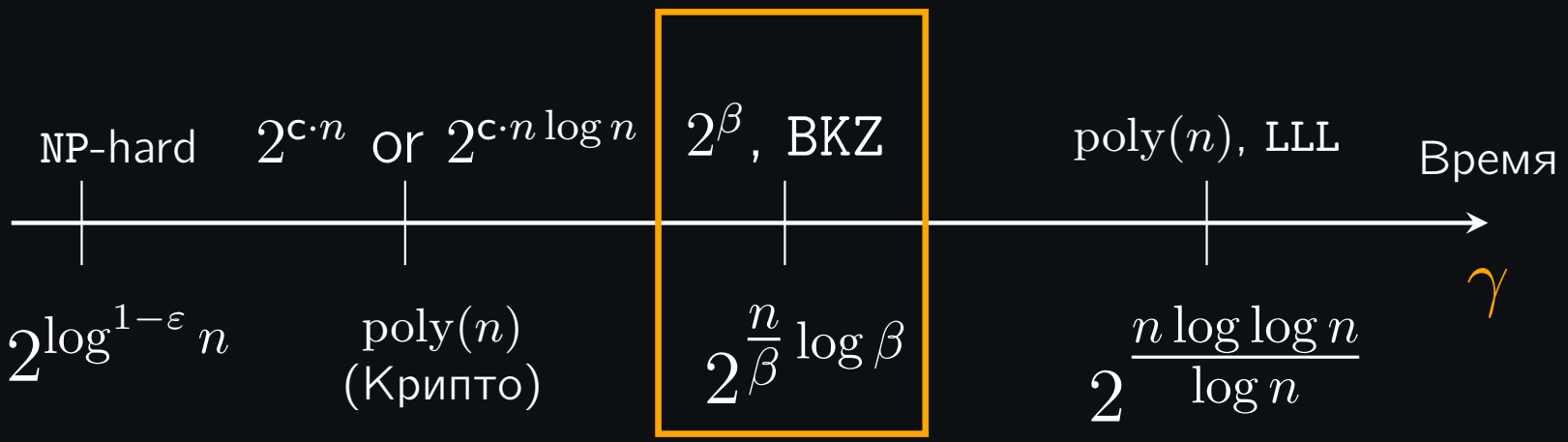
For all $\mathbf{u}_i$:
  For all $(\mathbf{x}, \mathbf{x}') \in \texttt{Bucket}(\mathbf{u}_i)$
    Проверить $||\mathbf{x} \pm \mathbf{x}'||$ – мало
    Проверить $||\mathbf{x} \pm \mathbf{x}' \pm \mathbf{u}||$ – мало

для “хорошего” ± 1



Часть III

Решение челленджей SVP алгоритмами просеивания [1]



[1] Основано на статье M. R. Albrecht, L. Ducas, G.Herold, E. Kirshanova, E. W. Postlethwaite, M. Stevens, EuroCrypt'19

Реальные SVP задачи

- TU Darmstadt предлагает решить задачу 1.05-SVP
- Для “случайных” решёток вида:

$$B = \begin{pmatrix} p & x_1 & \dots & x_{n-1} \\ 0 & 1 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & 1 \end{pmatrix}$$

- Рекорд (до 2019) принадлежал Teruya et al.: решение 1.05-SVP в решётке ранга 150-dim алгоритмом перечисления за 2^{22} core-часов

Решения 1.05-SVP с помощью алгоритма G6K[2]

SVP dim	γ	Sieve Wall time	Total CPU time	Memory usage
155	1.00803	$14d\ 16h$	$1056d$	246 GiB
153	1.02102	$11d\ 15h$	$911d$	139 GiB
151	1.04411	$11d\ 19h$	$457.5d$	160 GiB
149	0.98506	$60h\ 7m$	$4.66kh$	59 GiB
147	1.03863	$123h\ 29m$	$4.79kh$	67.0 GiB
145	1.04267	$39h\ 3m$	$1496h$	37.7 GiB

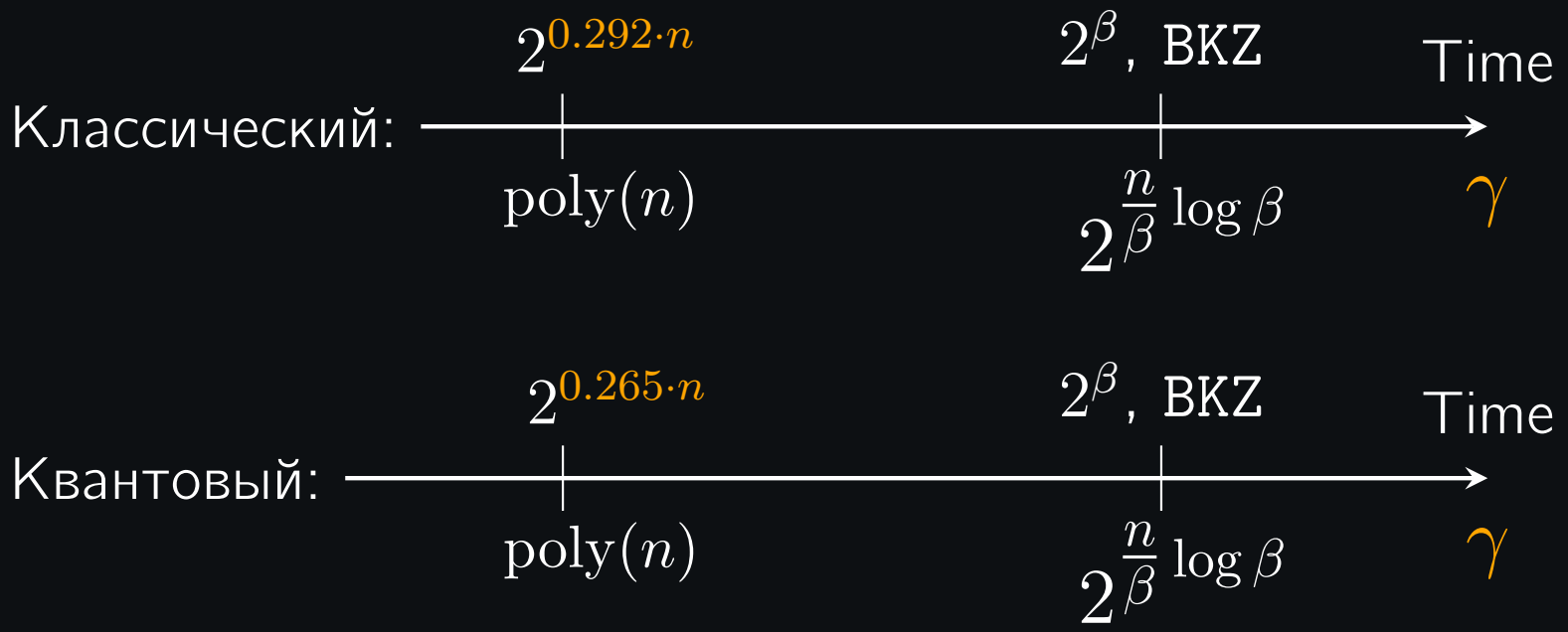
[2] https://github.com/fplll/g6k

Часть IV

Квантовые ускорения и открытые вопросы

Квантовые ускорения для SVP, [Laarhoven16, KMPR'19]

I. Незначительные ускорения для обычных решёток



- Для алгоритма с $0.265n$
- Существуют компромиссы время/память $T = 2^{0.2989 \cdot n}, M = 2^{0.1395n}$
- Существует алгоритм для "параллельного" квантового компьютера, решающий SVP с $T = 2^{0.1037 \cdot n}$, $M = 2^{0.2075 \cdot n}$

Открытые вопросы

1. Улучшенные алгоритмы для “идеальных” решёток
2. Практическая реализация асимптотически быстрых алгоритмов, распределенная версия просеивания
3. Алгоритмы для не ℓ_2 нормы (например, ℓ_∞)

Открытые вопросы

1. Улучшенные алгоритмы для “идеальных” решёток
2. Практическая реализация асимптотически быстрых алгоритмов, распределенная версия просеивания
3. Алгоритмы для не ℓ_2 нормы (например, ℓ_∞)

?