ПАРАЛЛЕЛЬНЫЕ АЛГОРИТМЫ ПОИСКА КРАТЧАЙШИХ ПУТЕЙ НА ГРАФАХ

Выполнил: Ткаченко Г.С.

Руководитель: Корнеев Г.А.

10 мая 2015 г.

Университет ИТМО

Проблема и задача

РЕШАЕМАЯ ПРОБЛЕМА

- Низкая производительность отдельных алгоритмов на специфичных графах
- Недостаточное разнообразие параллельных алгоритмов для поиска кратчайших путей

ПОСТАНОВКА ЗАДАЧИ

- Эффективное применение алгоритмов поиска кратчайшего пути на **многопроцессорных** архитектурах
- Разработка алгоритмов для поиска пути от одной вершины до всех (**one-to-many**)
- Разработка алгоритмов для поиска пути кратчайшего расстояния между каждой парой вершин (**many-to-many**)

ЗАДАЧА ONE-TO-MANY

Обзор решений

- Алгоритм Дейкстры
- Алгоритм Беллмана-Форда
- Алгоритм Джонсона (Дейкстра с потенциалами)
- Алгоритмы A* и D*

Классический Беллман-Форд

```
1: procedure ClassicBellmanFord(G, start)
2:     dist ← {∞...∞}
3:     dist[start] ← 0
4:     for i = 0 to |G.vertices| − 1 do
5:         for e ∈ G.edges do
6:             dist[e.to] ← min(dist[e.to], dist[e.from] + e.w)
7:     return dist
```

BFS Беллман-Форд

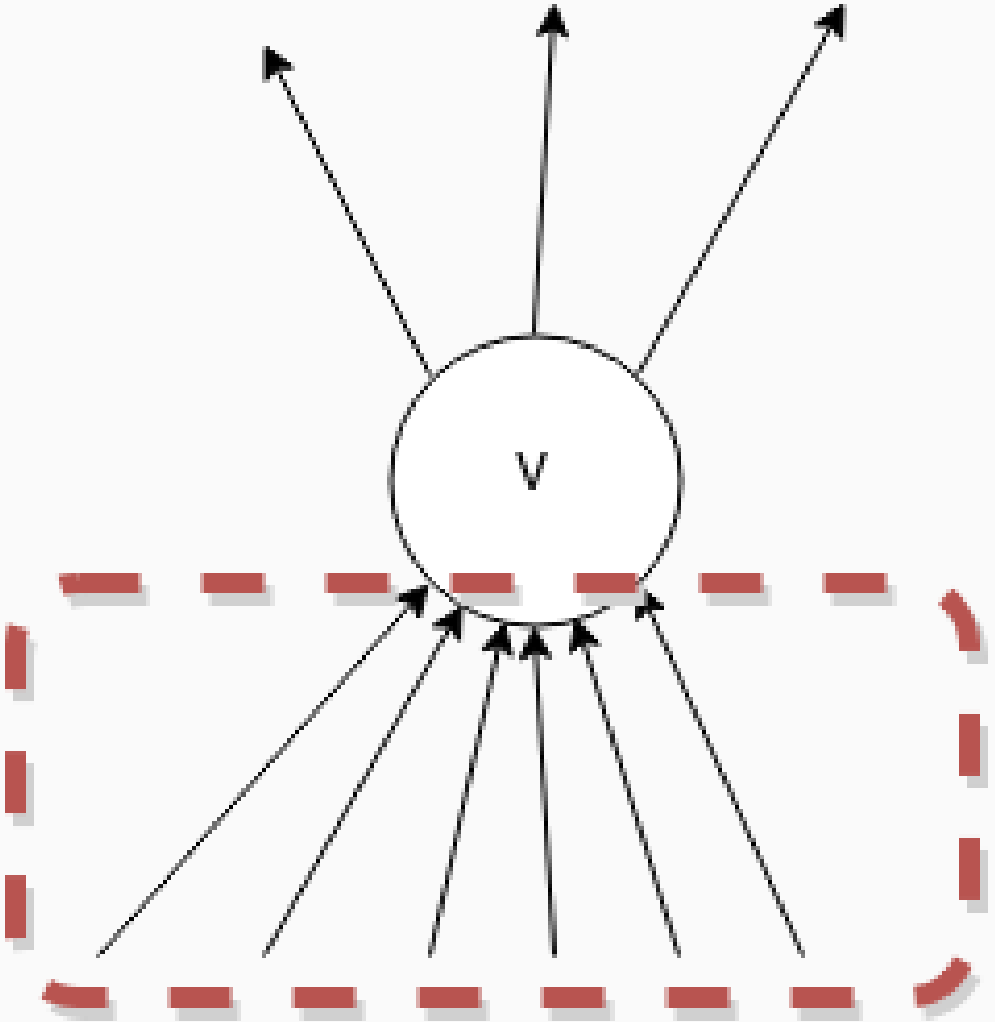
```
1:  procedure BFSBellmanFord(G, start)
2:      dist ← {∞...∞}
3:      dist[start] ← 0
4:      CurrentVertexSet ← {start}
5:      NextVertexSet ← ∅
6:      while CurrentVertexSet.empty() do
7:          NextVertexSet.clear()
8:          for v ∈ CurrentVertexSet do
9:              for e ∈ G.edgesFrom[v] do
10:                 if dist[e.to] > dist[e.from] + e.w then
11:                     dist[e.to] ← dist[e.from] + e.w
12:                     NextVertexSet.insert(e.to)
13:         CurrentVertexSet ← NextVertexSet
14:     return dist
```

Параллельный Беллман-Форд

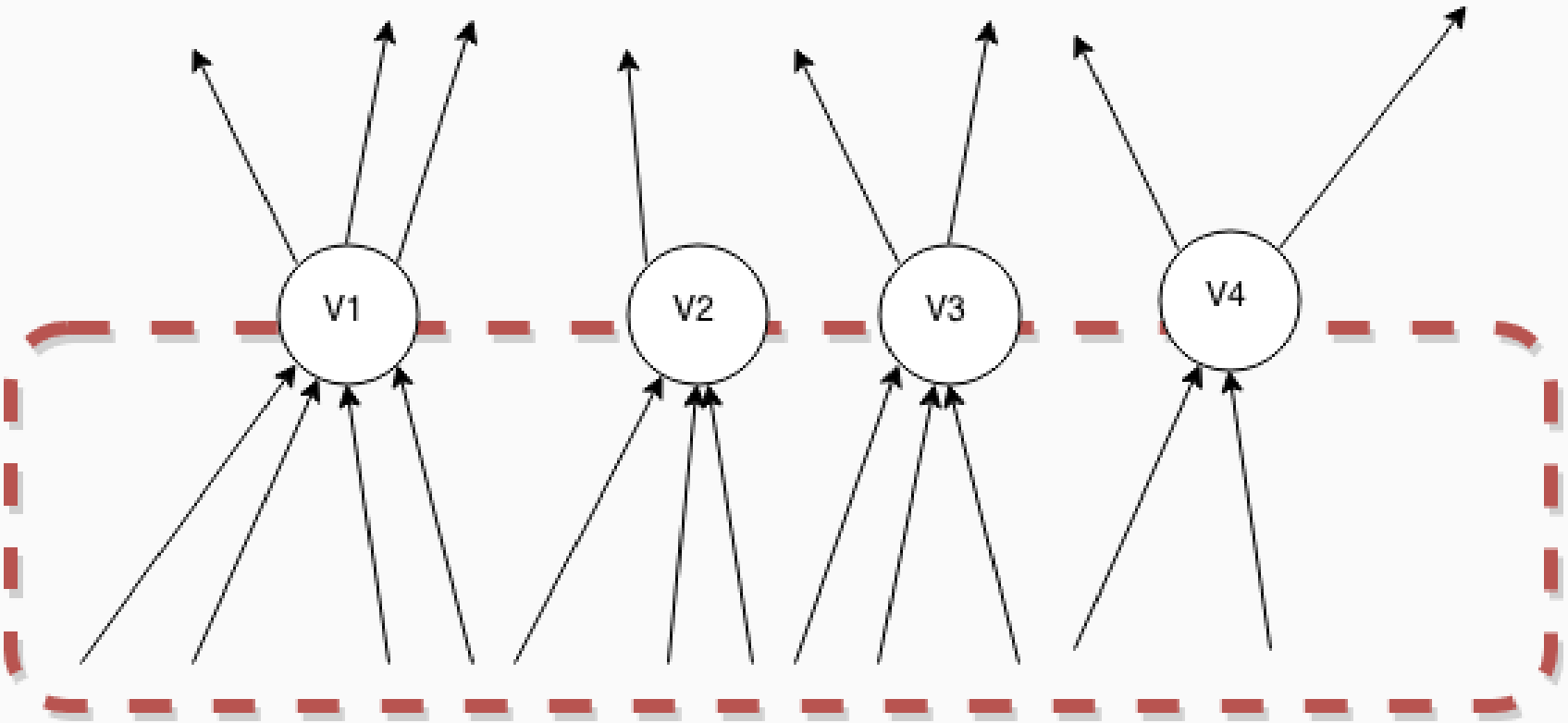
Три подхода

- Параллелизация по ребрам вершины
- Параллелизация по всем ребрам
- Использование параллельного обхода в ширину

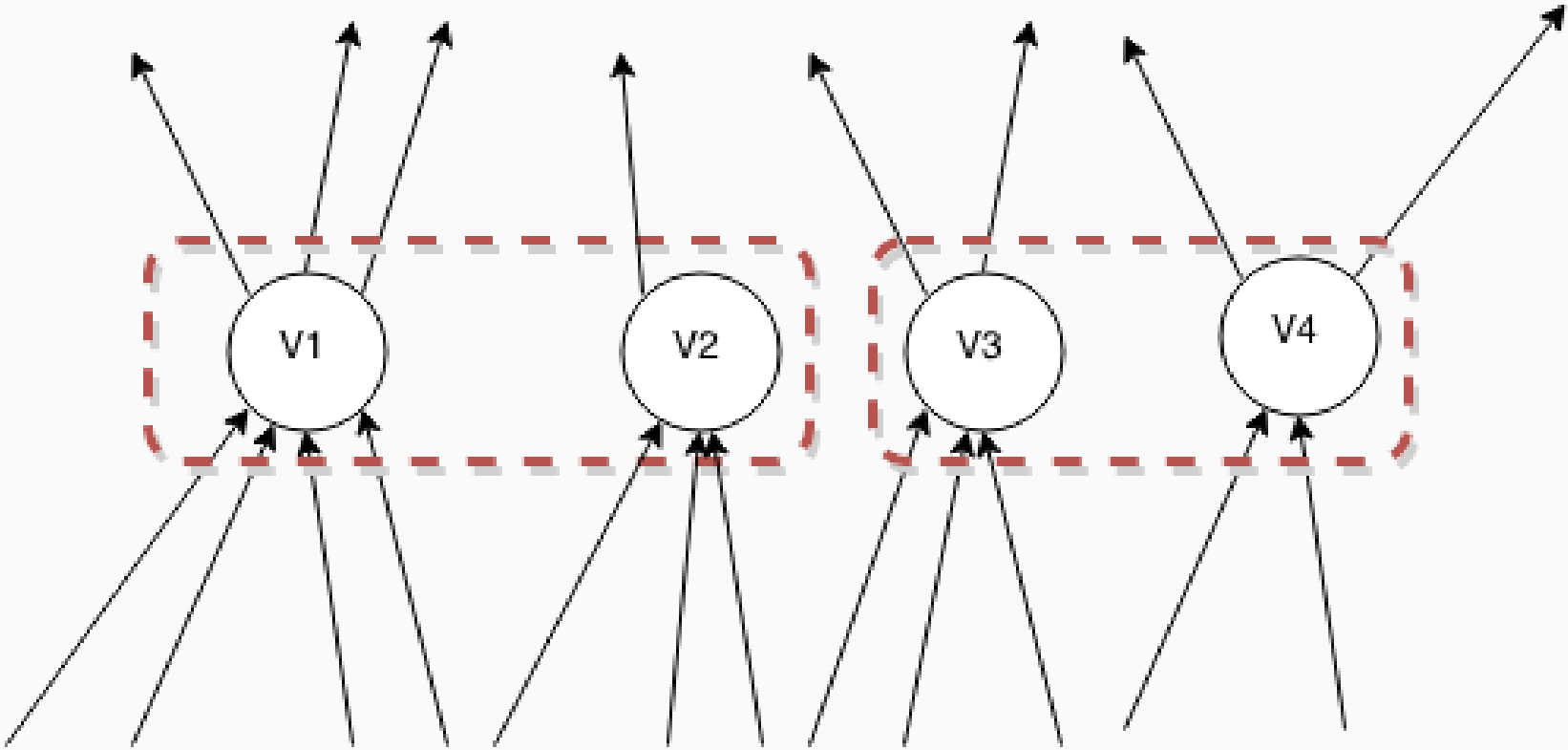
ПАРАЛЛЕЛИЗАЦИЯ ПО РЕБРАМ ВЕРШИНЫ



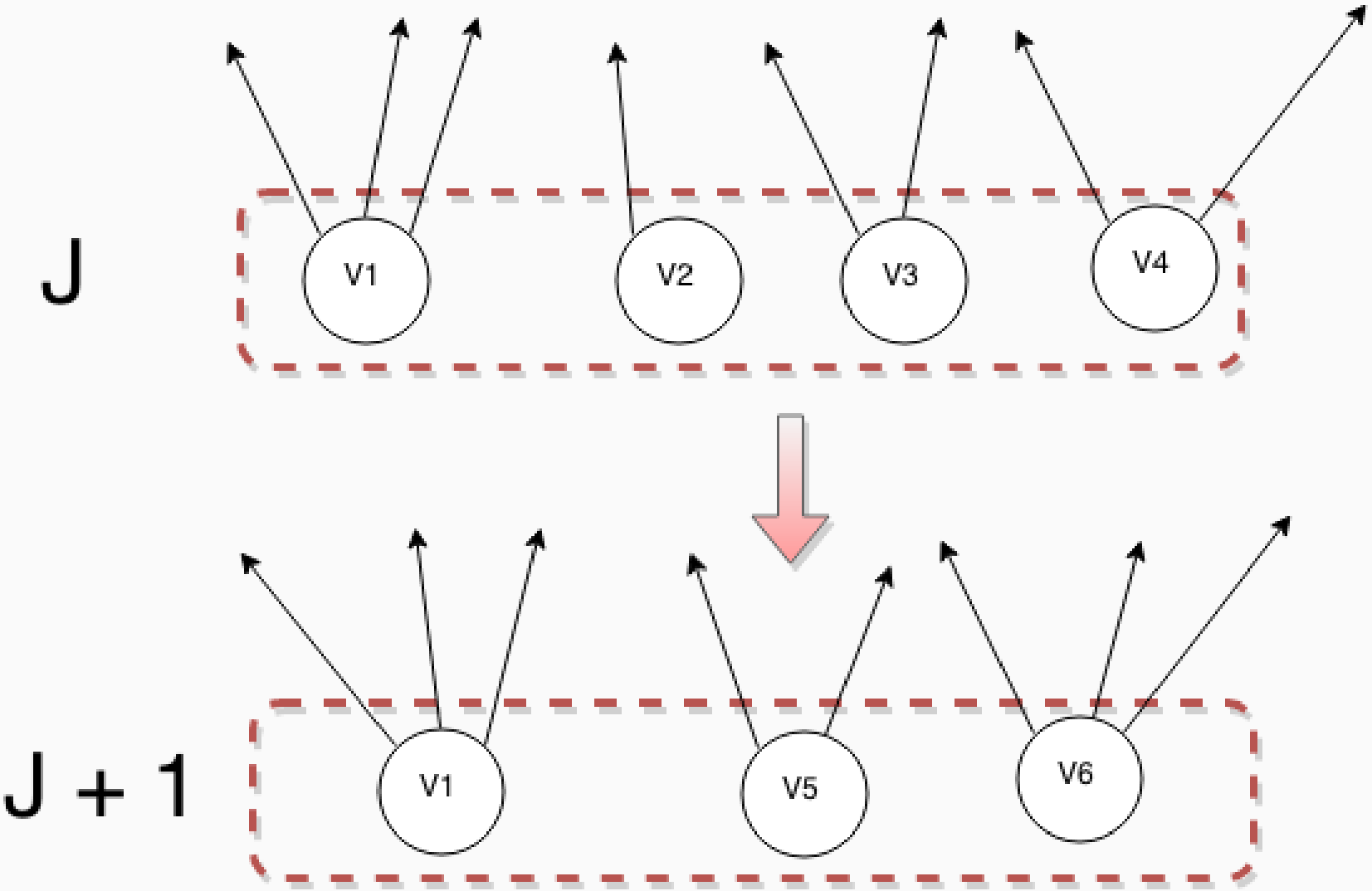
ПАРАЛЛЕЛИЗАЦИЯ ПО ВСЕМ РЕБРАМ



ПАРАЛЛЕЛИЗАЦИЯ ПО ВСЕМ РЕБРАМ



ИСПОЛЬЗОВАНИЕ ПАРАЛЛЕЛЬНОГО ОБХОДА В ШИРИНУ



ЗАДАЧА MANY-TO-MANY

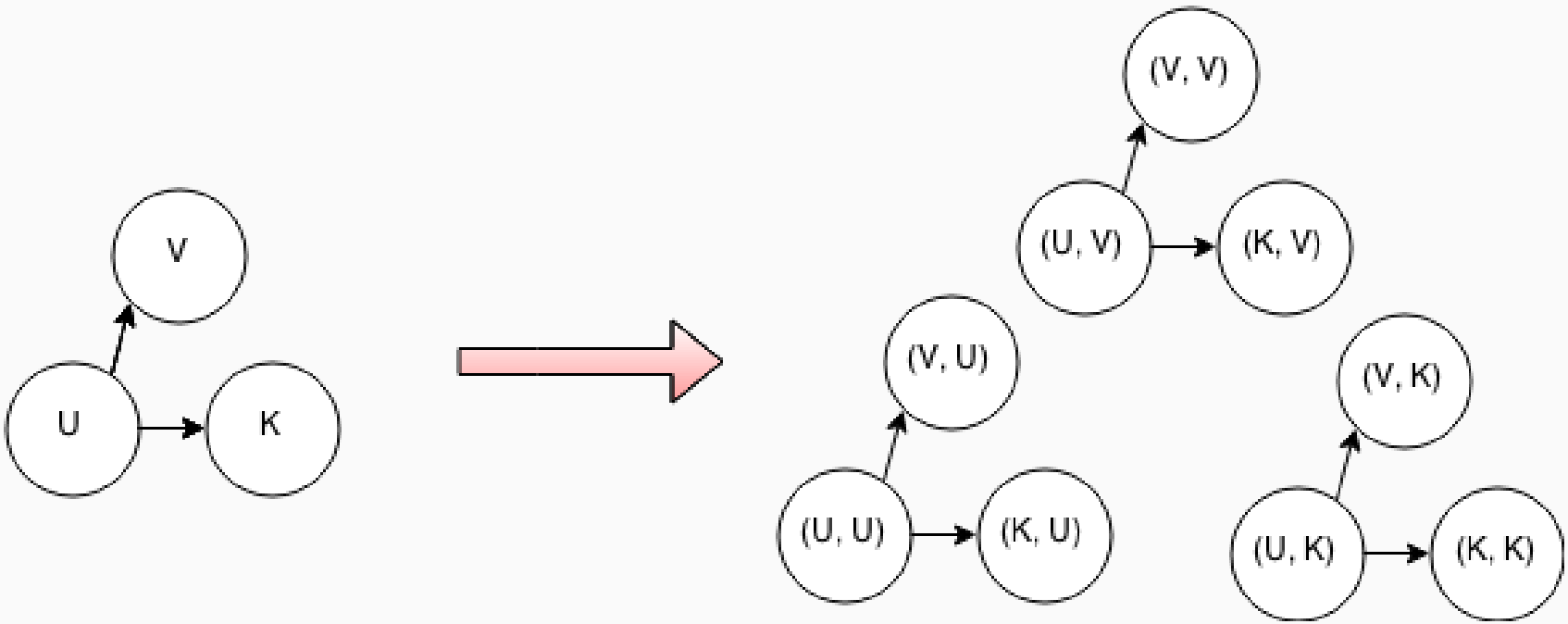
АЛГОРИТМ ФЛОЙДА

- В некоторых случаях классический алгоритм оказывается медленнее наивных алгоритмов
- Для каждой вершины можно использовать любой алгоритм поиска кратчайшего пути

НАИВНАЯ ПАРАЛЛЕЛЬНАЯ ВЕРСИЯ

```
1:  procedure AllPairsPar1(G)
2:      return HandleVertices(G, 0, |G.vertices|)
3:
4:  procedure HandleVertices(G, startVertex, endVertex)
5:      if endVertex − startVertex < threshold then
6:          distances ← run Bellman-Ford for [startVertex, endVertex)
7:          return distances
8:      else
9:          midV ← (startVertex + endVertex)/2
10:         fork2(
                HandleVertices(G, startV, midV),
                HandleVertices(G, midV, endV));
```

АЛГОРИТМ ДЛЯ ОБЪЕДИНЕННОГО ГРАФА



АЛГОРИТМ ДЛЯ СОЦИАЛЬНЫХ ГРАФОВ

- Основан на теории ”Шести рукопожатий”
- Работает не неориентированных невзвешенных социальных графах
- Использует динамическое программирование

РЕЗУЛЬТАТЫ

Таблица: Largest cities in the world (source: Wikipedia)

City	Population
Mexico City	20,116,842
Shanghai	19,210,000
Peking	15,796,450
Istanbul	14,160,467

SUMMARY

Get the source of this theme and the demo presentation from

`github.com/matze/mtheme`

ВЫВОДЫ

Вопросы?