Санкт-Петербургский государственный университет

Кафедра системного программирования

Остовные деревья. Алгоритмы Прима и Борувки

Мелашенко Ксения, Ахметов Темерлан, Сыресенков Илья

Группа 22.Б11-мм

Санкт-Петербург

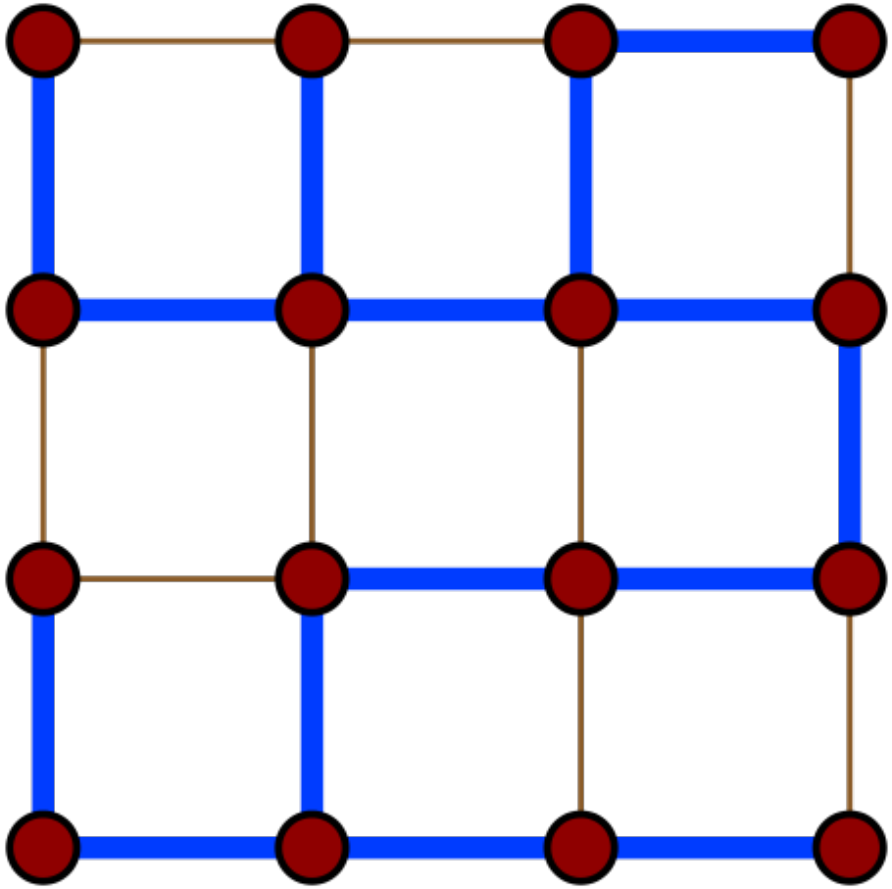
2025

Постановка задачи

Задача: дан связный, неориентированный взвешенный граф $G\,(V, E)$ и функция весов $W(u, v)$. Требуется найти **минимальное остовное дерево**.

- Остовное дерево – дерево, подграф неориентированного связного графа, с тем же числом вершин, что и у исходного графа
- Минимальное остовное дерево – остовное дерево данного графа минимального веса, где под весом дерева понимается сумма весов входящих в него ребер

Пример остовного дерева



Алгоритм Прима

- S – множество вершин, принадлежащих минимальному остовному дереву
- Минимальным ребром из S назовем такое ребро $(u, v), u \in S, v \notin S$, что $W(u, v) = min\{W(u', v'),\ u' \in S,\ v' \notin S\}$
- На каждой итерации в S добавляется вершина, инцидентная минимальному ребру из S.

Алгоритм Прима в терминах ЛА

- $A_{[N\times N]}$ – матрица смежности
  - $A[u, v] = W(u, v)$
- $s_{[1\times N]}$ – вектор для хранения уже добавленных в S вершин
  - $s[u] = \begin{cases} 0, & u \notin S \\ \infty, & u \in S \end{cases}$
- $d_{[1\times N]}$ – вектор для хранения минимальных расстояний от всех вершин до вершин из S
  - $d[u] = min\{A[u, v],\ v \in S\}$
  - $d[v] = 0,\ v \in S$

Алгоритм Прима в терминах ЛА

Принцип работы алгоритма:

1. Выбираем произвольную вершину $v \in V$, $s[v] \leftarrow \infty$, $d \leftarrow A[:, v]$
2. Пока есть вершины в $V \setminus S$:
   1. Ищем ближайшую вершину из $v' \in V \setminus S$,
   2. $s[v'] \leftarrow \infty$
   3. $d \leftarrow pairwise\_min(d, A[:, v'])$

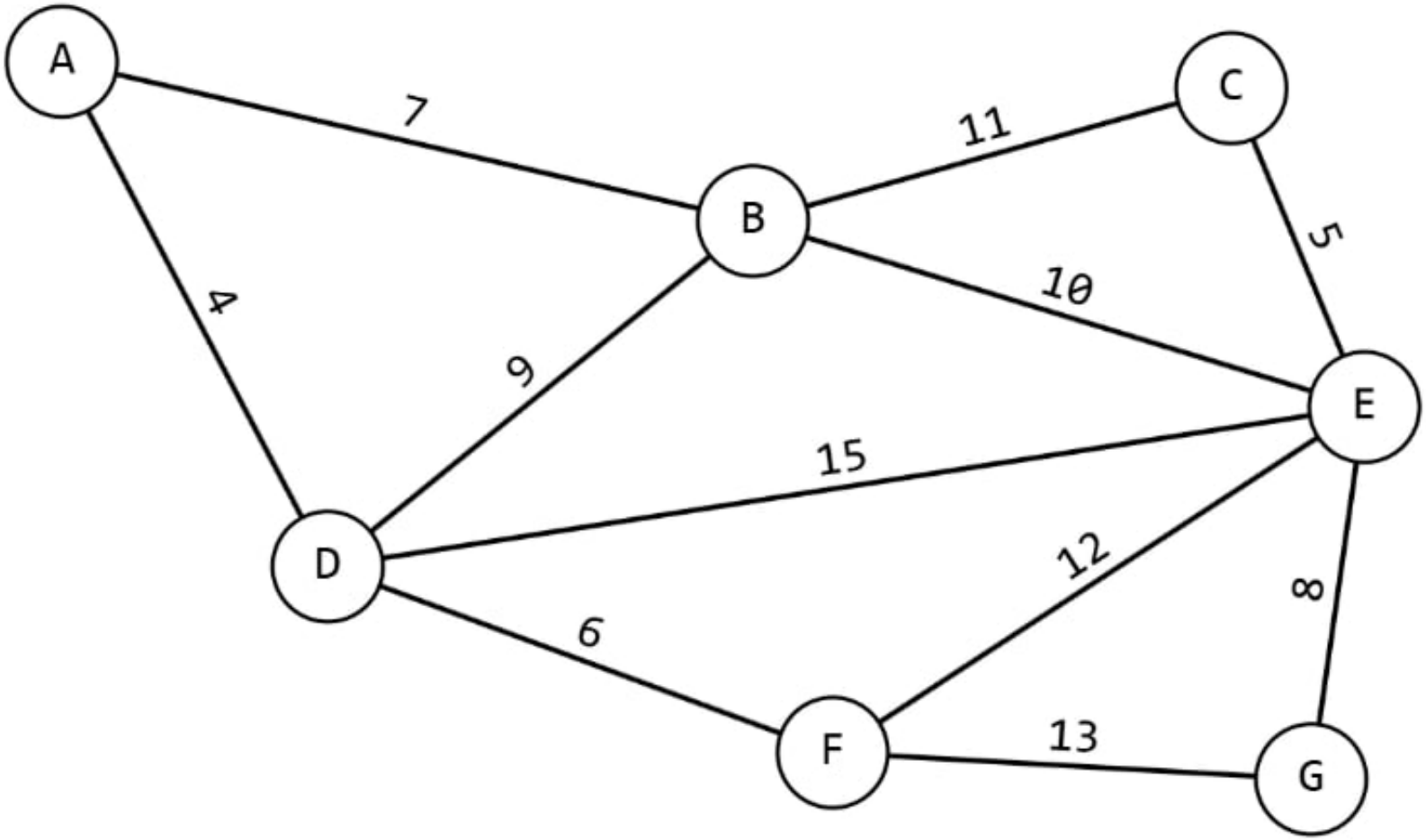
Минимальные изменения этого алгоритма позволяют сохранять ребра полученного минимального дерева и его вес.

Алгоритм Борувки

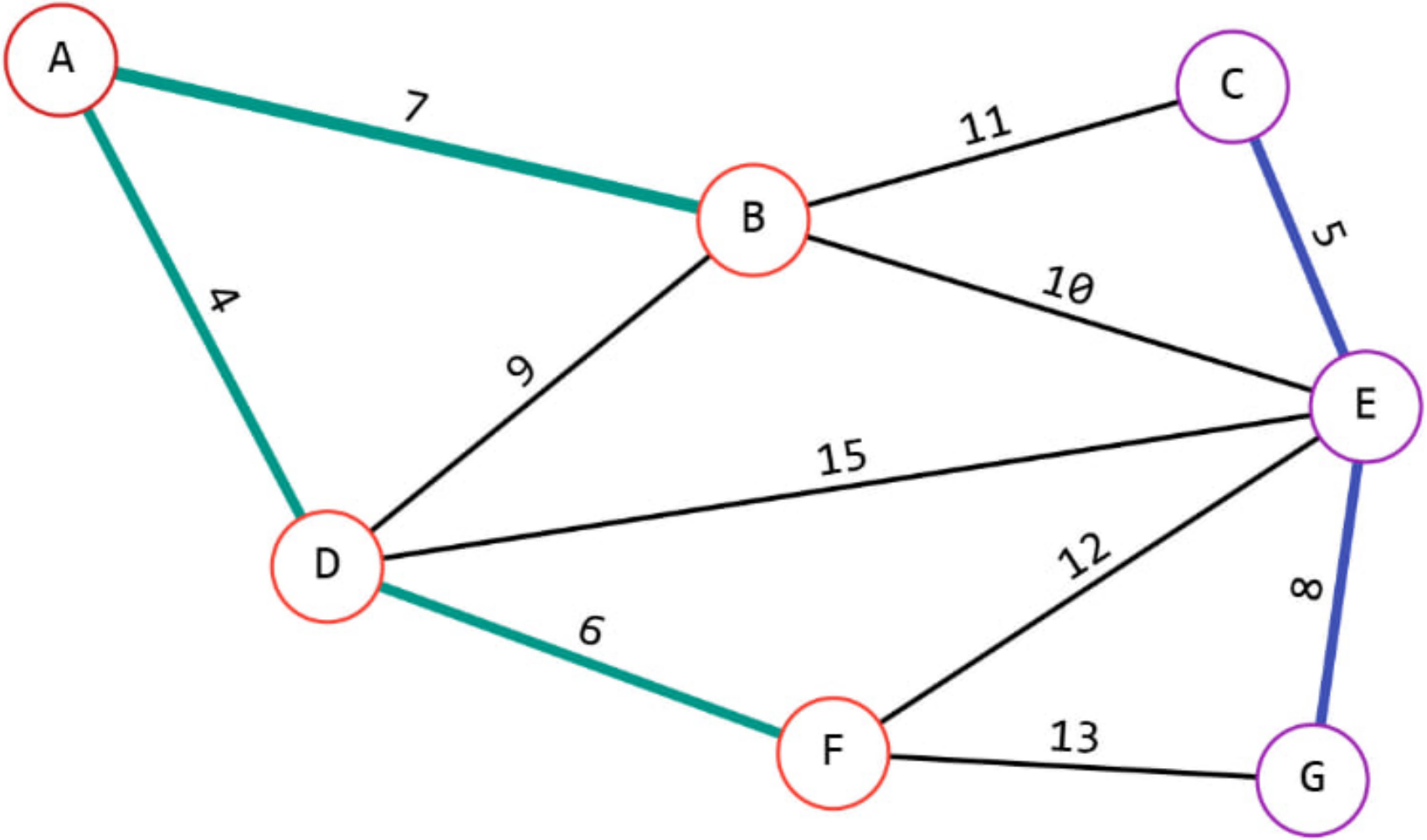
Принцип работы алгоритма:

1. Каждая вершина считается отдельной компонентой (деревом)
2. Для каждой компоненты определяется ребро минимального веса, соединяющее его с другой компонентой
   - Компоненты объединяются через выбранные ребра, что приводит к уменьшению общего количества компонент
3. Процесс повторяется, пока не останется единственный компонент, охватывающий все вершины графа.

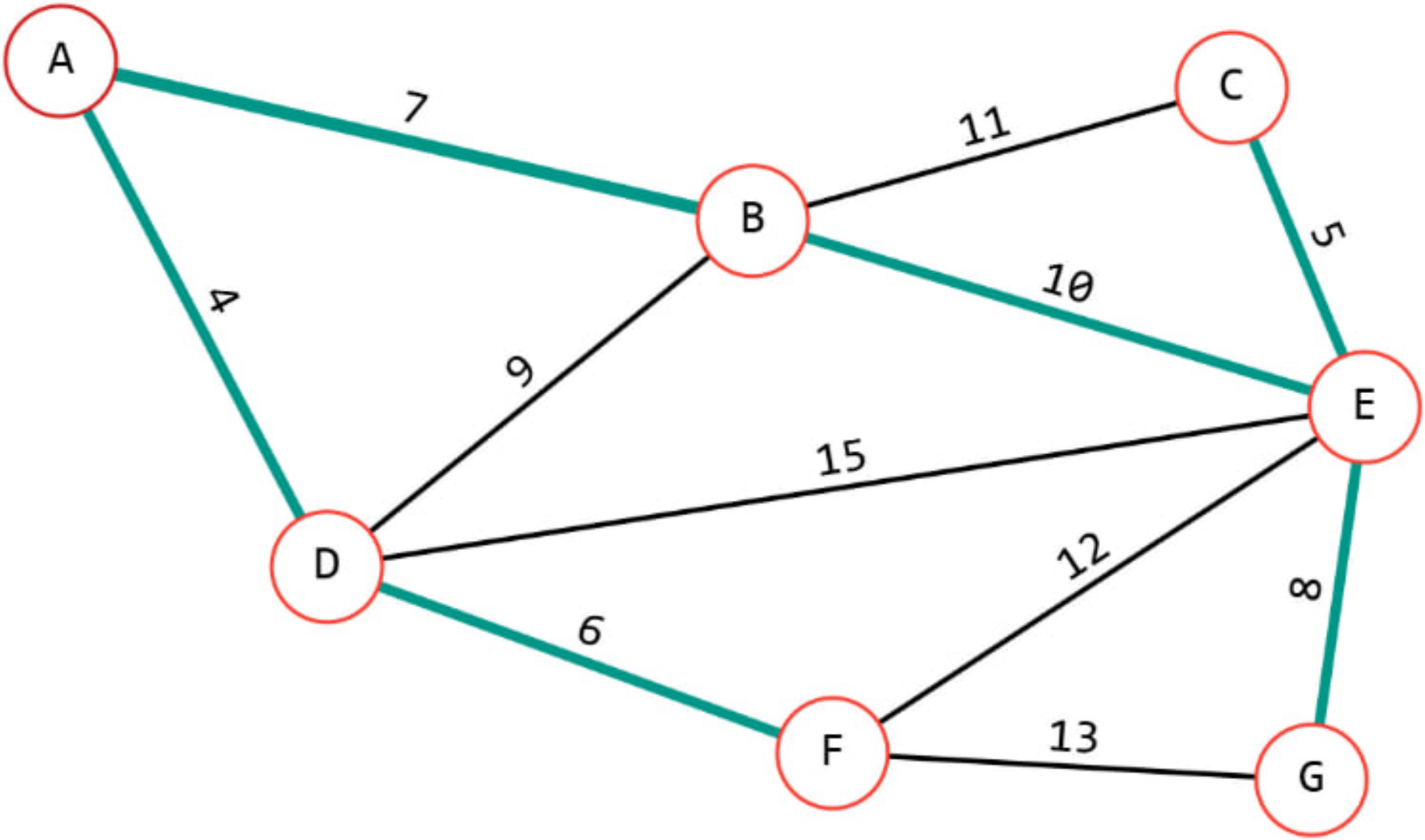
Алгоритм Борувки



Алгоритм Борувки



Алгоритм Борувки



Алгоритм Борувки в терминах ЛА

- S – матрица смежности графа
- Ребра представляем в виде (w, i), где
  - w – вес ребра
  - i – номер компоненты, с которой это ребро соединено
- $edge[v]$ – минимальное ребро, инцидентное вершине v
- $cedge[v]$ – минимальное ребро, инцидентное одному из потомков v, если потомок – корень дерева ($+\infty$ иначе)
- $parent[v]$ – корень дерева, которому принадлежит вершина v
- $comb((w_1, i_1), (w_2, i_2)) = (w_1, i_2)$
- $min(\perp, (w_2, i_2)) = (w_2, i_2)$

Алгоритм Борувки в терминах ЛА

Подсчет минимального ребра производим в полукольце $combMin = \{E \times I, \times, +\}$

- $+ := min$
- $\times := comb$

Алгоритм Борувки в терминах ЛА

- Каждую вершину представляем в виде отдельной компоненты (дерева)
- Пока $S \neq \emptyset$:
  - $edge = S \times parent$
  - $cedge[parent[i]] \leftarrow min\{cedge[parent[i]], edge[i]\}$
  - Для каждого корня берем потомка с минимальным ребром
  - Обновляем *parent*

Набор данных

Графы взяты из 9th DIMACS Implementation Challenge – Shortest Paths[1], где собраны реальные данные дорожных сетей США и синтетические данные для стресс-тестов алгоритмов.

- Прикладные
- Связные
- Взвешенные, без отрицательных весов
- Неориентированные

[1] https://www.diag.uniroma1.it/challenge9/

Набор данных

Название	**Кол-во вершин**	**Кол-во ребер**
New York City	264,346	733,846
San Francisco Bay Area	321,270	800,172
Colorado	435,666	1,057,066
Florida	1,070,376	2,712,798
Northwest USA	1,207,945	2,840,208
Northeast USA	1,524,453	3,897,636
California and Nevada	1,890,815	4,657,742
Great Lakes	2,758,119	6,885,658
Eastern USA	3,598,623	8,778,114
Western USA	6,262,104	15,248,146
Central USA	14,081,816	34,292,496
Full USA	23,947,347	58,333,344

Эксперимент

Цель: выяснить, какой из алгоритмов и библиотек лучше всего подходит для задачи поиска минимального остовного дерева.

- 3 алгоритма: Прим на `GraphBLAS`, Борувка на `GraphBLAS` и Борувка на `Google Pregel`
- 3 сравнения:
  - Реализации Прима и Борувки на `GraphBLAS`
  - Реализации Борувки на `GraphBLAS` и `Google Pregel`
  - Сравнение производительности трех описанных выше алгоритмов

Эксперимент

Характеристики вычислительной машины:

- ОС Ubuntu 24.04
- 11th Gen Intel Core i5-11400H
  - 6 ядер, 12 потоков
  - Частота до 4.5 ГГц
  - 12 Mb кеш-память 3 уровня
- 16 GB RAM
  - DDR4
  - 3200 МГц