Fast Implementation of the Node2Vec Algorithm

Пластова Полина Игоревна, Морковкин Андрей Сергеевич, Соколов Андрей Павлович

Что такое Node2Vec?

Постановка задачи

Как работает Node2Vec?

Оптимизация 1. Блочная генерация случайных чисел

Оптимизация 2. Разбиение циклов

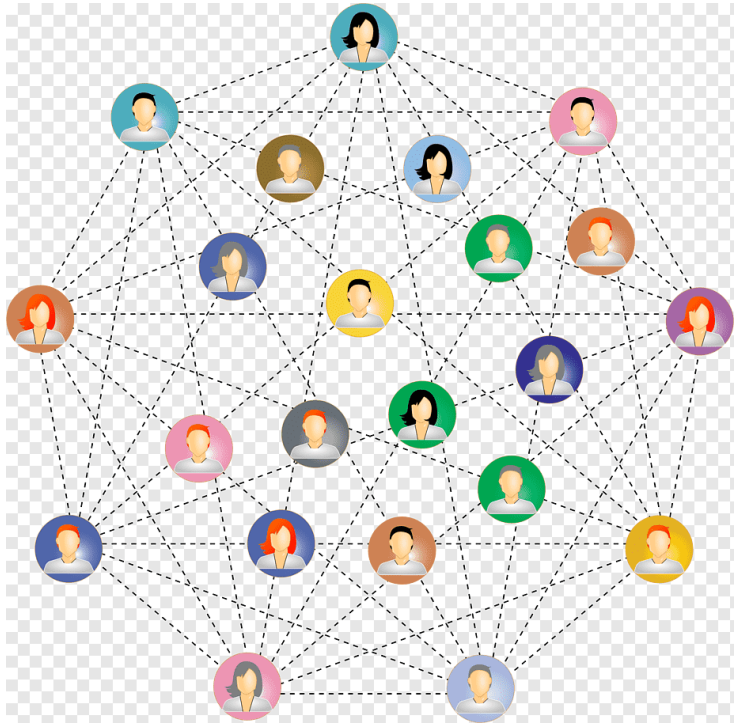
Оптимизация 3. Выравнивание по кэш-линиям

Контроль точности

Результаты

Обучение графовых представлений

Обучение графовых представлений позволяет сопоставить элементам графа некоторые представления (вектора), которые предположительно отражают структуру и свойства графа.

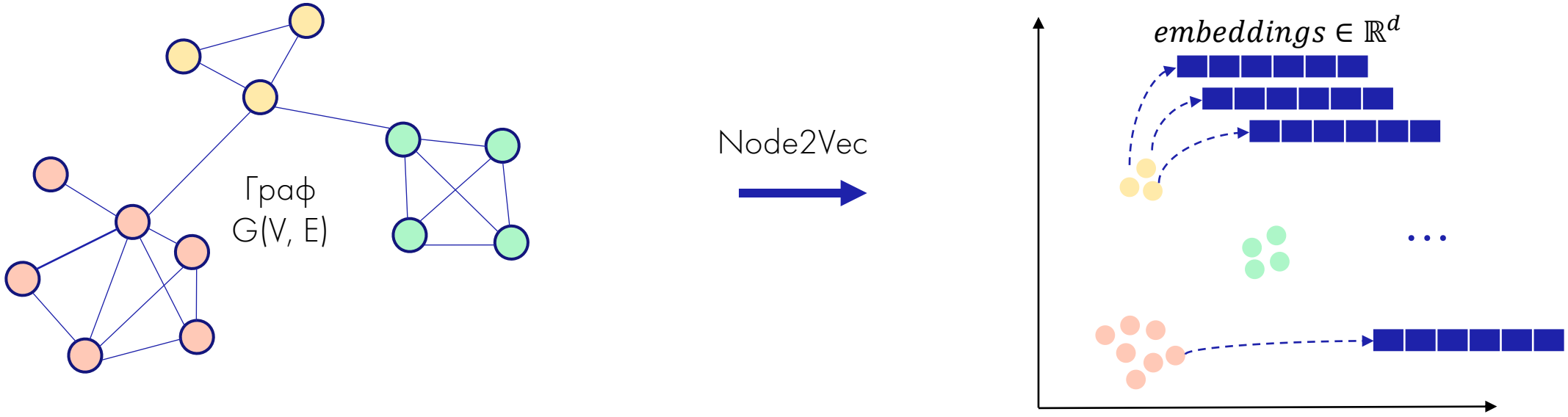


Приложения:

- ✓ классификация вершин
- ✓ link prediction
- ✓ детекция аномалий
- ✓ рекомендательные и поисковые системы

Что такое Node2Vec?

Node2Vec – алгоритм обучения представлений вершин графа, основанный на случайных блужданиях и модели Skip-gram.



Что такое Node2Vec?

Мотивация и постановка задачи

Как работает Node2Vec?

Оптимизация 1. Блочная генерация случайных чисел

Оптимизация 2. Разбиение циклов

Оптимизация 3. Выравнивание по кэш-линиям

Контроль точности

Результаты

Мотивация и постановка задачи

- Мотивация:
  Рекомендательные и поисковые системы часто используют представления графов. Поэтому производительность алгоритмов обучения представлений важна.

- Наша задача:
  Ускорить базовую C++ реализацию Node2Vec алгоритма от Stanford Network Analysis Project (SNAP).

node2vec: Scalable Feature Learning for Networks. A. Grover, J. Leskovec. ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD), 2016.

Что такое Node2Vec?

Мотивация и постановка задачи

Как работает Node2Vec?

Оптимизация 1. Блочная генерация случайных чисел

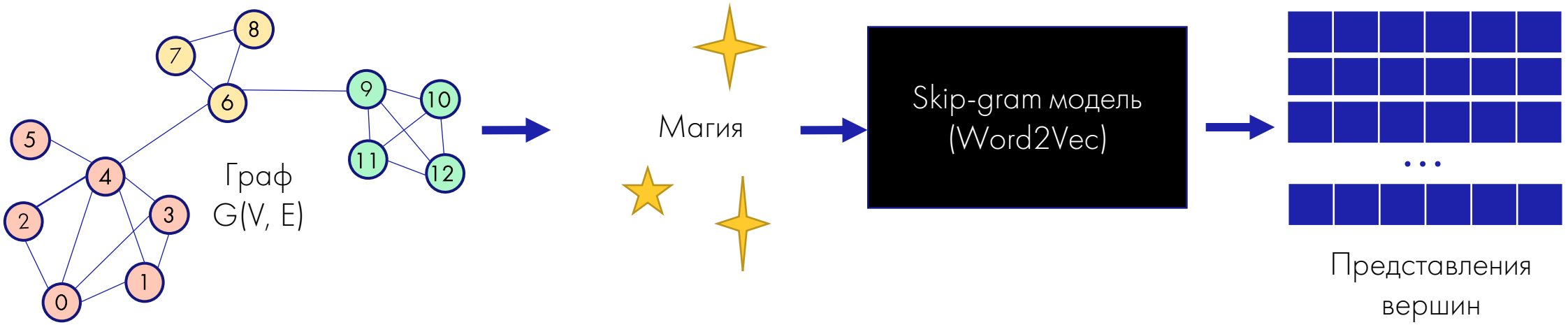
Оптимизация 2. Разбиение циклов

Оптимизация 3. Выравнивание по кэш-линиям

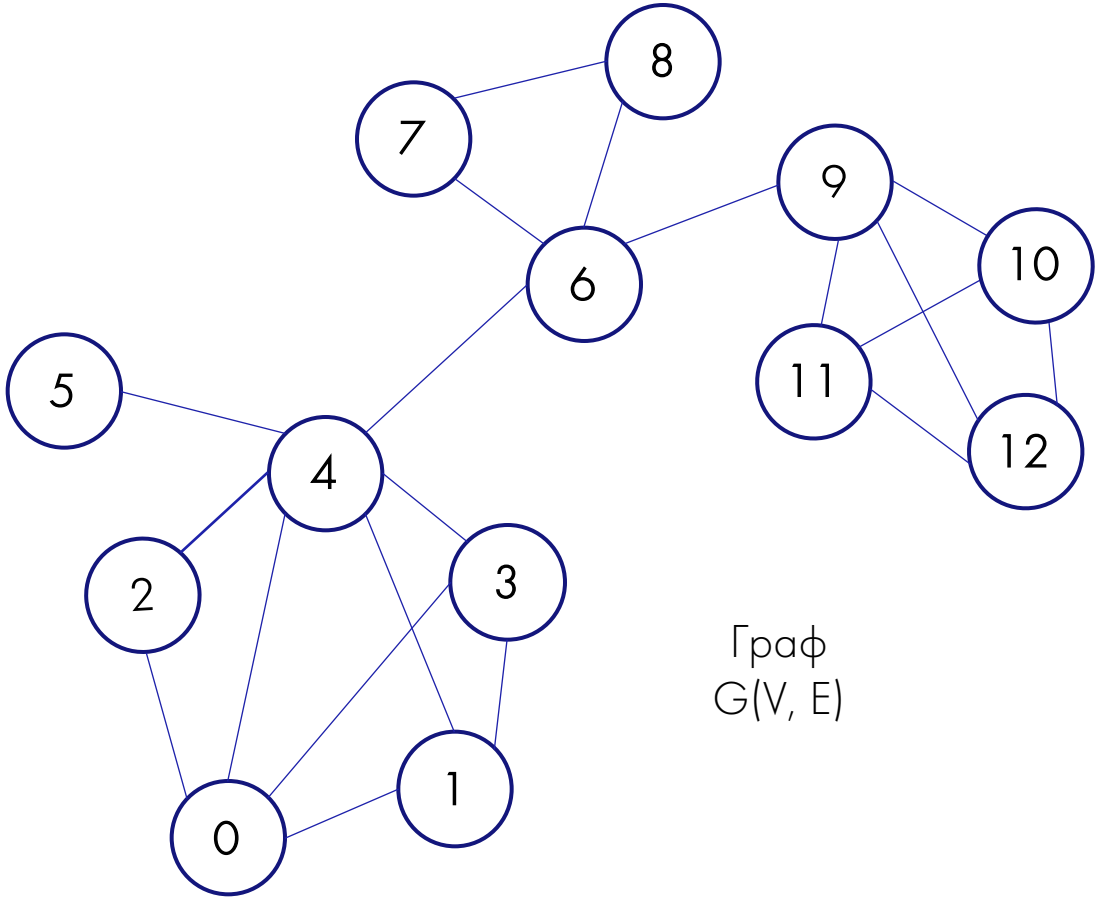
Контроль точности

Результаты

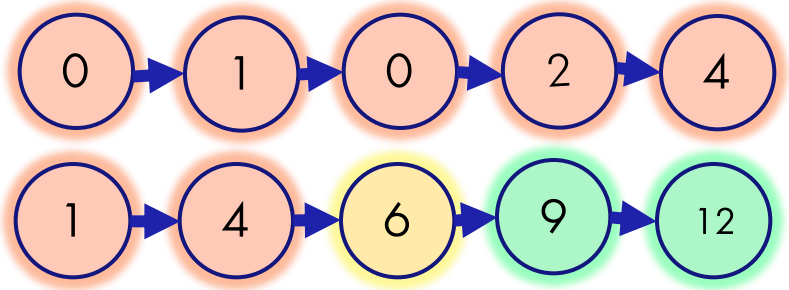
Как получить представления вершин графа?



Как работает Node2Vec?

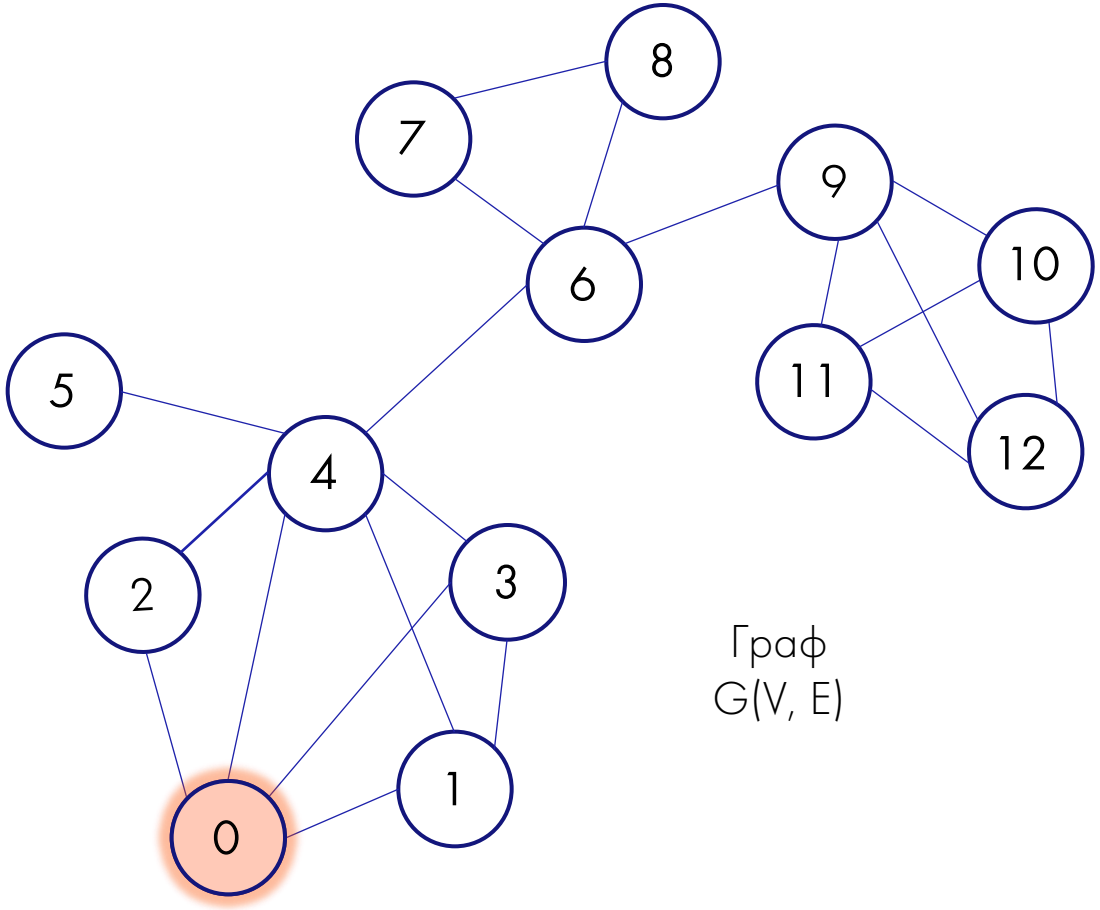


Граф
G(V, E)



Как работает Node2Vec?

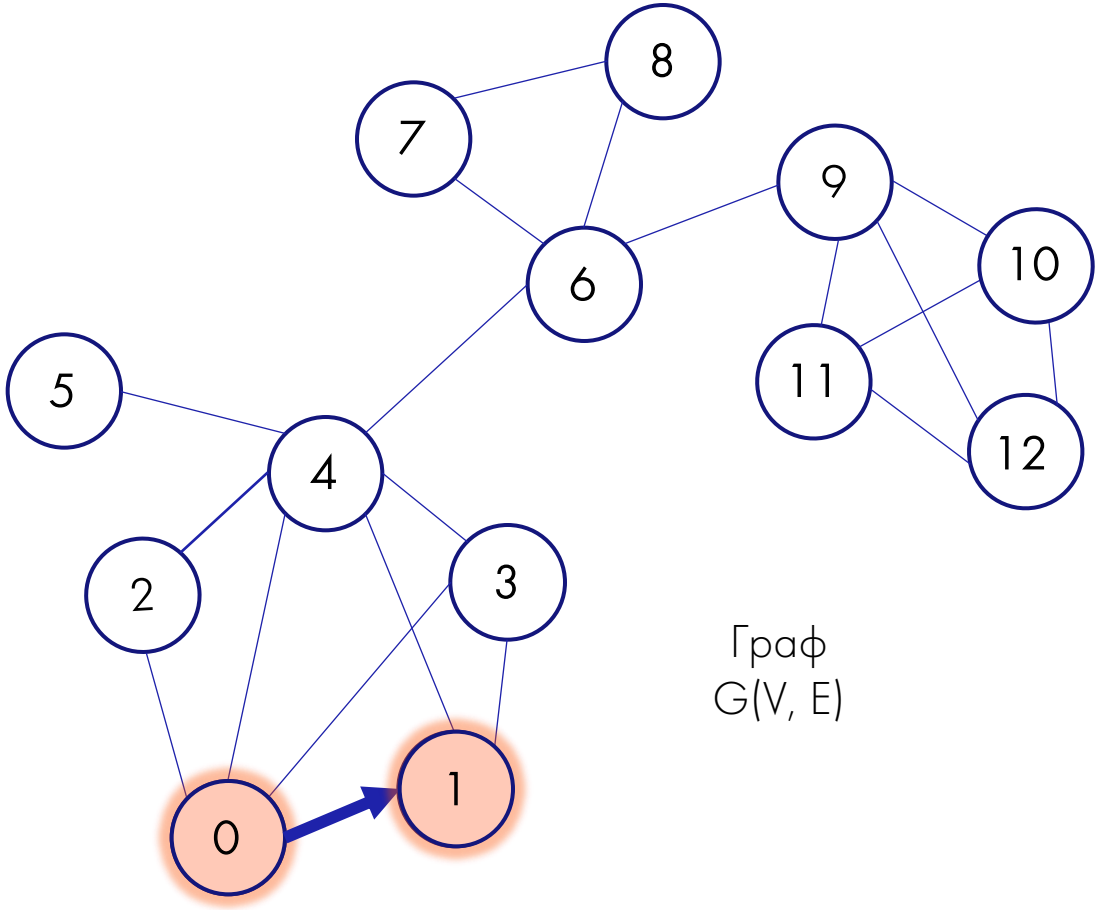
Шаг 1. Генерация последовательностей вершин



Граф
G(V, E)

Как работает Node2Vec?

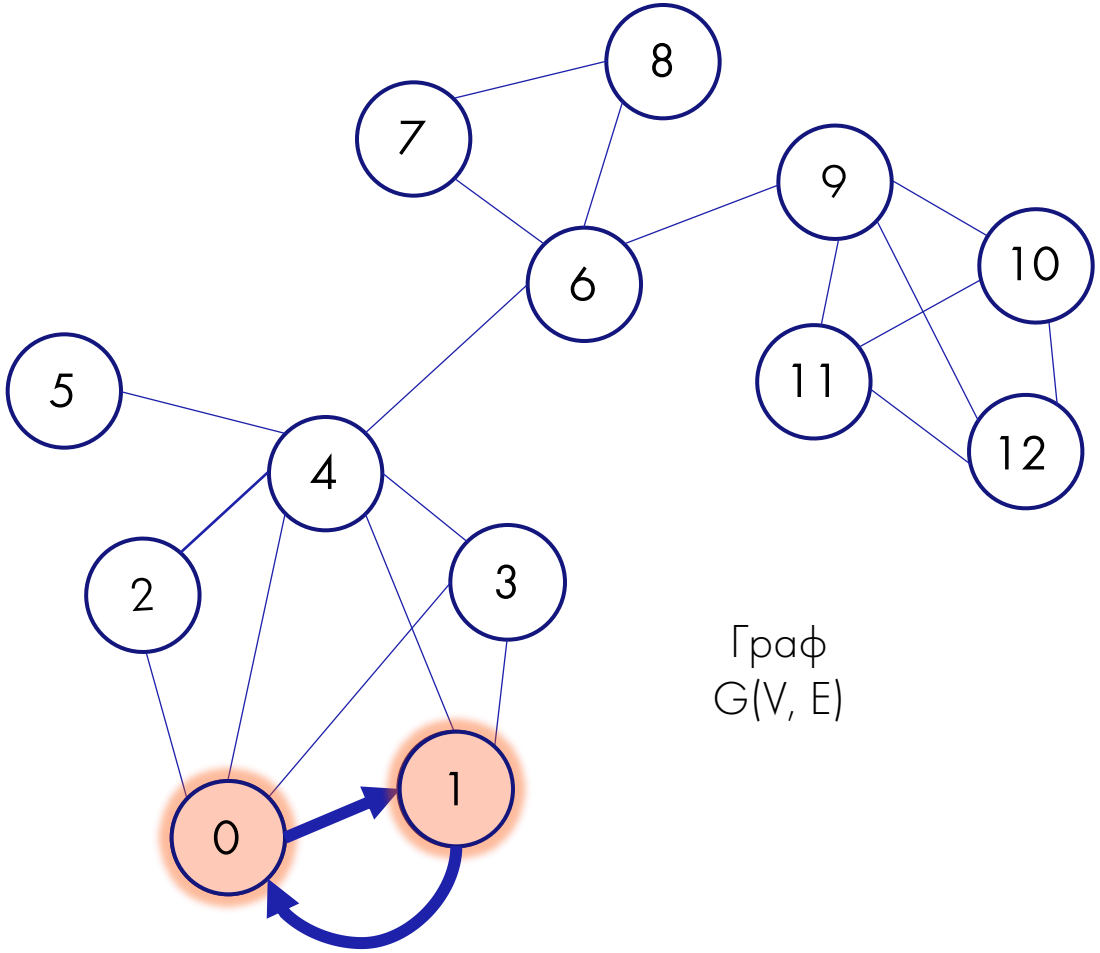
Шаг 1. Генерация последовательностей вершин



Граф
G(V, E)

Как работает Node2Vec?

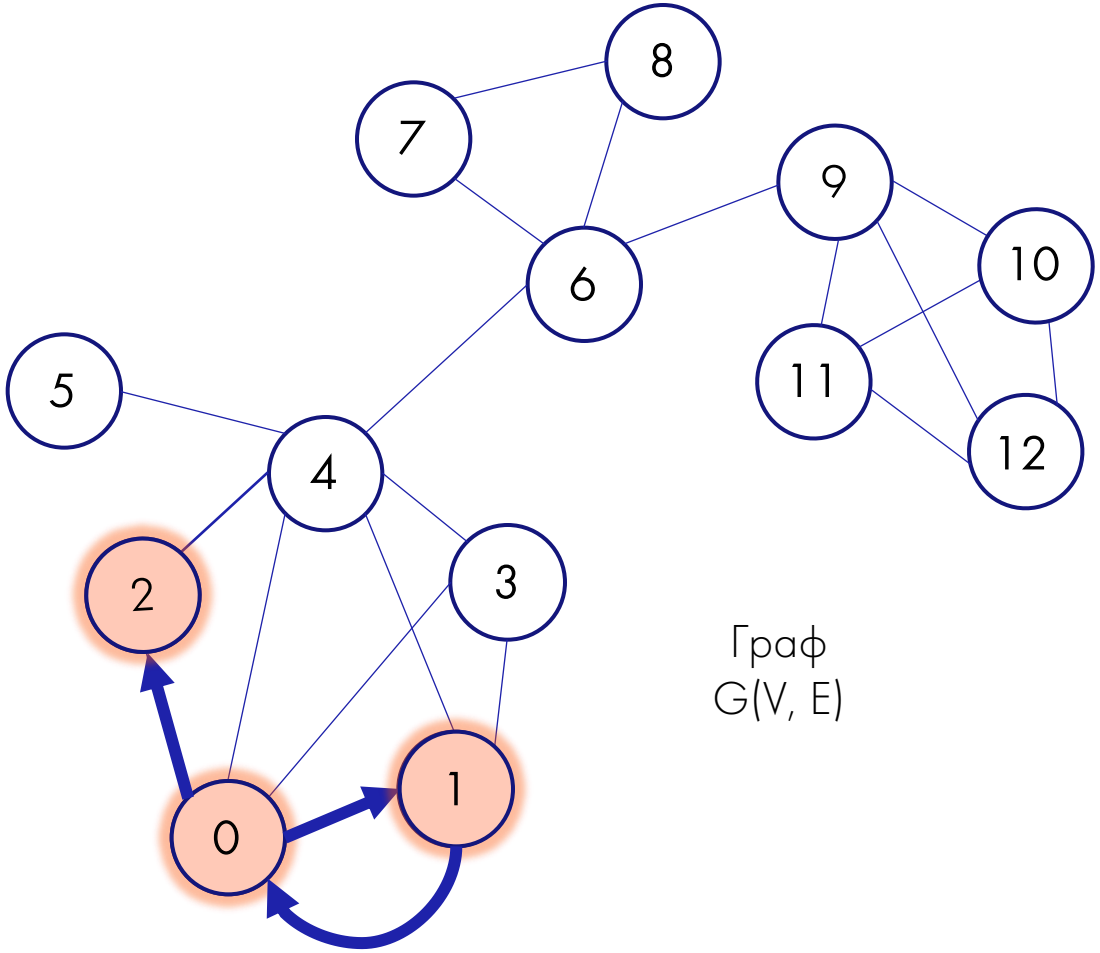
Шаг 1. Генерация последовательностей вершин



Граф
G(V, E)

Как работает Node2Vec?

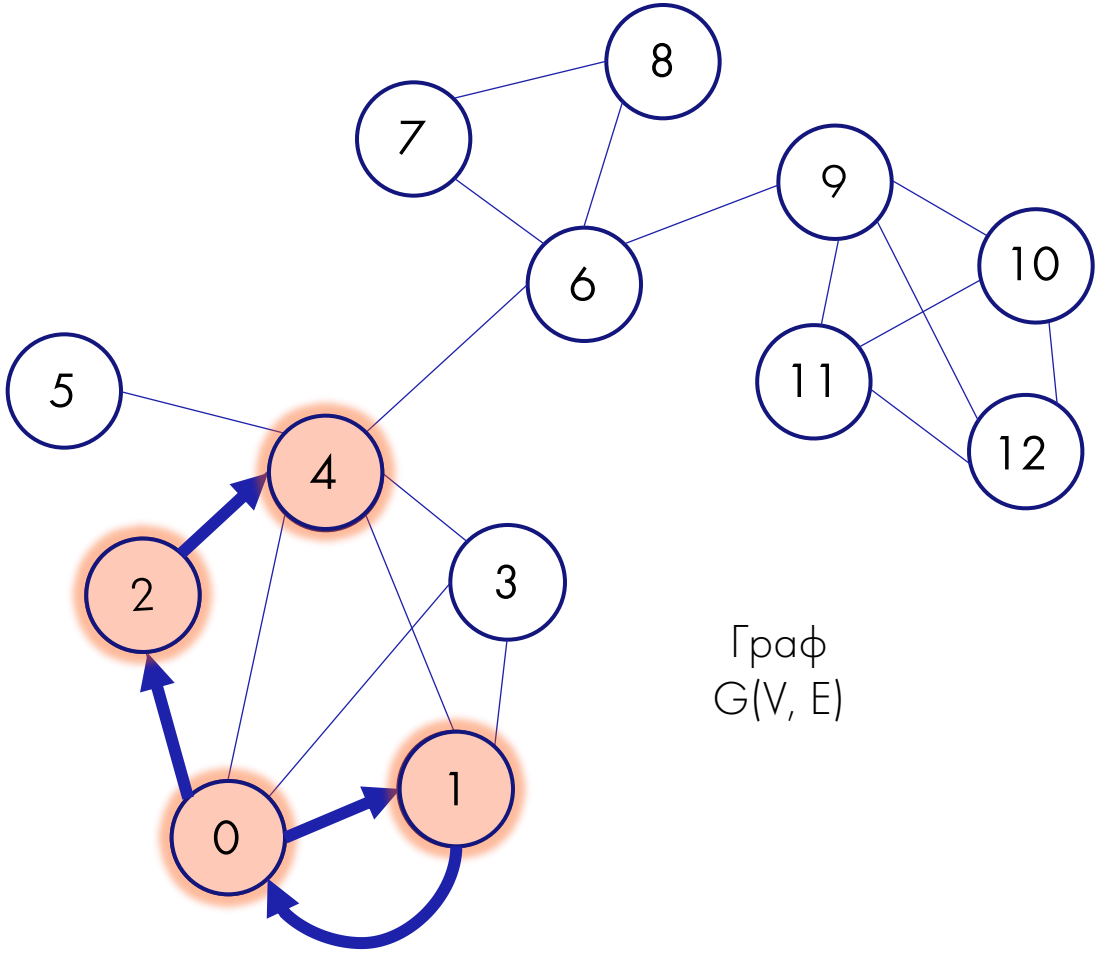
Шаг 1. Генерация последовательностей вершин



Граф
G(V, E)

Как работает Node2Vec?

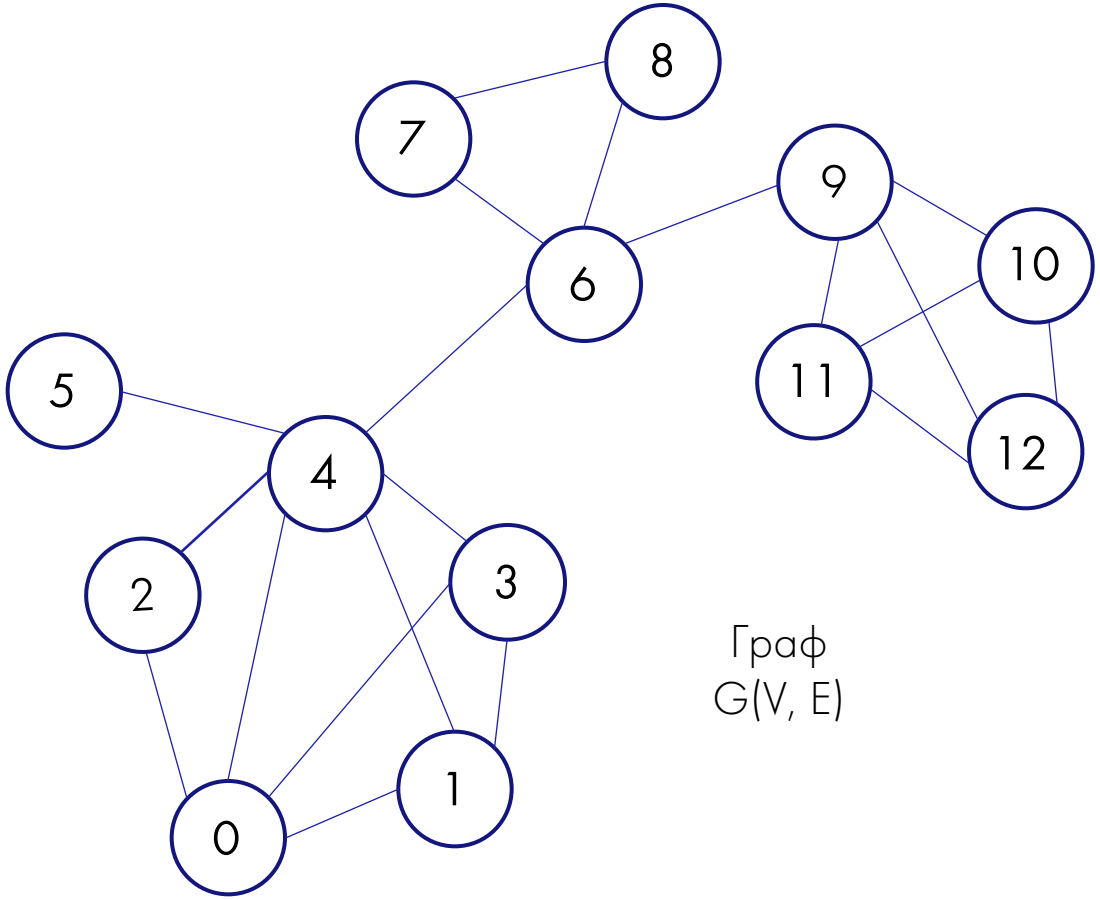
Шаг 1. Генерация последовательностей вершин



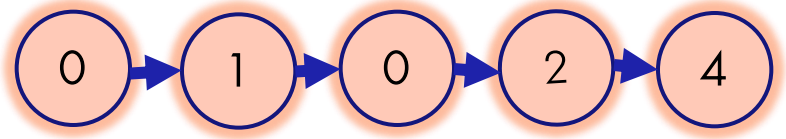
Граф
G(V, E)

Как работает Node2Vec?

Шаг 1. Генерация последовательностей вершин

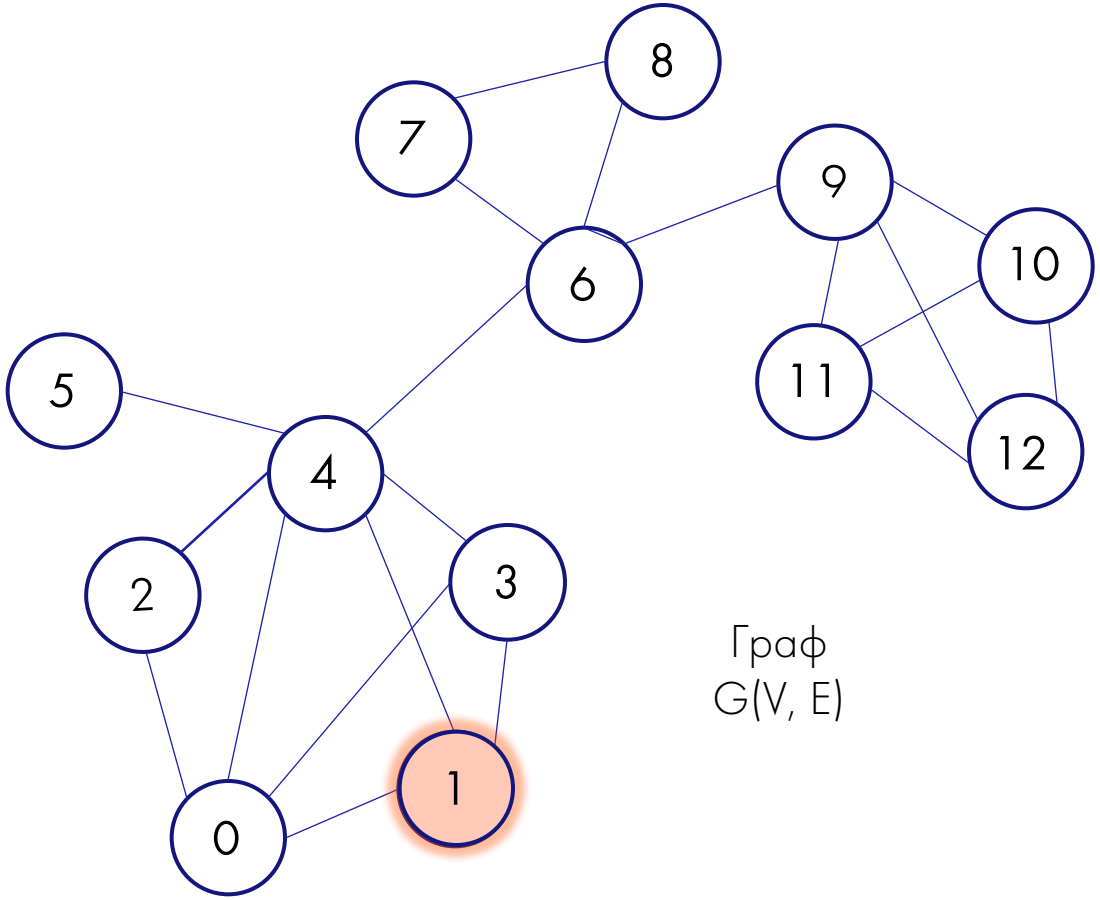


Граф
G(V, E)

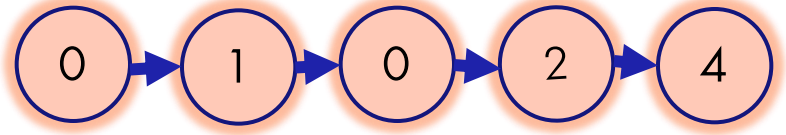


Как работает Node2Vec?

Шаг 1. Генерация последовательностей вершин

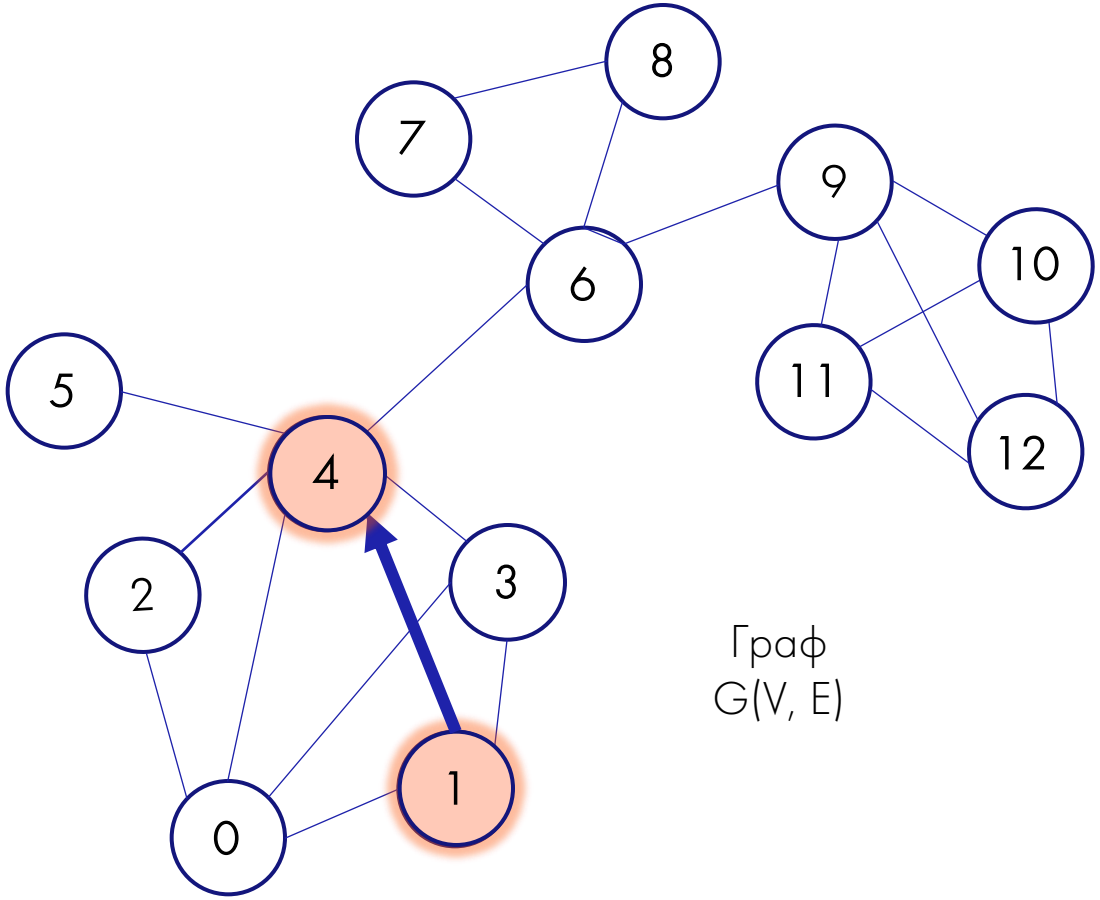


Граф
G(V, E)

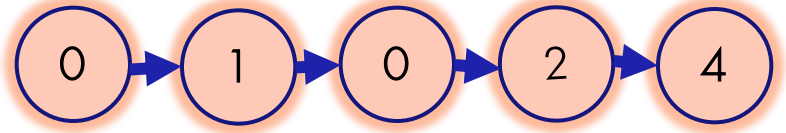


Как работает Node2Vec?

Шаг 1. Генерация последовательностей вершин

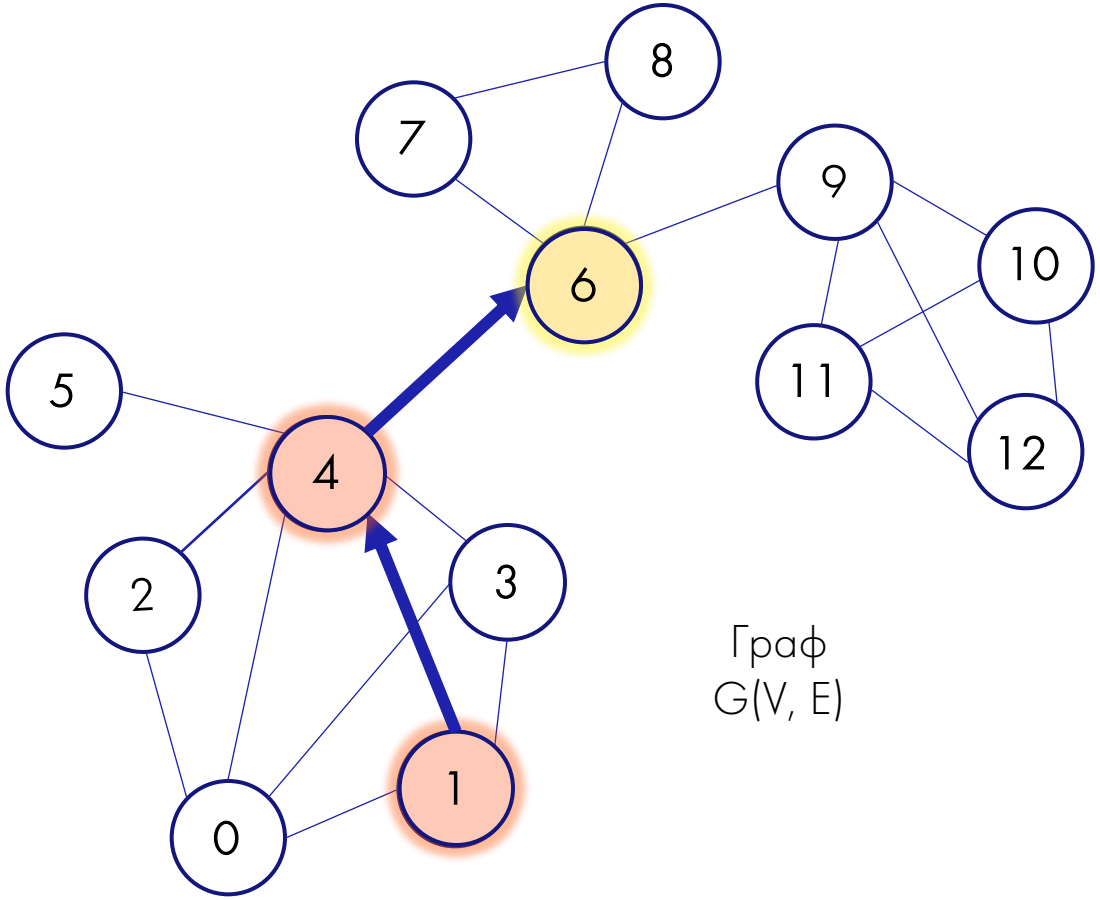


Граф
G(V, E)

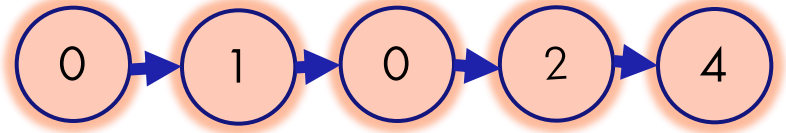


Как работает Node2Vec?

Шаг 1. Генерация последовательностей вершин

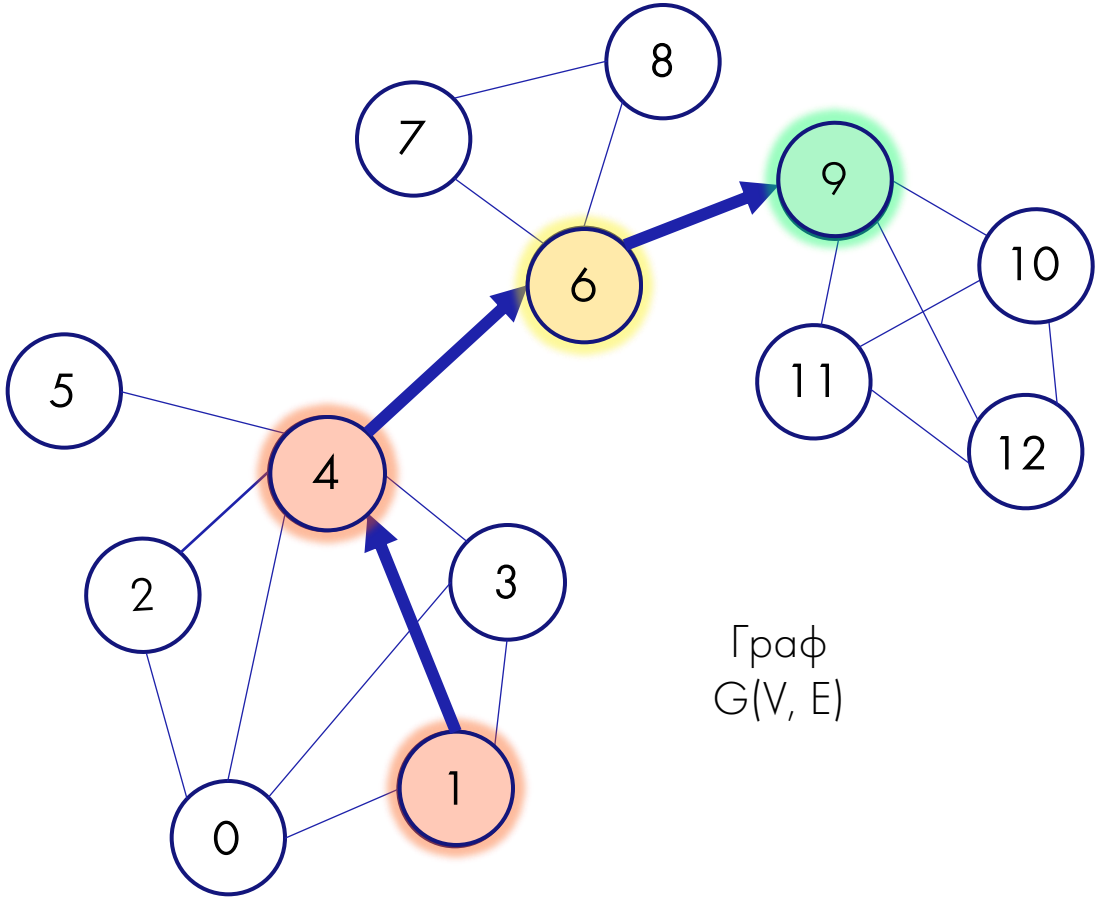


Граф
G(V, E)

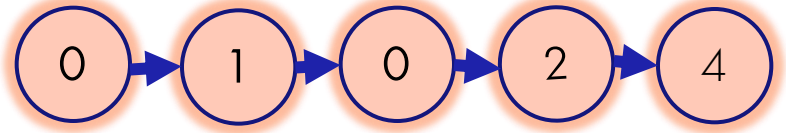


Как работает Node2Vec?

Шаг 1. Генерация последовательностей вершин

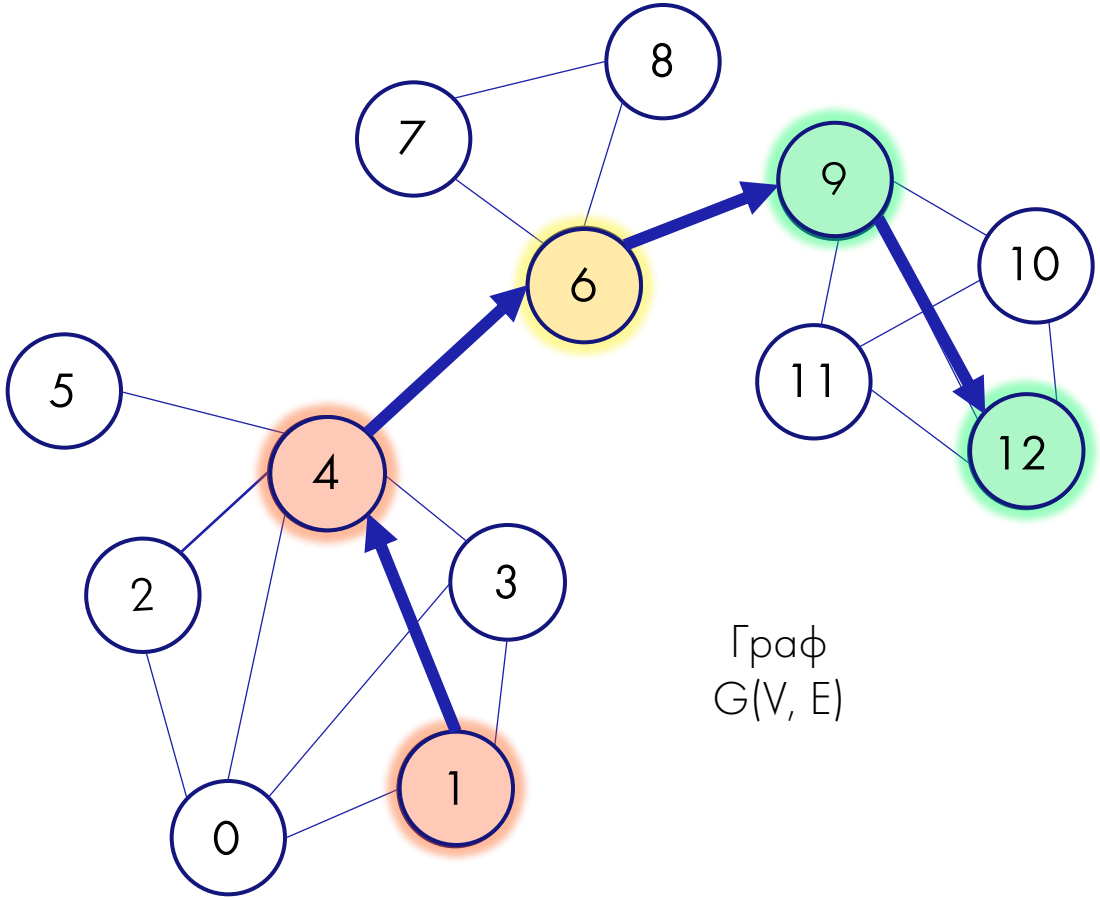


Граф
G(V, E)

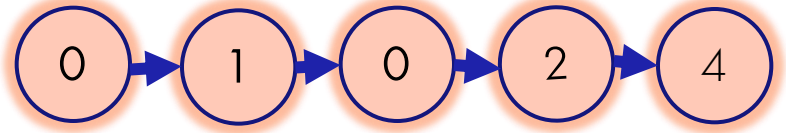


Как работает Node2Vec?

Шаг 1. Генерация последовательностей вершин

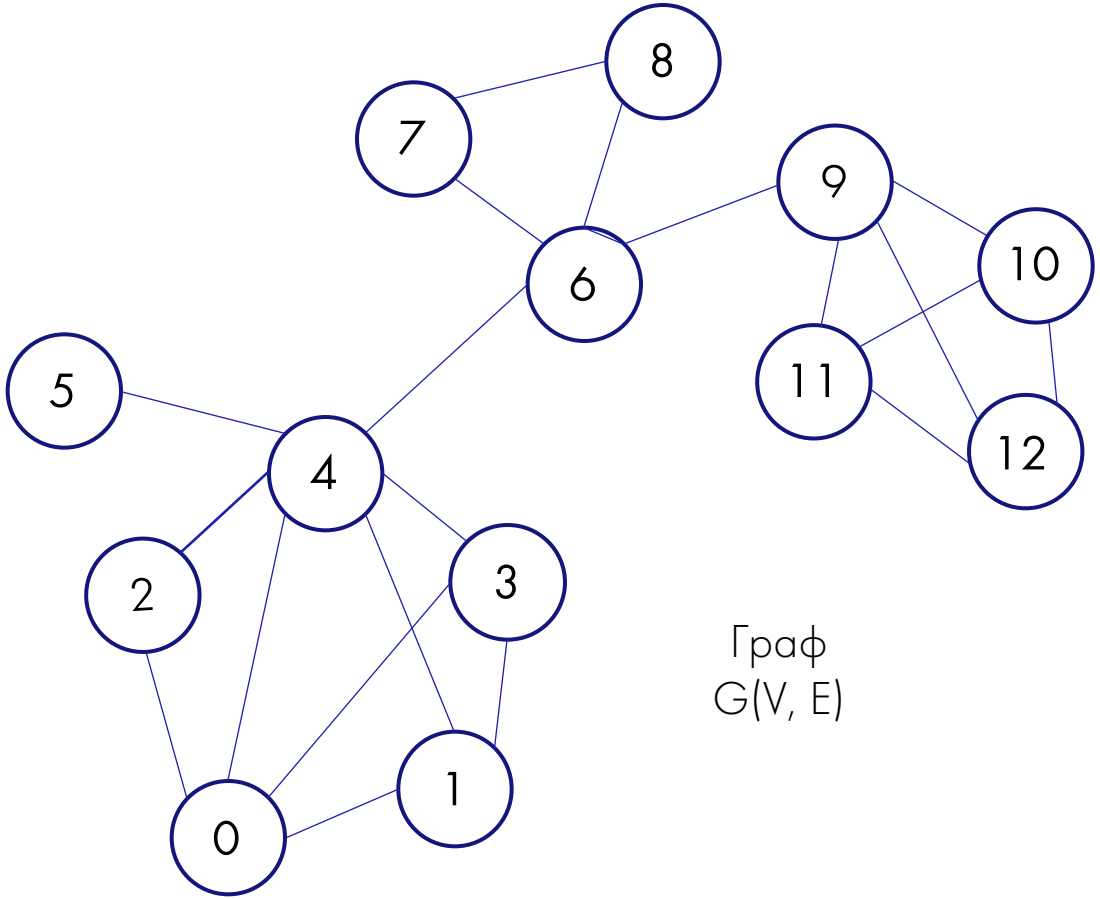


Граф
G(V, E)

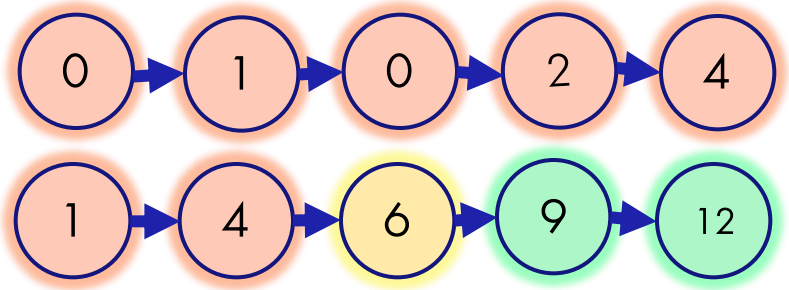


Как работает Node2Vec?

Шаг 1. Генерация последовательностей вершин



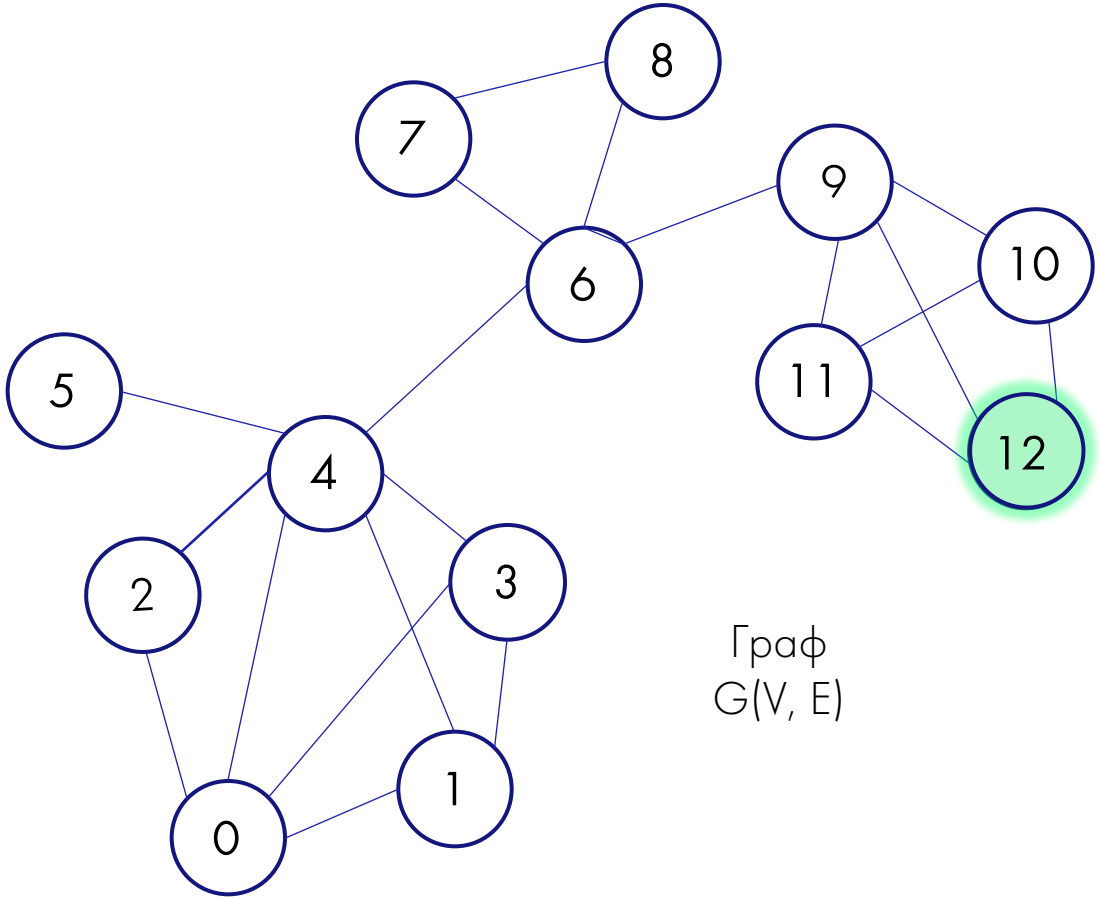
Граф
G(V, E)



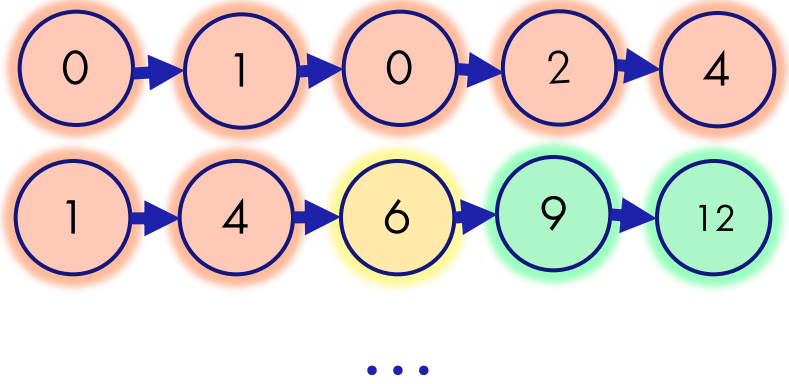
...

Как работает Node2Vec?

Шаг 1. Генерация последовательностей вершин



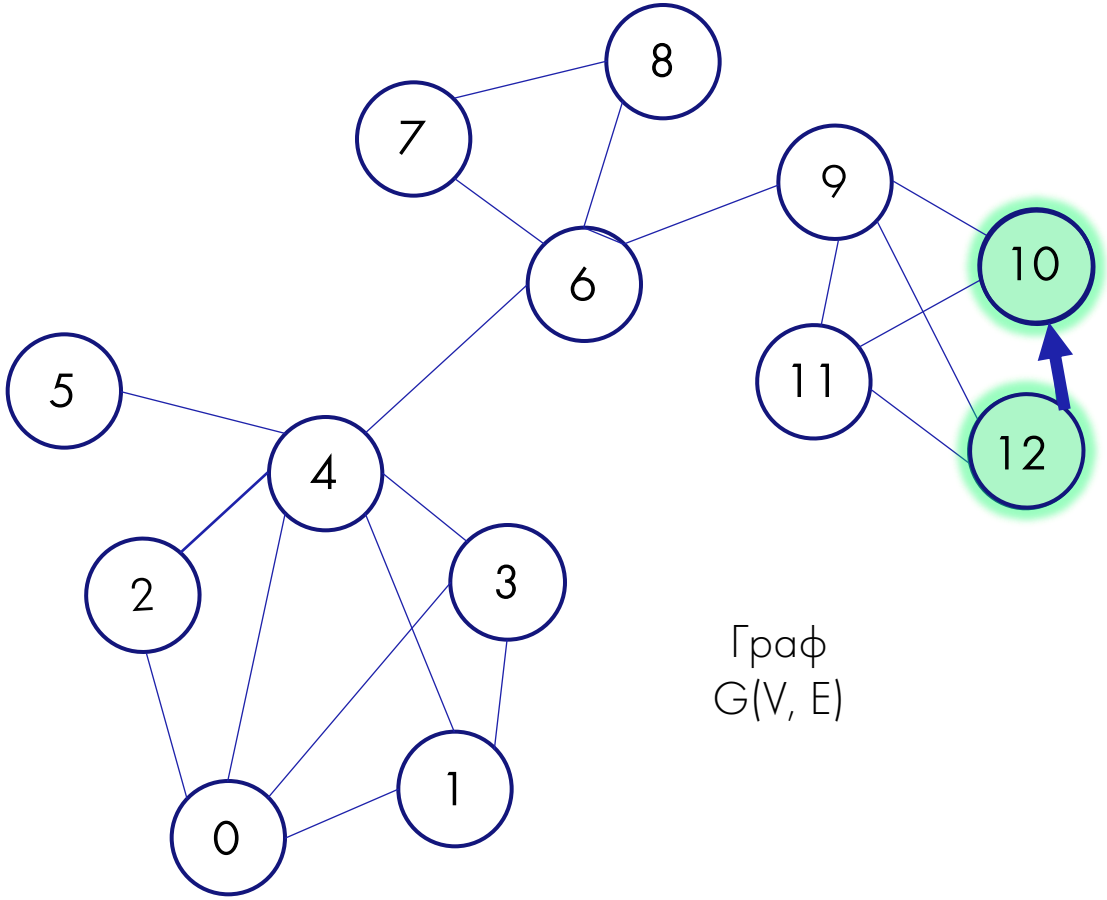
Граф
G(V, E)



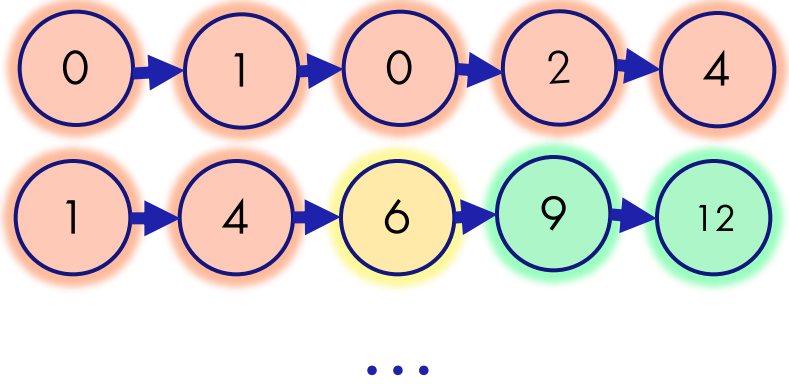
...

Как работает Node2Vec?

Шаг 1. Генерация последовательностей вершин



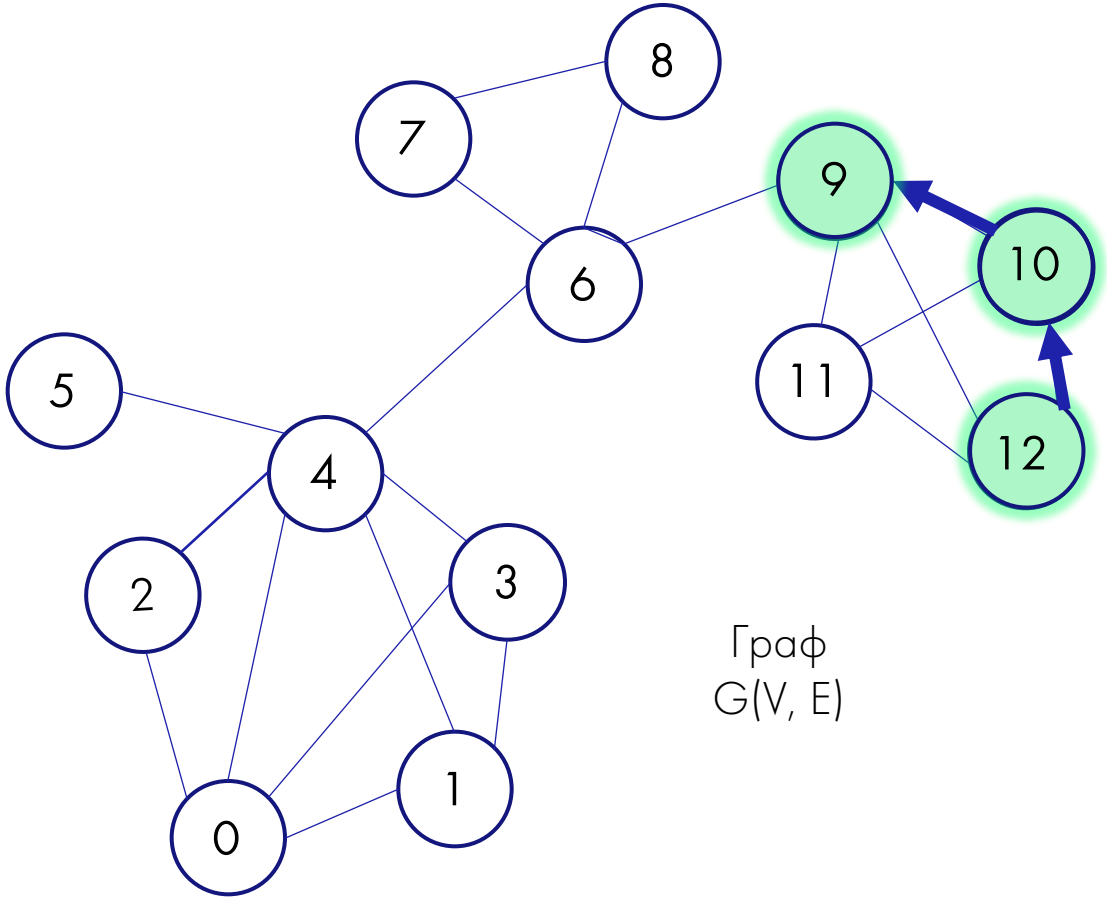
Граф
G(V, E)



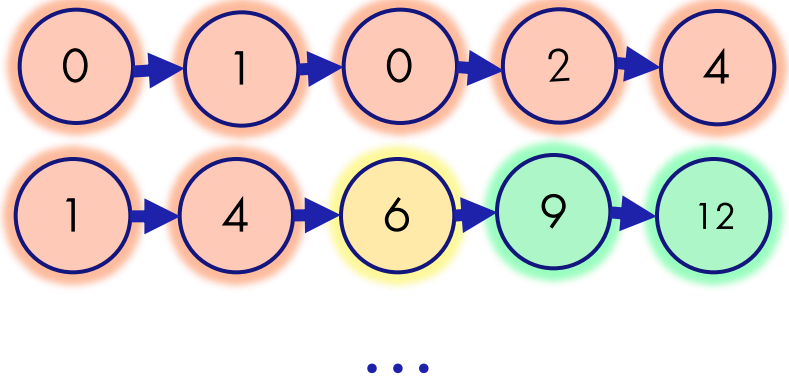
...

Как работает Node2Vec?

Шаг 1. Генерация последовательностей вершин



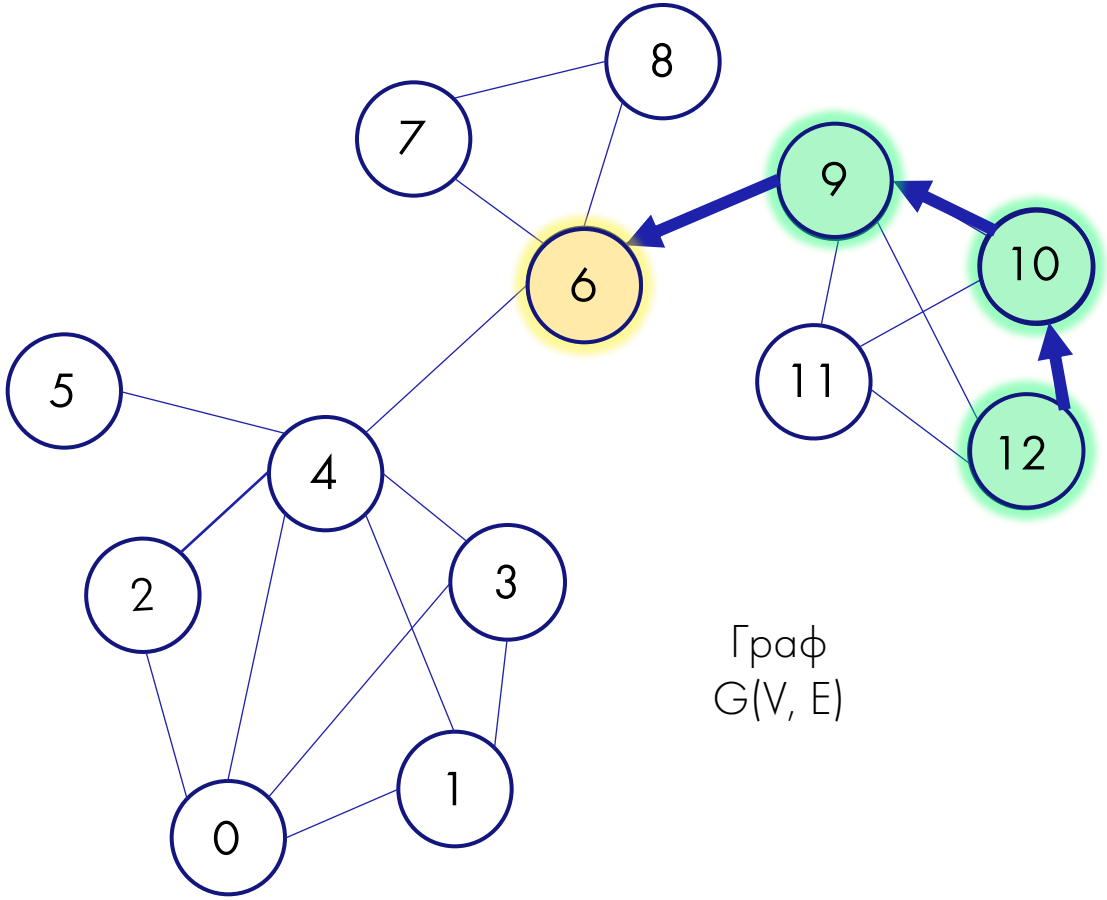
Граф
G(V, E)



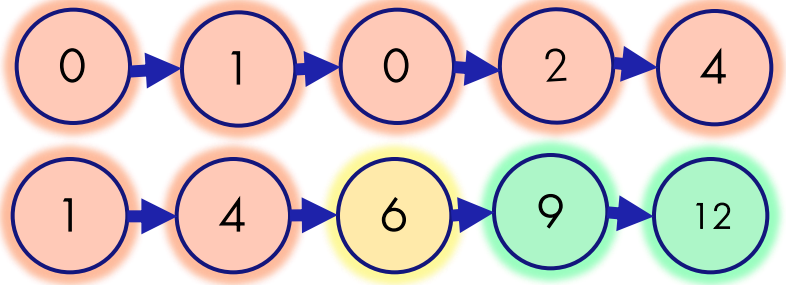
...

Как работает Node2Vec?

Шаг 1. Генерация последовательностей вершин



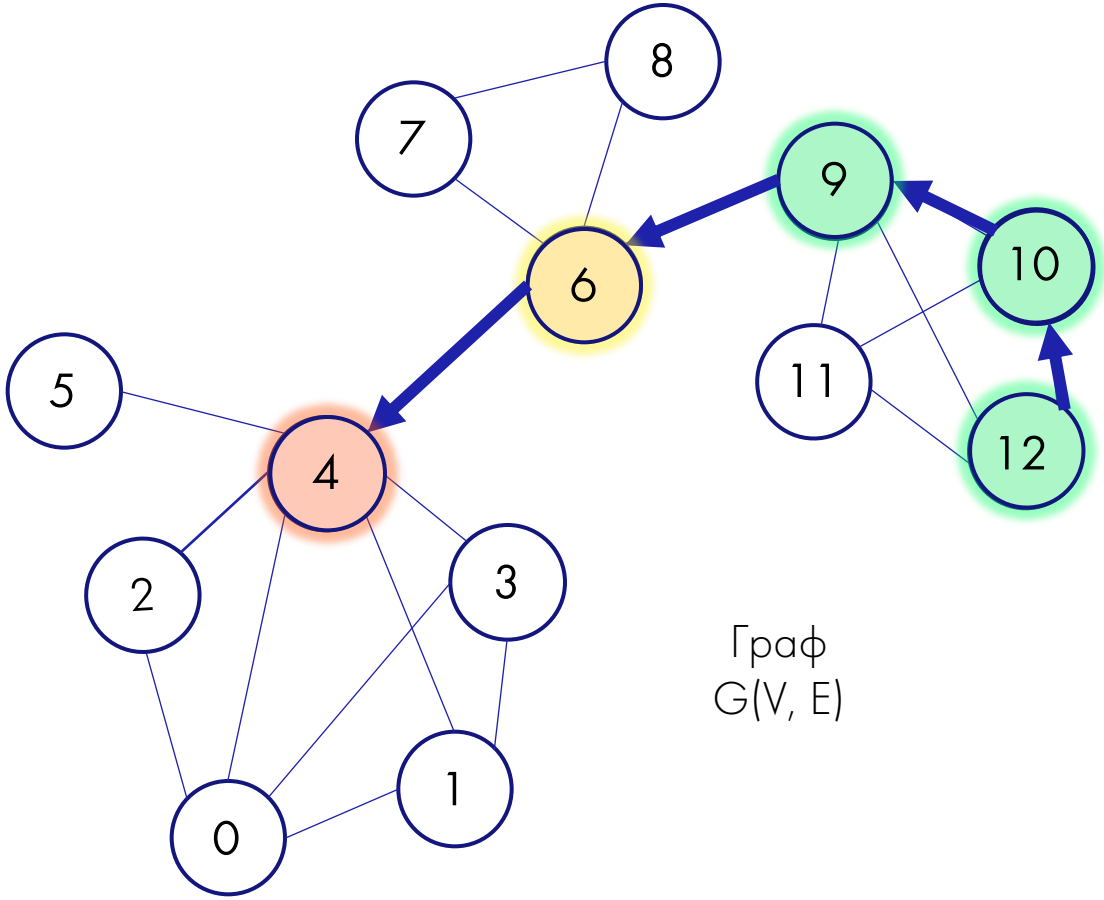
Граф
G(V, E)



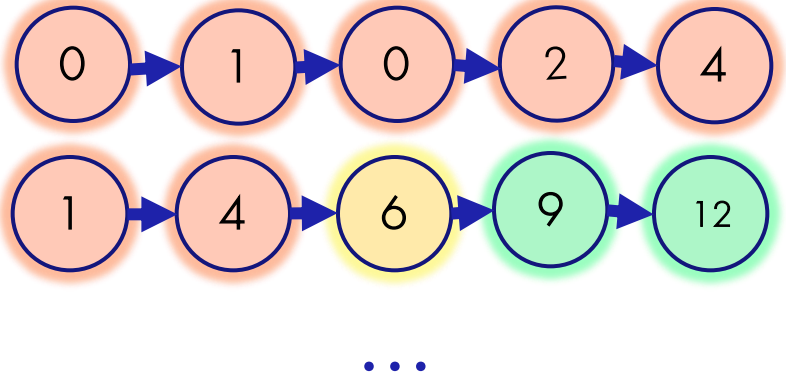
...

Как работает Node2Vec?

Шаг 1. Генерация последовательностей вершин



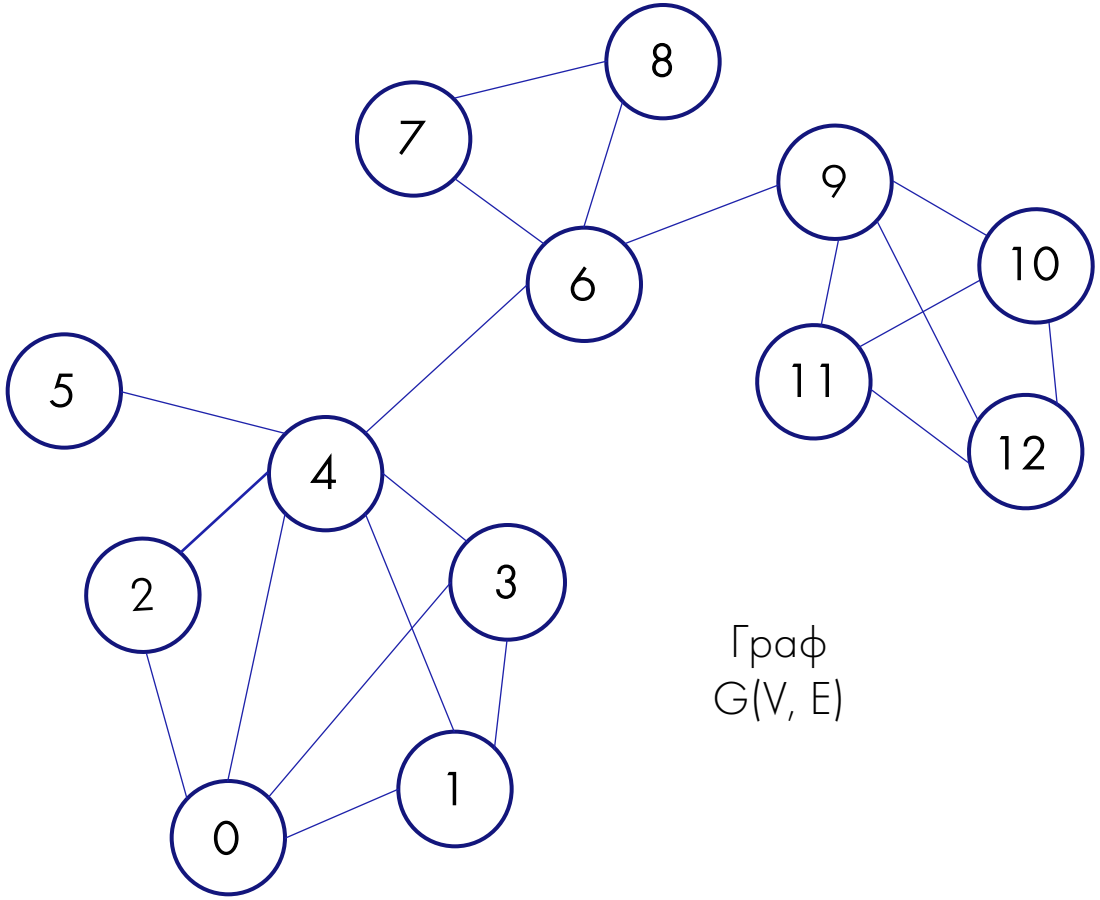
Граф
G(V, E)



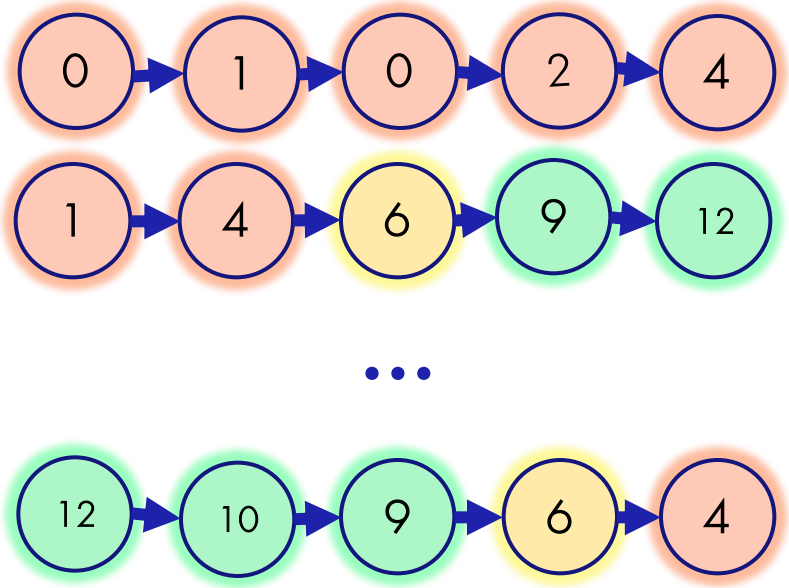
...

Как работает Node2Vec?

Шаг 1. Генерация последовательностей вершин

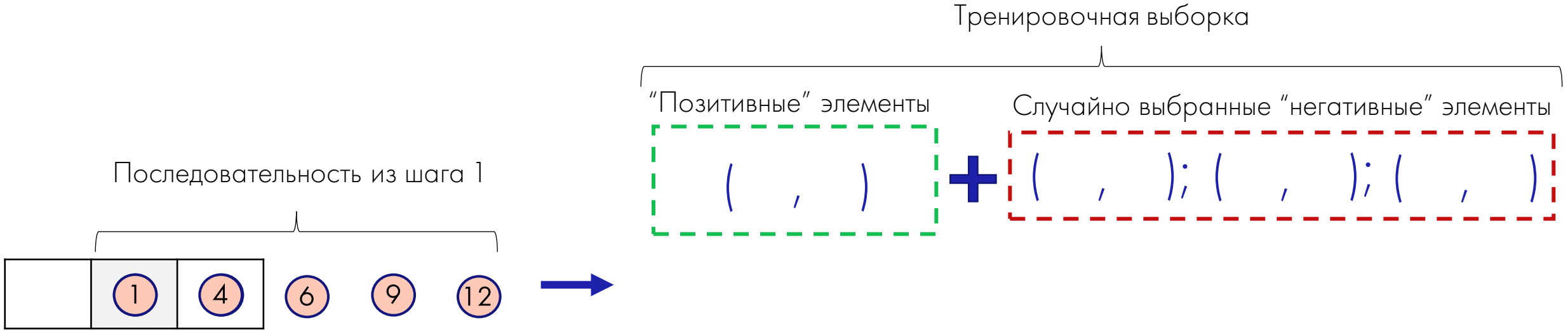


Граф
G(V, E)



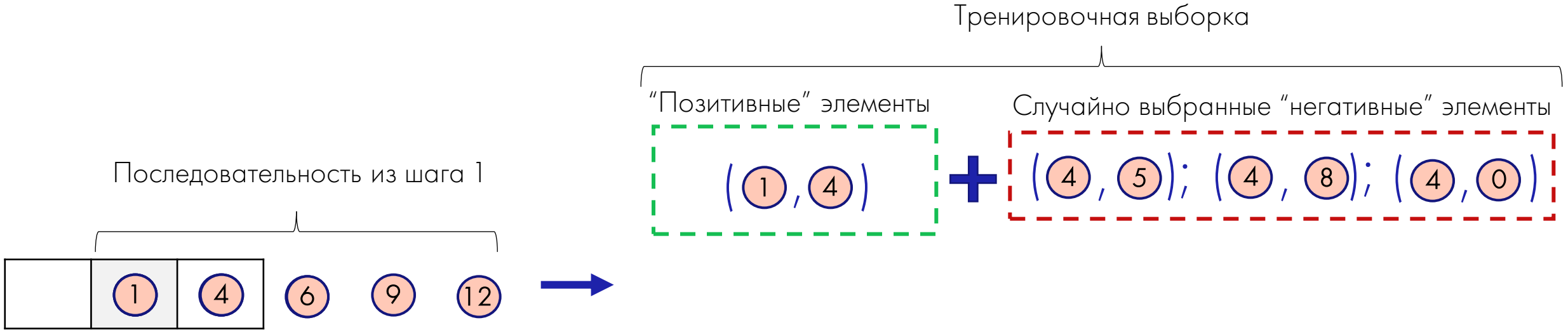
Как работает Node2Vec?

Шаг 2. Получение тренировочной выборки для Skip-gram



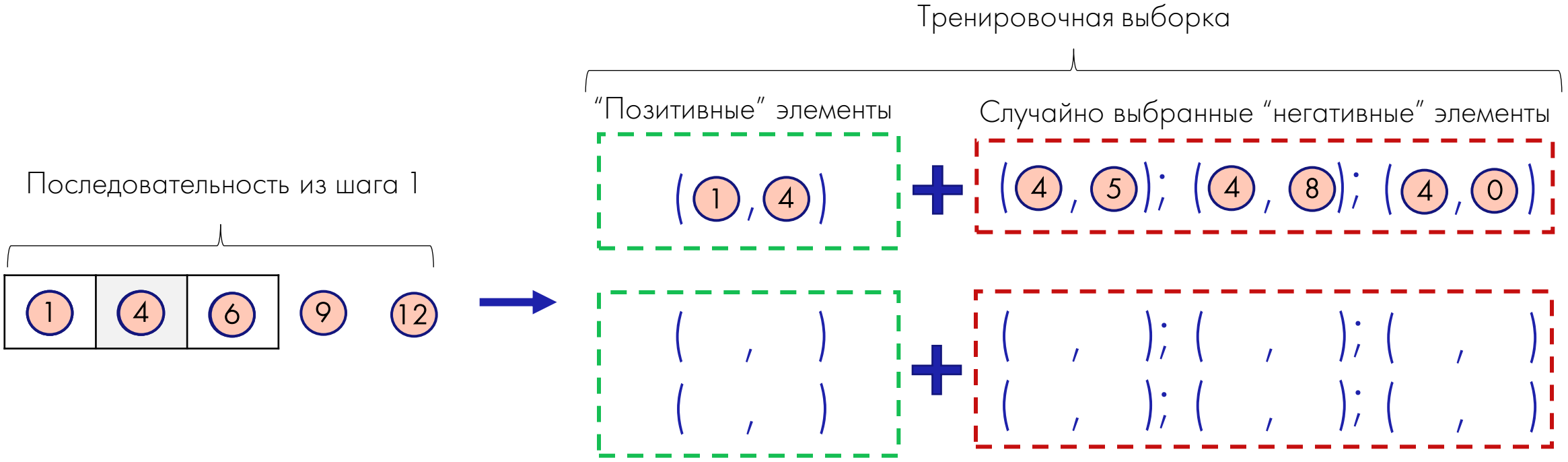
Как работает Node2Vec?

Шаг 2. Получение тренировочной выборки для Skip-gram



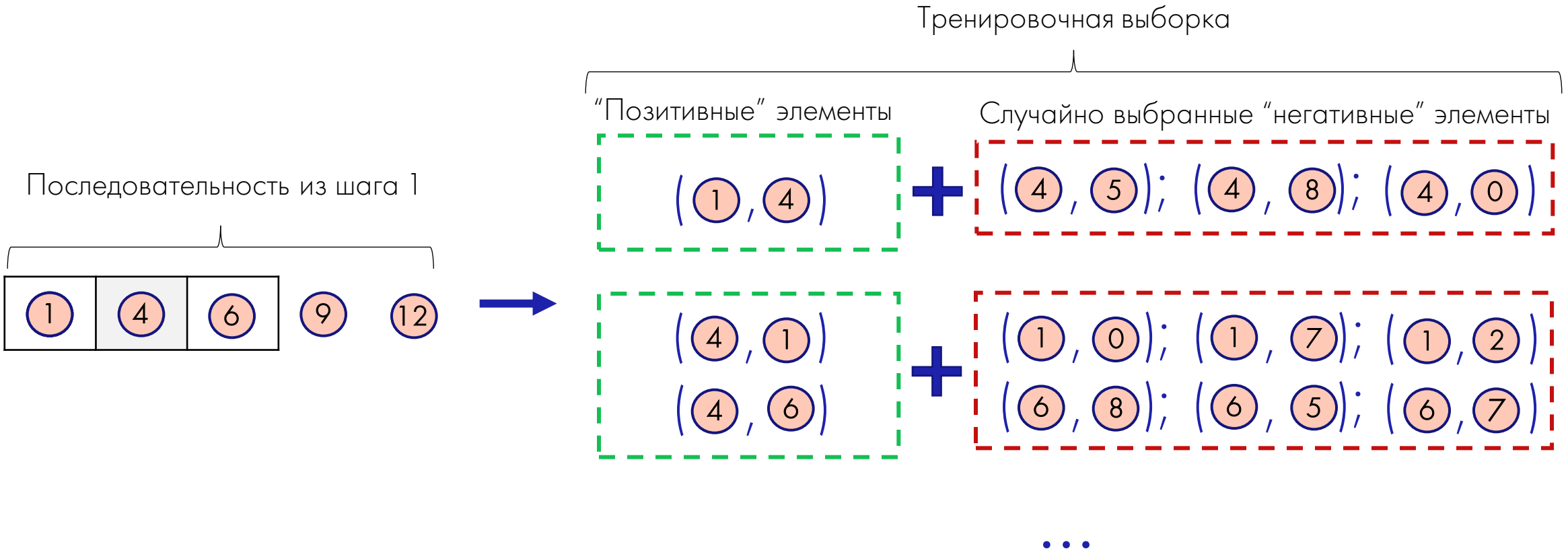
Как работает Node2Vec?

Шаг 2. Получение тренировочной выборки для Skip-gram



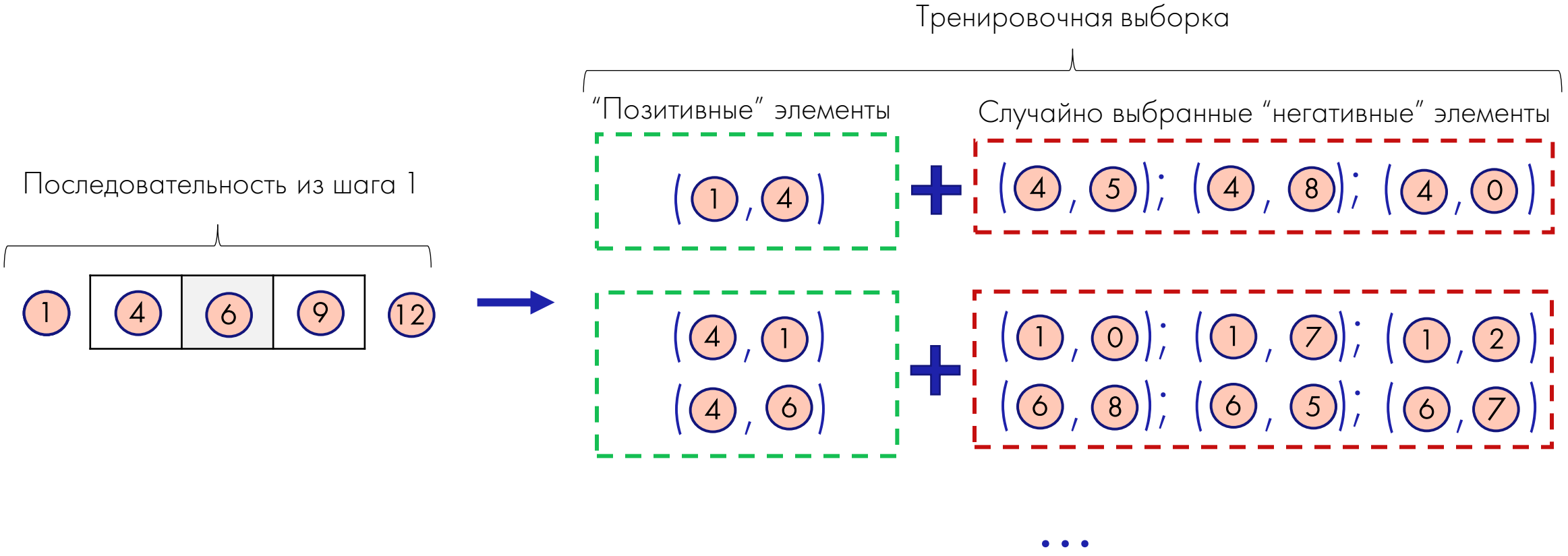
Как работает Node2Vec?

Шаг 2. Получение тренировочной выборки для Skip-gram



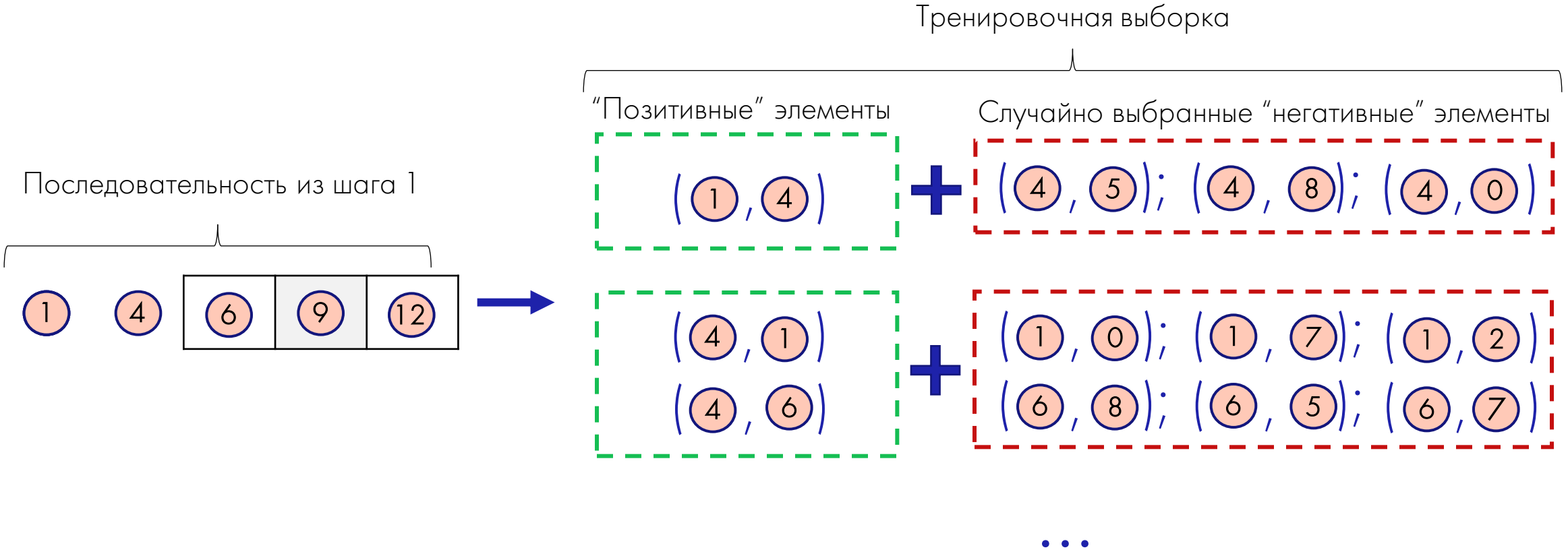
Как работает Node2Vec?

Шаг 2. Получение тренировочной выборки для Skip-gram



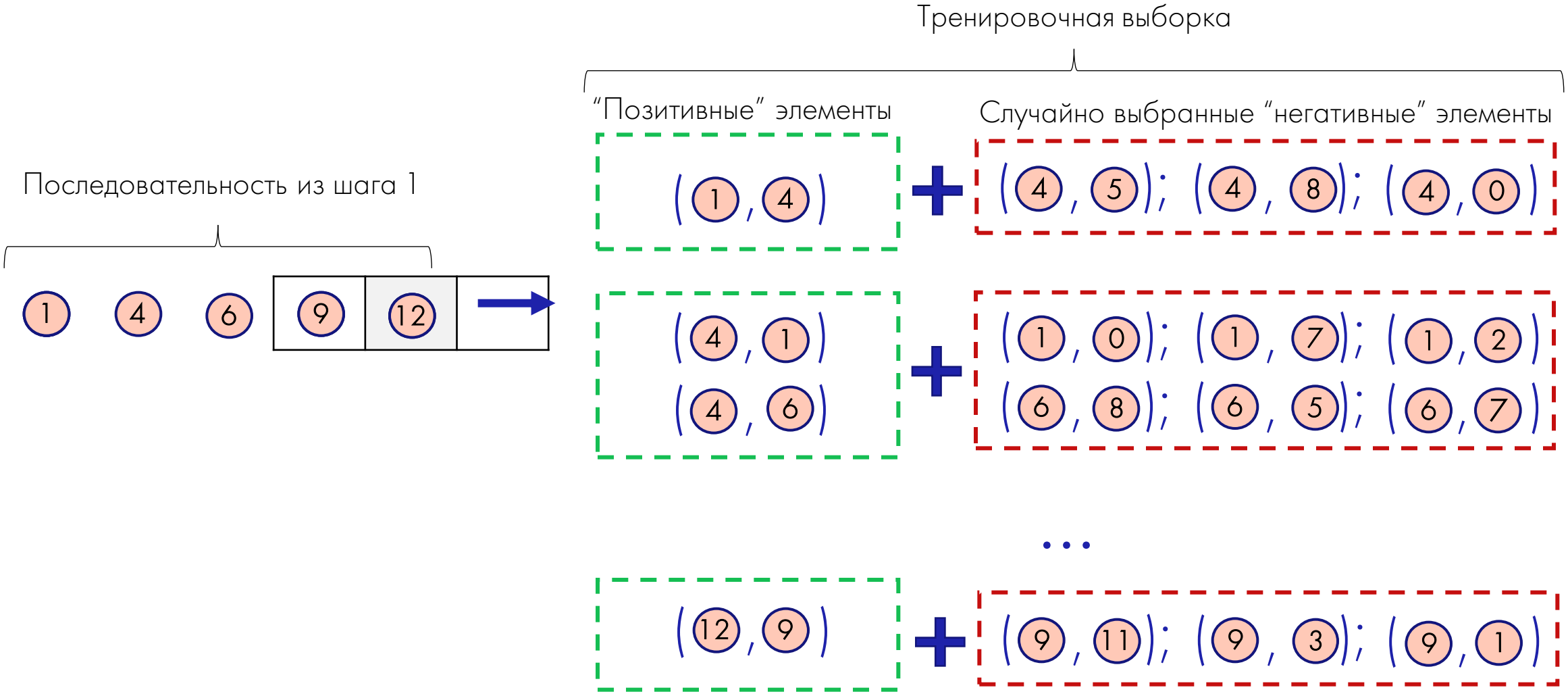
Как работает Node2Vec?

Шаг 2. Получение тренировочной выборки для Skip-gram



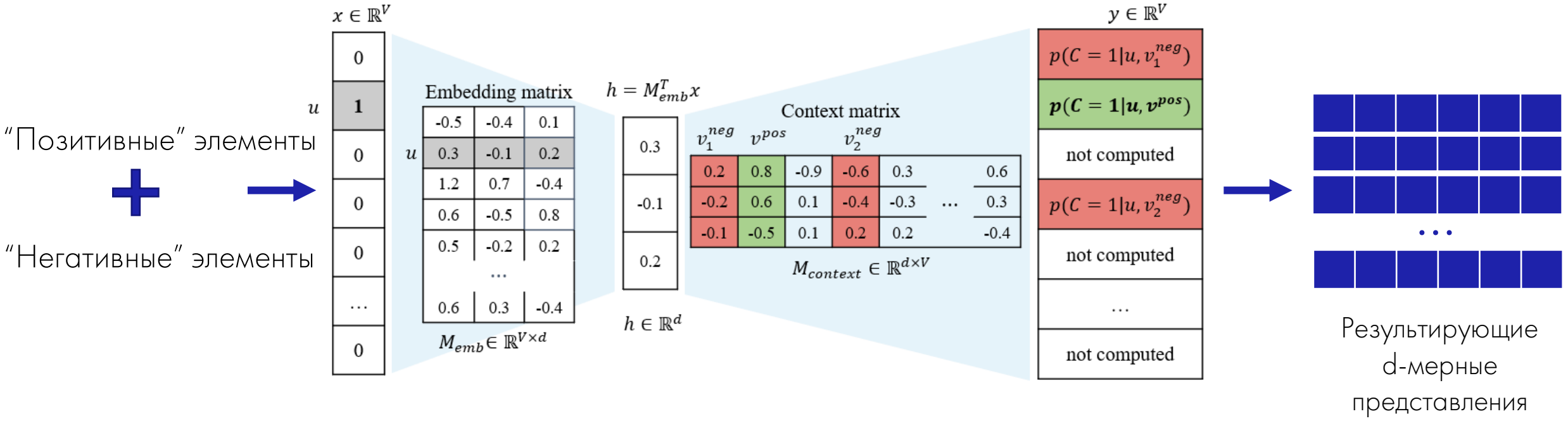
Как работает Node2Vec?

Шаг 2. Получение тренировочной выборки для Skip-gram



Как работает Node2Vec?

Step 3. Тренировка модели Skip-gram



Skip-gram модель с негативной выборкой

Что такое Node2Vec?

Мотивация и постановка задачи

Как работает Node2Vec?

Оптимизация 1. Блочная генерация случайных чисел

Оптимизация 2. Разбиение циклов

Оптимизация 3. Выравнивание по кэш-линиям

Контроль точности

Результаты

- Базовая реализация: случайные числа генерируются по мере надобности

```
do in parallel:
for all walks walk ∈ walks do
    for node u ∈ walk do
        Generate random integer number offset in interval [0, k)
        for all nodes v ∈ w inside the window of size 2 * (k - offset) around u do
            for iter = 1 to NegSamN + 1 do
                if iter == 1 then
                    target_node = u
                    Train embeddings of nodes from ”positive” pair (v, target_node)
                else
                    Generate two floating-point numbers num1 and num2
                    Select target_node using Alias method from the graph based on num1 and num2
                    Train embeddings of nodes from ”negative” pair (v, target_node)
…
```

- Базовая реализация: случайные числа генерируются по мере надобности

```
do in parallel:
for all walks walk ∈ walks do
    for node u ∈ walk do
        Generate random integer number offset in interval [0, k)
        for all nodes v ∈ w inside the window of size 2 * (k - offset) around u do
            for iter = 1 to NegSamN + 1 do
                if iter == 1 then
                    target_node = u
                    Train embeddings of nodes from ”positive” pair (v, target_node)
                else
                    Generate two floating-point numbers num1 and num2
                    Select target_node using Alias method from the graph based on num1 and num2
                    Train embeddings of nodes from ”negative” pair (v, target_node)
…
```

- **Оптимизированная реализация**: каждый поток имеет собственный блок памяти, заранее наполненный всеми случайными числами, которые потребуются

```
Allocate memory RndF* and RndI* for each thread
do in parallel:
for all walks walk ∈ walks do
    Generate random float numbers and fill the RndF* memory associated with the current thread
    Generate random integer numbers and fill the RndI* memory associated with the current thread
    for node u ∈ walk do
        Get offset value from RndI*
        for all nodes v ∈ walk inside the window around u do
            for iter = 1 to NegSamN + 1 do
                if iter == 1 then
                    target_node = u
                    Train embeddings of nodes from ”positive” pair (v, target_node)
                else
                    Get two floating-point numbers num1 and num2 from RndF*
                    Randomly select target_node from the graph based on num1 and num2
                    Train embeddings of nodes from ”negative” pair (v, target_node)
```

Что такое Node2Vec?

Мотивация и постановка задачи

Как работает Node2Vec?

Оптимизация 1. Блочная генерация случайных чисел

Оптимизация 2. Разбиение циклов

Оптимизация 3. Выравнивание по кэш-линиям

Контроль точности

Результаты

- **Базовая реализация**: обработка позитивных и негативных элементов производится в одном цикле. Обновление $M_{context}$ и M_{emb} также происходит в одном цикле.

```
do in parallel:
for all walks walk ∈ walks do
    for node u ∈ walk do
        Generate random integer number offset in interval [0, k)
        for all nodes v ∈ w inside the window of size 2 * (k - offset) around u do
            for iter = 1 to NegSamN + 1 do
                if iter == 1 then
                    target_node = u
                    Train embeddings of nodes from ”positive” pair (v, target_node)
                else
                    Generate two floating-point numbers num1 and num2
                    Select target_node using Alias method from the graph based on num1 and num2
                    Train embeddings of nodes from ”negative” pair (v, target_node)
```

…

- **Базовая реализация**: обработка позитивных и негативных элементов производится в одном цикле. Обновление $M_{context}$ и M_{emb} также происходит в одном цикле.

```
do in parallel:
for all walks walk ∈ walks do
    for node u ∈ walk do
        Generate random integer number offset in interval [0, k)
        for all nodes v ∈ w inside the window of size 2 * (k - offset) around u do
            Train embeddings of nodes from ”positive” pair (v, u)
            for iter = 1 to NegSamN do
                    Generate two floating-point numbers num1 and num2
                    Select target_node using Alias method from the graph based on num1 and num2
                    Train embeddings of nodes from ”negative” pair (v, target_node)
…
```

Что такое Node2Vec?

Мотивация и постановка задачи

Как работает Node2Vec?

Оптимизация 1. Блочная генерация случайных чисел

Оптимизация 2. Разбиение циклов

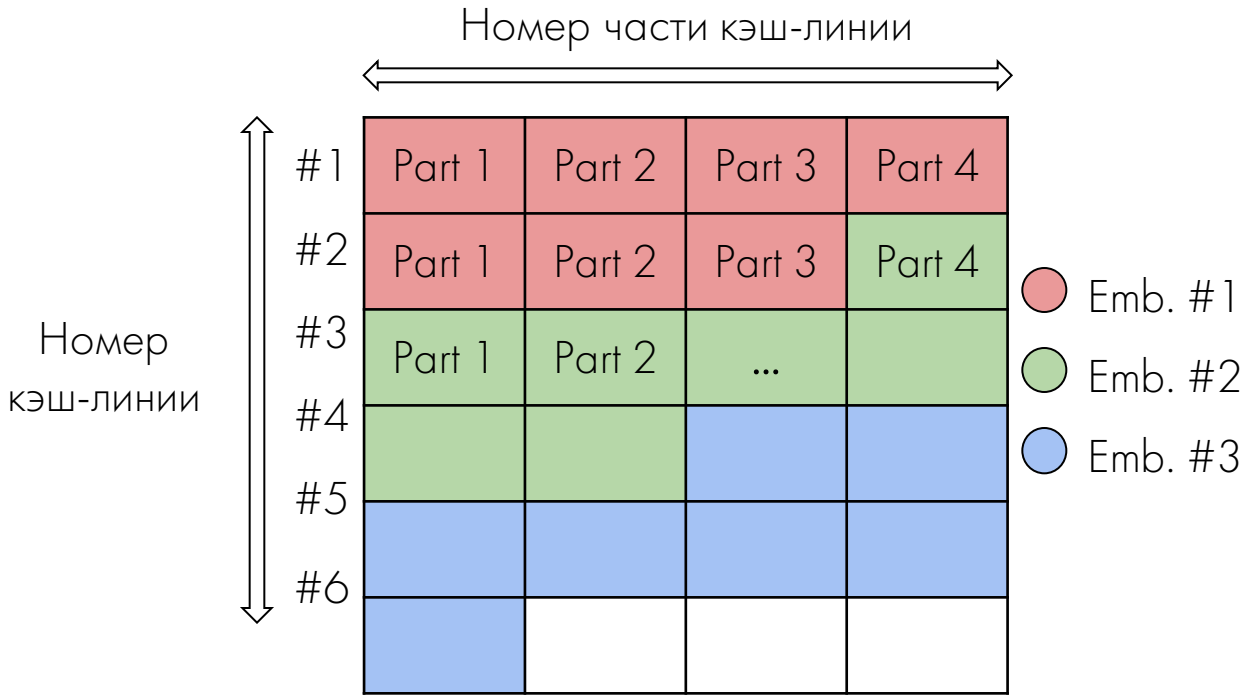
Оптимизация 3. Выравнивание по кэш-линиям

Контроль точности

Результаты

Оптимизация 3. Выравнивание по кэш-линиям

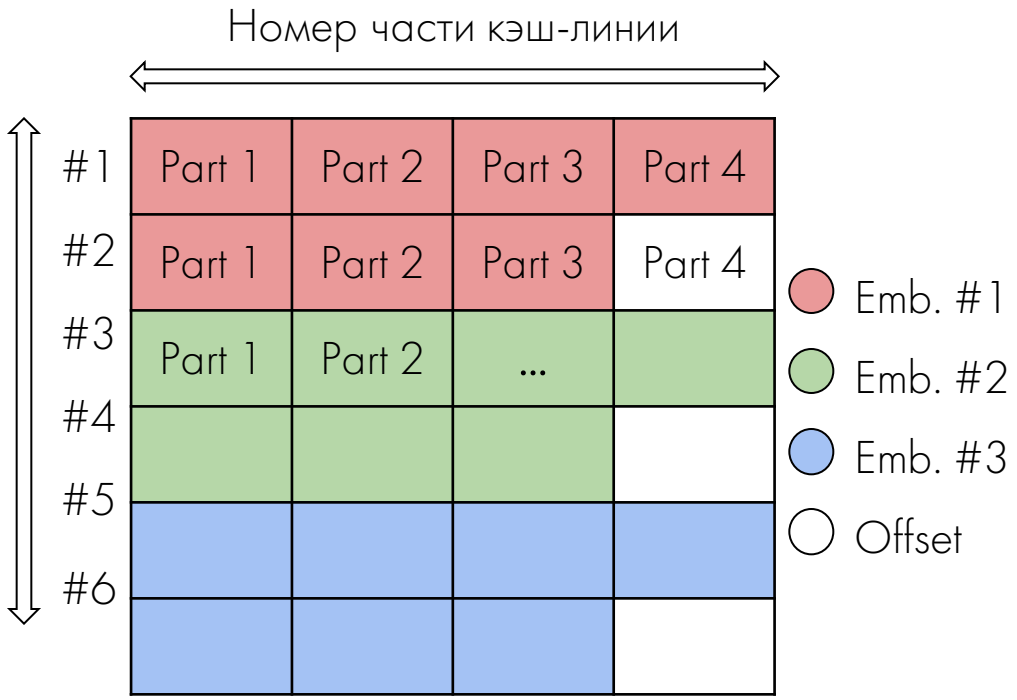
- **Базовая реализация**:
  векторные представления могут разделять кэш-линии, что приводит к излишней нагрузке на память (“false sharing”)



Оптимизация 3. Выравнивание по кэш-линиям

- **Базовая реализация**:
  векторные представления могут разделять кэш-линии, что приводит к излишней нагрузке на память (“false sharing”)

- **Optimized implementation**:
  каждое векторное представление расположено на эксклюзивно выделенных для него кэш-линиях, что исключает “false sharing”



Что такое Node2Vec?

Мотивация и постановка задачи

Как работает Node2Vec?

Оптимизация 1. Блочная генерация случайных чисел

Оптимизация 2. Разбиение циклов

Оптимизация 3. Выравнивание по кэш-линиям

Контроль точности

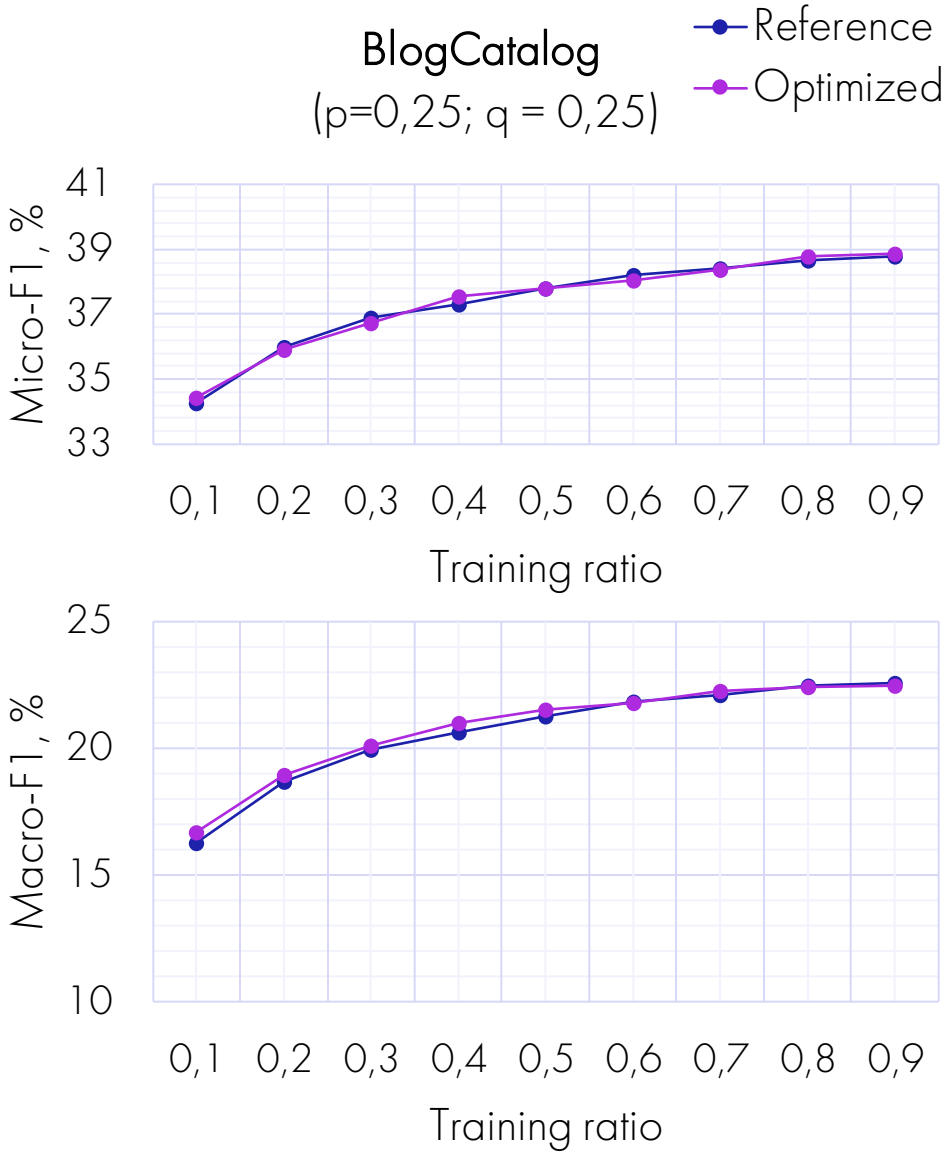
Результаты

Контроль точности

1. Вычислить представления вершин графа используя Node2Vec
2. Провести multi-label классификацию вершин на основании полученных представлений
3. Вычислить Micro-F1 и Macro-F1 метрики качества

Метрики качества остались на прежнем уровне

Dataset	Training ratio	Micro-F1					Macro-F1				
		0.1	0.3	0.5	0.7	0.9	0.1	0.3	0.5	0.7	0.9
PPI	Reference	15.8	19.4	20.7	21.7	22.3	11.5	15.7	17.3	18.2	18.0
	Optimized	16.5	19.7	21.0	21.7	22.1	12.5	16.3	17.7	18.5	18.3
Wikipedia	Reference	45.4	48.9	50.6	51.8	51.7	7.6	9.4	10.3	10.6	11.0
	Optimized	46.0	49.1	50.4	51	51.8	8.0	9.5	10.1	10.5	10.8
BlogCatalog	Reference	34.2	36.9	37.8	38.4	38.8	16.3	19.9	21.2	22.1	22.6
	Optimized	34.4	36.7	37.8	38.4	38.9	16.7	20.1	21.5	22.3	22.5
DBLP	Reference	58.6	59.9	60.1	60.3	60.4	5.7	5.9	5.9	6.0	6.0
	Optimized	59.2	60.0	60.3	60.4	60.4	5.9	6.1	6.1	6.1	6.1



Что такое Node2Vec?

Мотивация и постановка задачи

Как работает Node2Vec?

Оптимизация 1. Блочная генерация случайных чисел

Оптимизация 2. Разбиение циклов

Оптимизация 3. Выравнивание по кэш-линиям

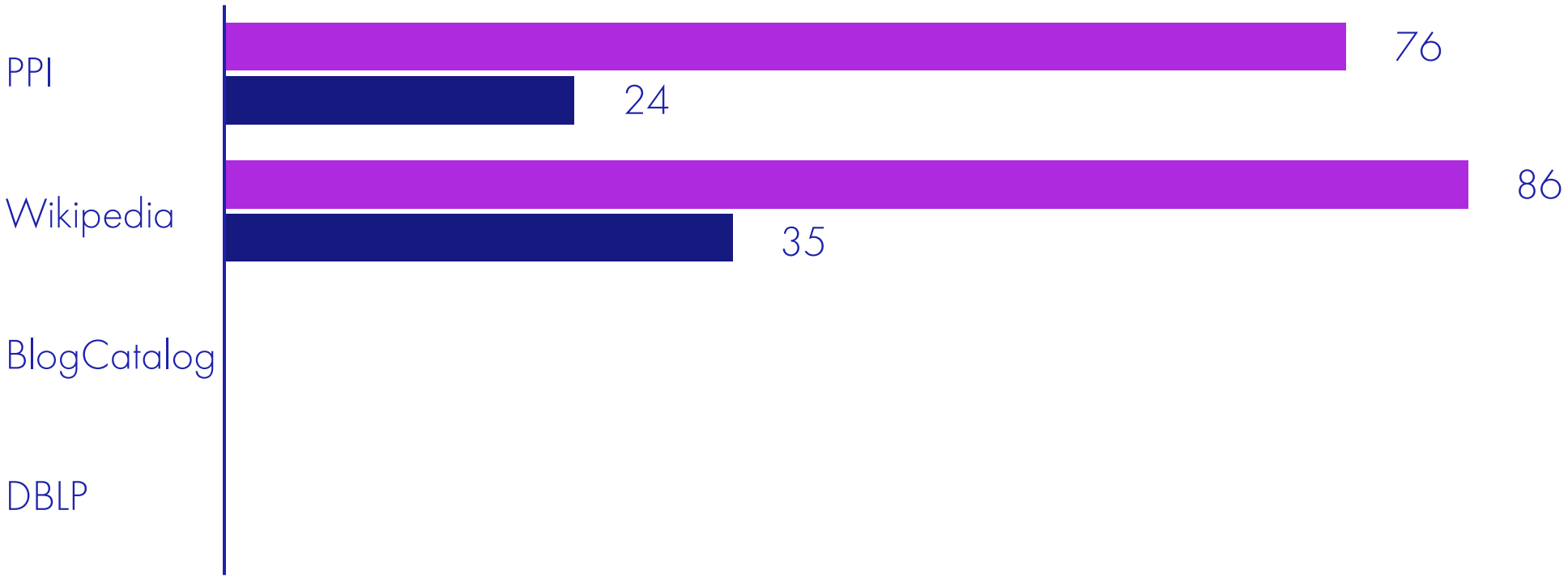
Контроль точности

Результаты

Графы для оценки производительности

Набор данных	Число вершин	Число ребер
PPI	3 890	76 584
Wikipedia	4 777	184 812
BlogCatalog	10 312	333 983
DBLP	51 264	127 968

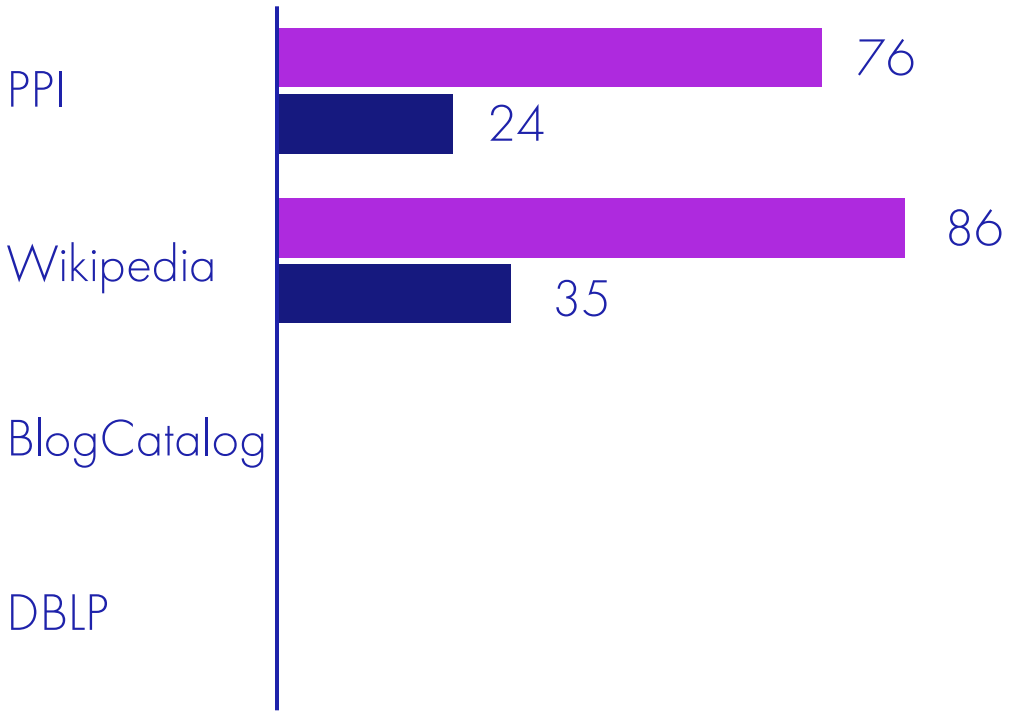
Оптимизированная реализация в 2.5-5.1 раз быстрее, чем базовая



Время работы алгоритма обозначено в секундах

Базовая
Оптимизированная

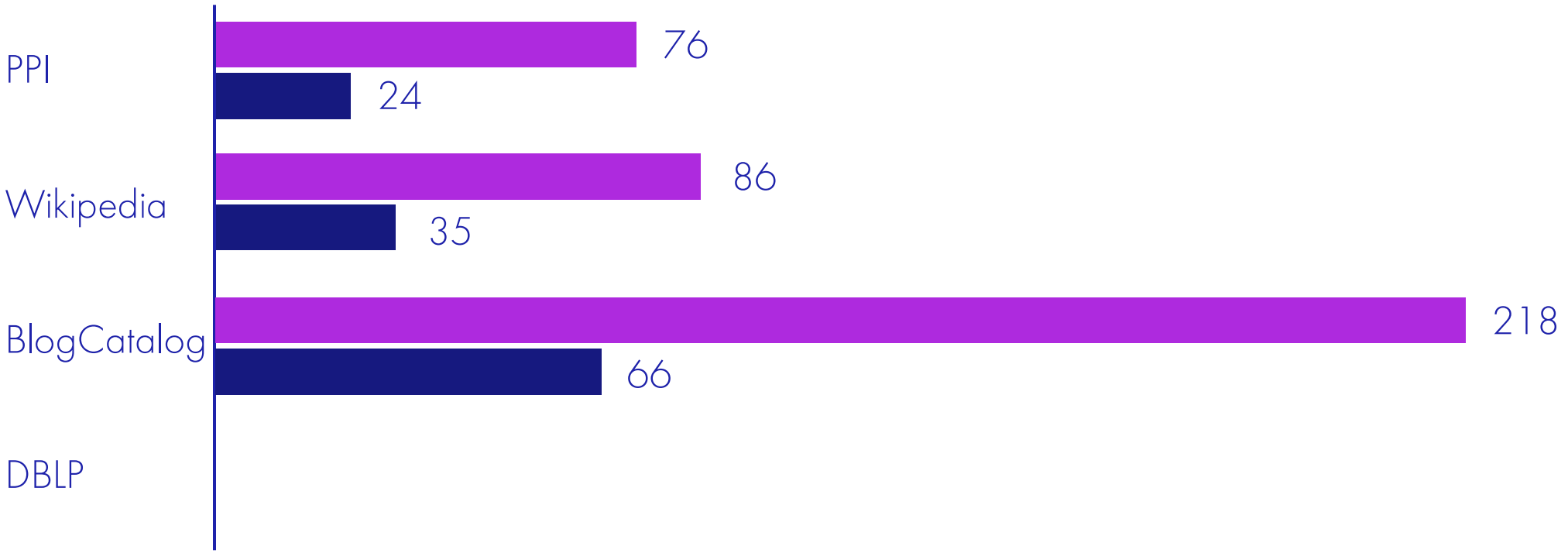
Оптимизированная реализация в 2.5-5.1 раз быстрее, чем базовая



Время работы алгоритма обозначено в секундах

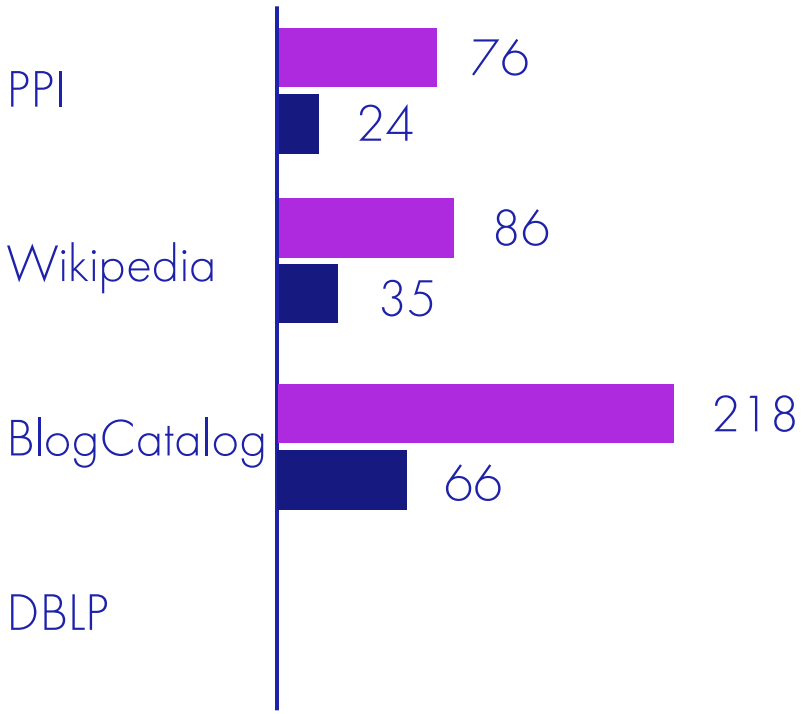
Базовая
Оптимизированная

Оптимизированная реализация в 2.5-5.1 раз быстрее, чем базовая



Время работы алгоритма обозначено в секундах

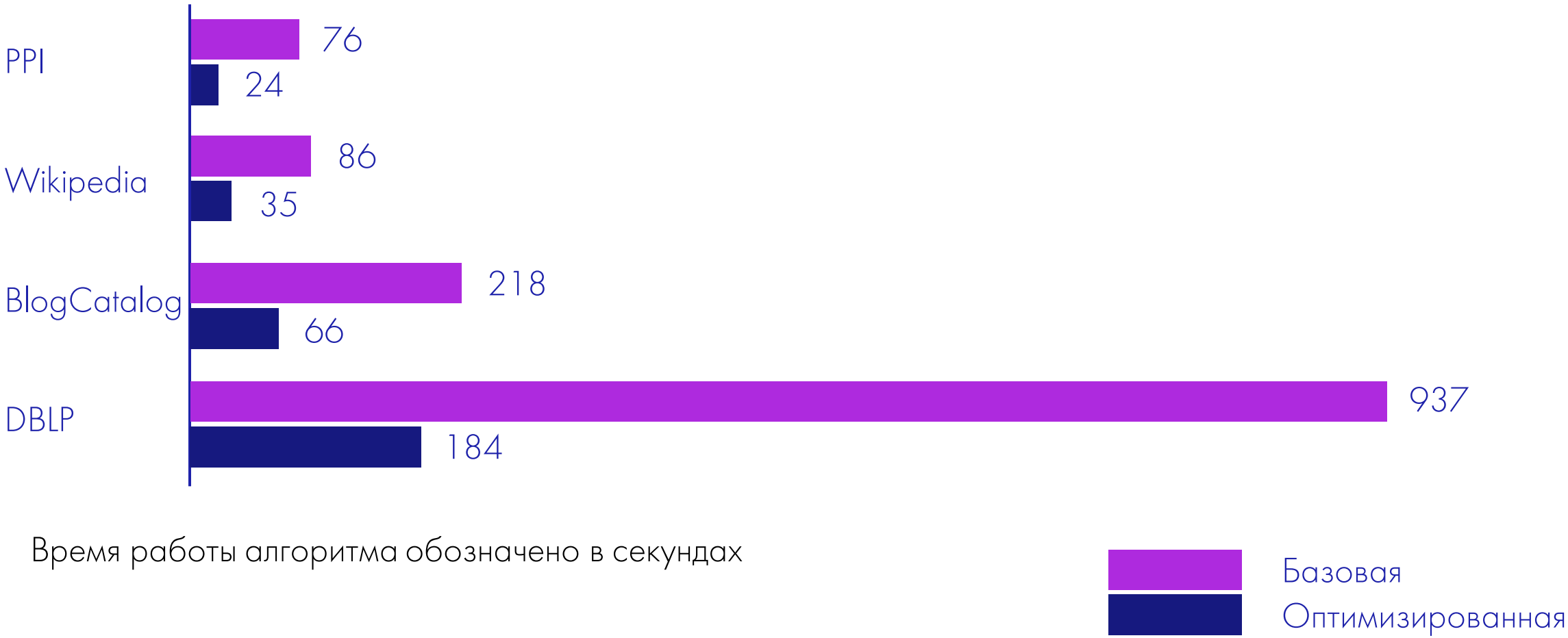
Оптимизированная реализация в 2.5-5.1 раз быстрее, чем базовая



Время работы алгоритма обозначено в секундах

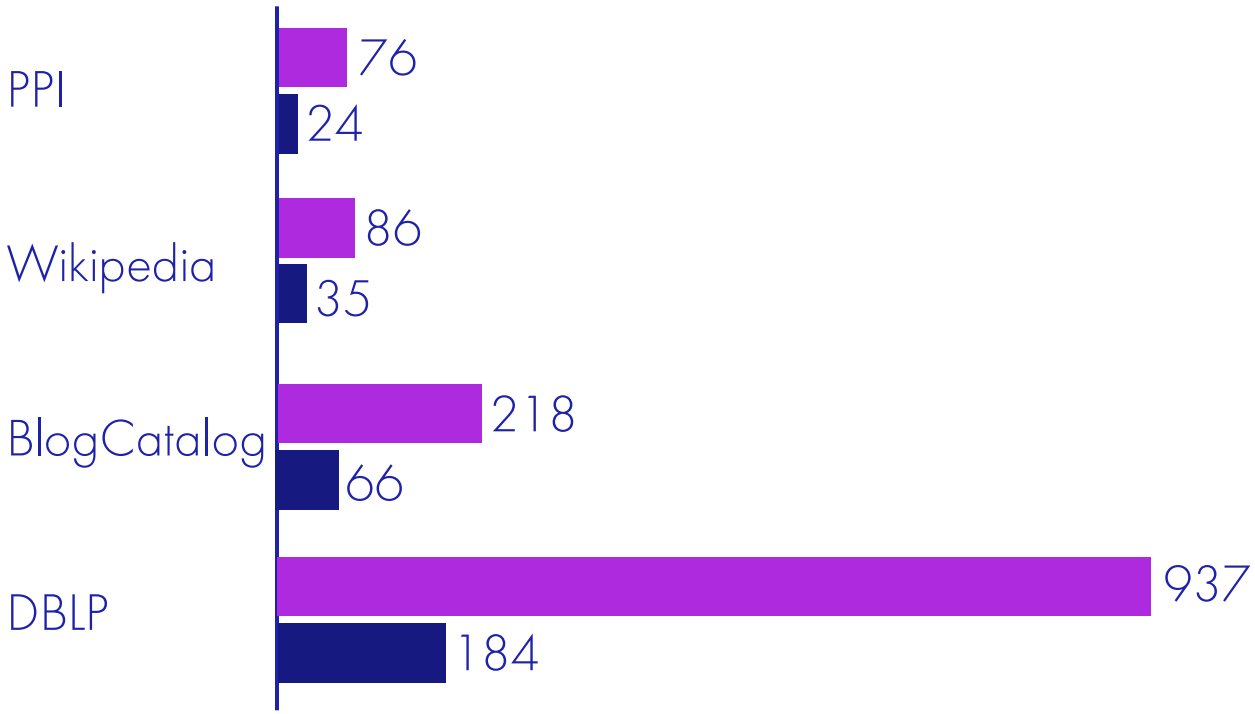
Базовая
Оптимизированная

Оптимизированная реализация в 2.5-5.1 раз быстрее, чем базовая



Время работы алгоритма обозначено в секундах

Оптимизированная реализация в 2.5-5.1 раз быстрее, чем базовая



Время работы алгоритма обозначено в секундах

Базовая
Оптимизированная

Сравнение с другими реализациями

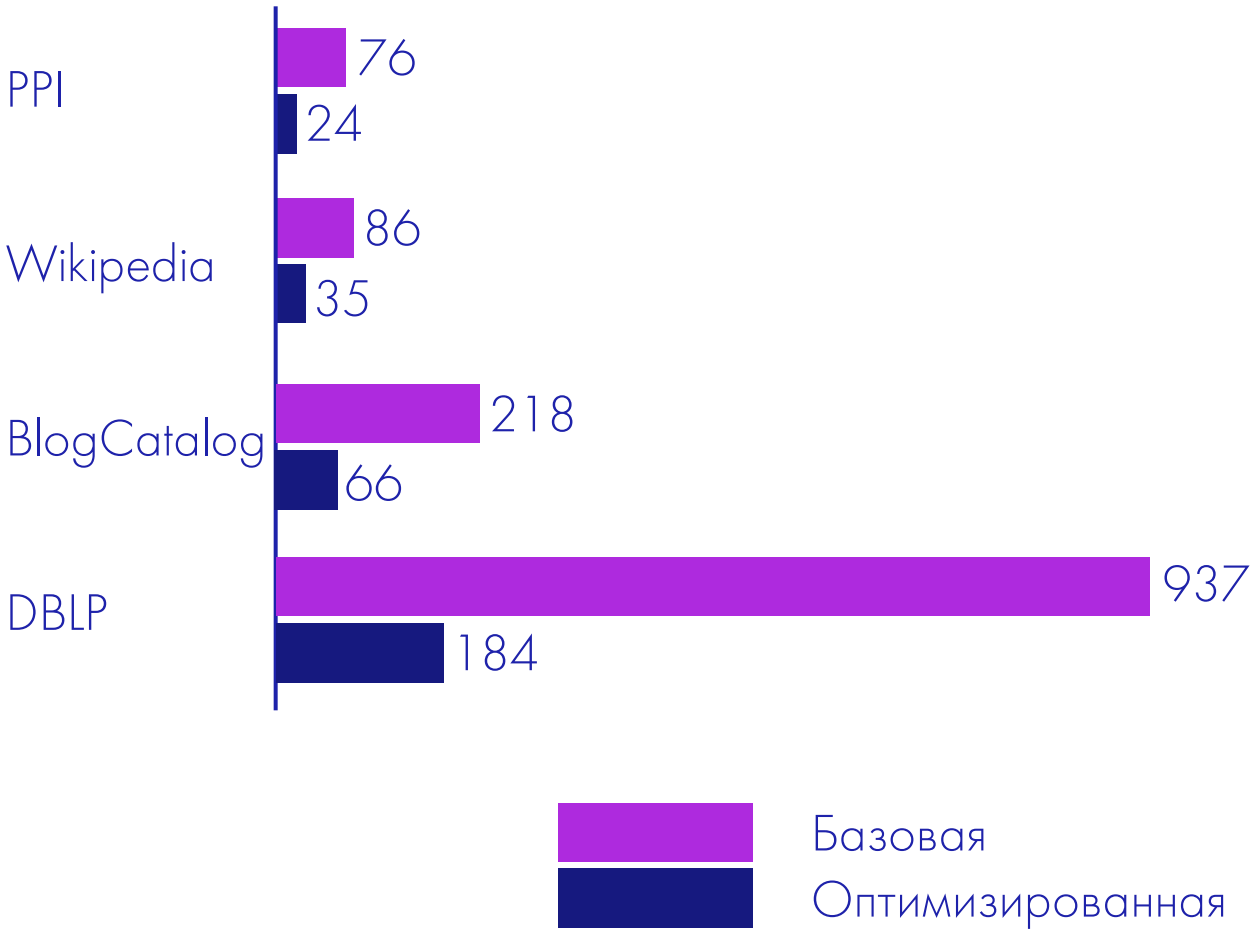
Набор данных	VHRanger/ nodevectors	eliorc/ node2vec	shenweichen/ GraphEmbedding	SNAP	Наша реализация
PPI	**21**	76	42	76	24
Wikipedia	49	377	140	86	**35**
BlogCatalog	139	1366	363	218	**66**
DBLP	**132**	360	348	937	184

Время работы алгоритма обозначено в секундах
Наименьшее время работы подчеркнуто в каждой строке

Заключение

- Мы сделали оптимизацию Node2Vec алгоритма, которая ускоряет работу в 2.5-5.1 раз по сравнению с базовой C++ реализацией от Stanford Network Analysis Platform (SNAP)

- Качество работы оптимизированного алгоритма осталось на прежнем уровне



Спасибо!