

# ПАРАЛЛЕЛЬНЫЕ АЛГОРИТМЫ ПОИСКА КРАТЧАЙШИХ ПУТЕЙ НА ГРАФАХ

---

Выполнил: Ткаченко Г.С.

Руководитель: Корнеев Г.А.

12 мая 2015 г.

Университет ИТМО

# ПРОБЛЕМА И ЗАДАЧА

---

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

- 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

---

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

```
1: procedure CLASSICBELLMANFORD( $G$ , start)
2:    $\text{dist} \leftarrow \{\infty \dots \infty\}$ 
3:    $\text{dist}[\text{start}] \leftarrow 0$ 
4:   for  $i = 0$  to  $|G.\text{vertices}| - 1$  do
5:     for  $e \in G.\text{edges}$  do
6:        $\text{dist}[e.\text{to}] \leftarrow \min(\text{dist}[e.\text{to}], \text{dist}[e.\text{from}] + e.w)$ 
7:   return  $\text{dist}$ 
```

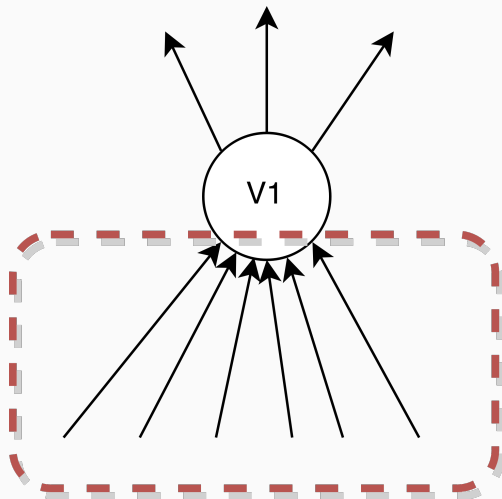
```
1: procedure BFSBELLMANFORD( $G$ , start)
2:    $\text{dist} \leftarrow \{\infty \dots \infty\}$ 
3:    $\text{dist}[\text{start}] \leftarrow 0$ 
4:    $\text{CurrentVertexSet} \leftarrow \{\text{start}\}$ 
5:    $\text{NextVertexSet} \leftarrow \emptyset$ 
6:   while  $\text{CurrentVertexSet.empty}()$  do
7:      $\text{NextVertexSet.clear}()$ 
8:     for  $v \in \text{CurrentVertexSet}$  do
9:       for  $e \in G.\text{edgesFrom}[v]$  do
10:        if  $\text{dist}[e.\text{to}] > \text{dist}[e.\text{from}] + e.w$  then
11:           $\text{dist}[e.\text{to}] \leftarrow \text{dist}[e.\text{from}] + e.w$ 
12:           $\text{NextVertexSet.insert}(e.\text{to})$ 
13:      $\text{CurrentVertexSet} \leftarrow \text{NextVertexSet}$ 
14:   return  $\text{dist}$ 
```



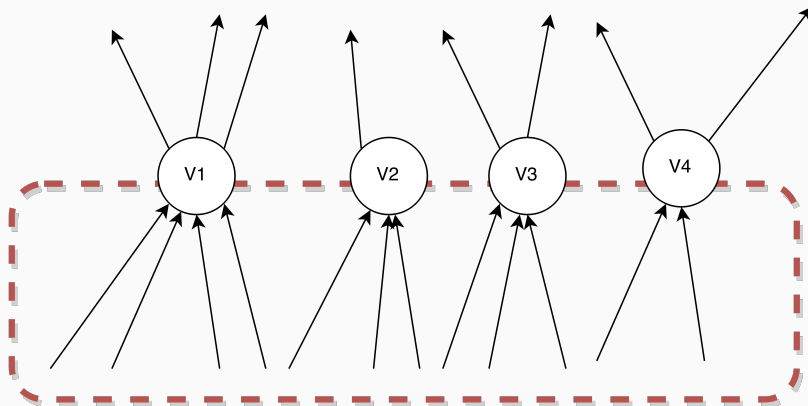
Три подхода

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

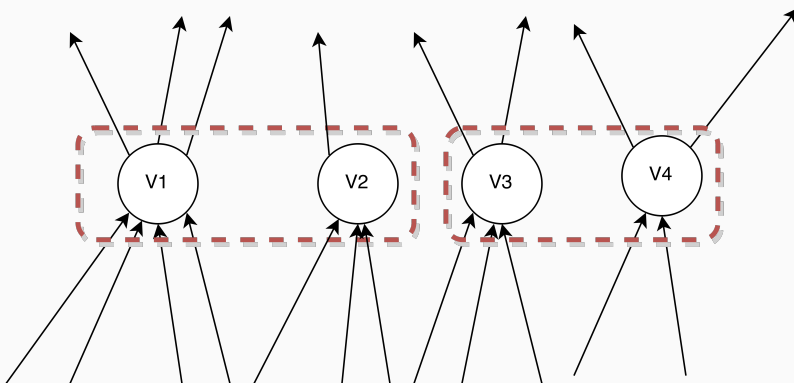
## ПАРАЛЛЕЛИЗАЦИЯ ПО РЕБРАМ ВЕРШИНЫ



# ПАРАЛЛЕЛИЗАЦИЯ ПО ВСЕМ РЕБРАМ

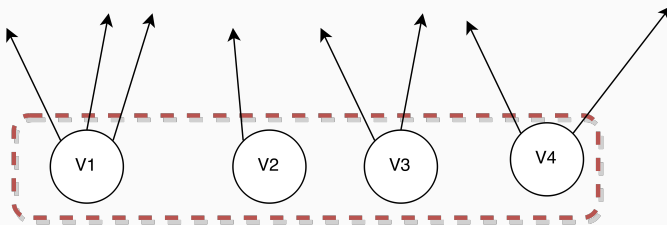


# ПАРАЛЛЕЛИЗАЦИЯ ПО ВСЕМ РЕБРАМ

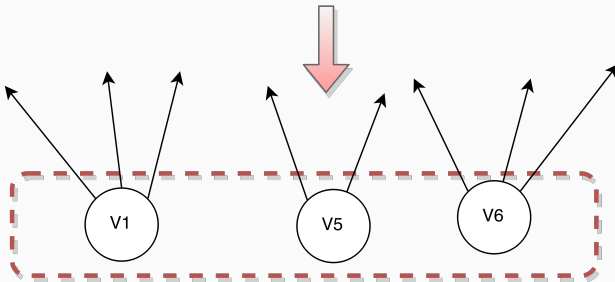


# ИСПОЛЬЗОВАНИЕ ПАРАЛЛЕЛЬНОГО ОБХОДА В ШИРИНУ

J



J + 1



# ЗАДАЧА MANY-TO-MANY

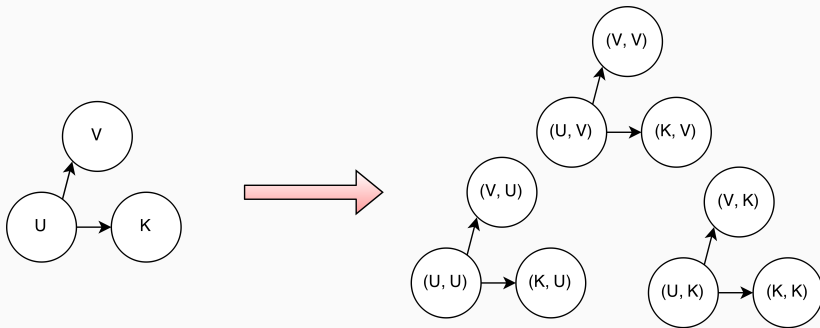
---

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

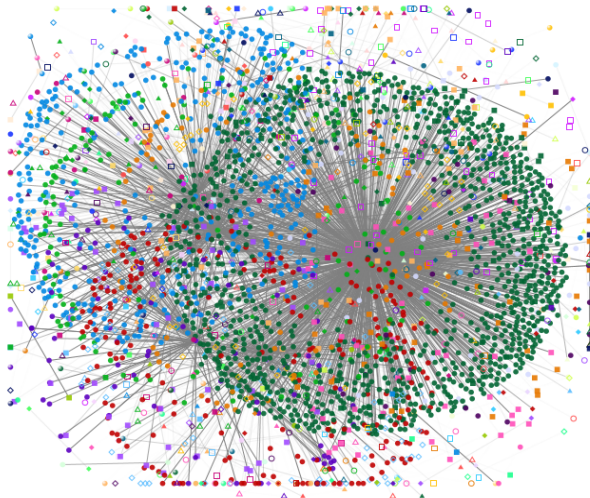
```
1: procedure ALLPAIRSPAR1(G)
2:   return HANDLEVERTICES(G, 0, |G.vertices|)
3:
4: procedure HANDLEVERTICES(G, startVertex, endVertex)
5:   if endVertex – startVertex < threshold then
6:     distances  $\leftarrow$  run Bellman-Ford for [startVertex, endVertex)
7:     return distances
8:   else
9:     midV  $\leftarrow$  (startVertex + endVertex)/2
10:    fork2(
      HANDLEVERTICES(G, startV, midV),
      HANDLEVERTICES(G, midV, endV));
```

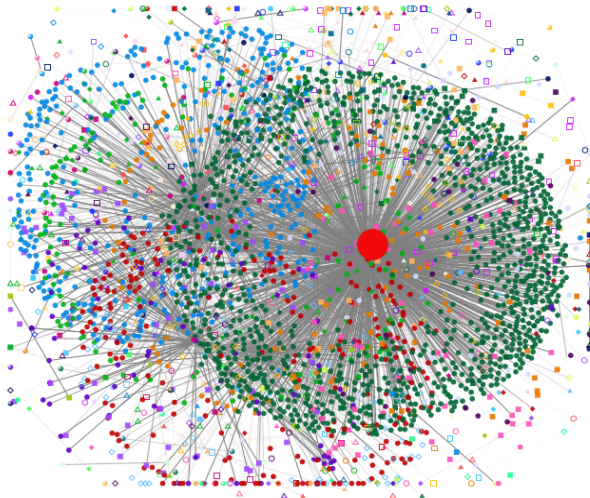


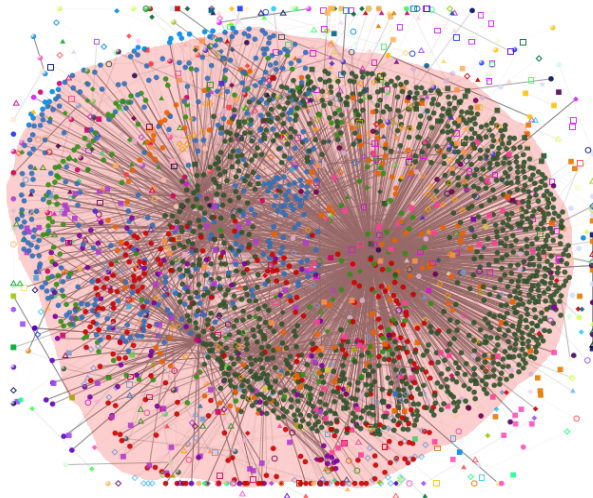
# АЛГОРИТМ ДЛЯ ОБЪЕДИНЕННОГО ГРАФА

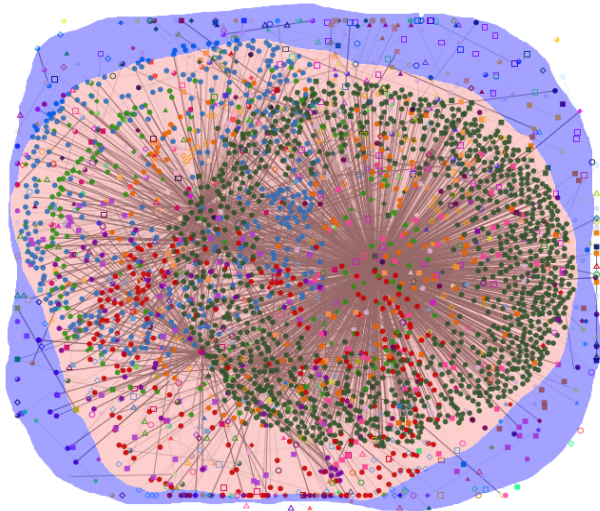


- Основан на теории "Шести рукопожатий"
- Работает не неориентированных невзвешенных социальных графах
- Использует идею динамического программирования









- Посчитаем расстояние от базовой вершины до всех других
- Каждая из вершин  $u$  может находиться только в  $[\text{dist}[\text{base}][u] - K, \text{dist}[\text{base}][u] + K]$  слоях
- Будем поддерживать две динамики  $\text{mask}$  и  $\text{calc}$ . Причем  $\text{mask}[u][i]$  - субсет вершин, расстояние от которых до  $u$  равно  $i$ . В свою очередь  $\text{calc}[u][i]$  - не более  $i$

$$\text{mask}[v][i] = \neg \text{calc}[v][i - 1] \wedge \bigvee_{\exists(u,v) \in E} \text{mask}[u][i - 1] \quad (1)$$

$$\text{calc}[v][i] = \text{calc}[v][i - 1] \vee \text{mask}[v][i] \quad (2)$$



- Пересчет расстояний для группы вершин выполняется быстрее за счет битовых операций.
- Все битовые операции выполняются без выделения дополнительной памяти.
- Нет необходимости в построении следующего фронта по предыдущему в процессе обработки
- Каждый из получившихся фронтов довольно большой, что увеличивает его способность к параллелизации
- Все остальные этапы также хорошо параллелятся

# РЕЗУЛЬТАТЫ

---

# СРАВНЕНИЕ ПАРАЛЛЕЛЬНЫХ ВЕРСИЙ БЕЛЛМАНА-ФОРДА

Algo №	Complete			BalancedTree		SquareGrid	
	TS	+	-	0.5	1	+	+-
1	2.43	4.65	nc	116.31	9.04	5.49	13.40
2	5.17	0.18	10.84	3.59	3.08	5.92	7.10
3	44.63	0.37	23.55	0.44	0.31	4.42	0.58

**Таблица:** Типичные графы

Algo №	RandomSparse			RandomDense		
	0.5+	0.5-	0.96+	0.5+	0.5-	0.96+
1	nc	nc	24.35	nc	nc	5.01
2	2.77	14.68	2.42	0.48	6.38	0.46
3	0.98	22.59	0.76	0.60	10.25	0.71

**Таблица:** Случайные графы

Алгоритм	Twitter graph
Наивная параллельная версия	427.217
Алгоритм для социальных графов	210.322

**Таблица:** Сравнение алгоритмов

# ВЫВОДЫ

---

Вопросы?