Алгоритмы на графах

Модуль 2. Кратчайшие расстояния.

Лекция 8.

Алгоритм Прима.

Адигеев Михаил Георгиевич

2023

План лекции

1. Алгоритм Прима
   - ✓ Общая схема алгоритма
   - ✓ Реализация на основе кучи
3. Сравнение алгоритмов

Алгоритм Прима

Алгоритм Прима

Вход: связный неориентированный граф $G(V, E)$, $|V| = n, |E| = m$.

Выход: минимальное остовное дерево $G'(V', E')$, $V' = V$.

Решение будем строить итерационно. На каждом шаге текущее решение – *дерево* $G'(V', E')$. Добавляем к G' новые рёбра и вершины, пока не получим *остовное* дерево.

Алгоритм Прима

1. $G'(V', E')$: $V' = \{s\}, E' = \emptyset$.
2. Создать массивы C[1..*n*], P[1..*n*].
   - $C[s] = 0$; P[s]=*s*.
   - For each $v \in V \backslash V'$: $C[v] = w(s, v)$; $P[v] = s$
3. While $V' \neq V$:
   - Найти $v \in V \backslash V'$: для v значение $C[v]$ минимально.
   - Добавить вершину v в V'; добавить ребро $(P[v], v)$ в E'.
   - Update_C&P(v).

Алгоритм Прима

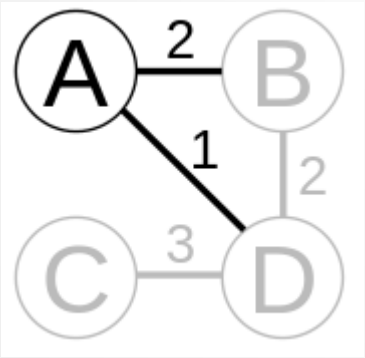
Update_C&P(v)

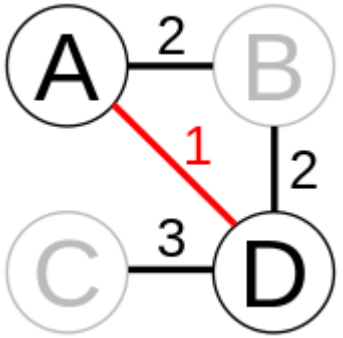
For each $(v, u) \in E$:

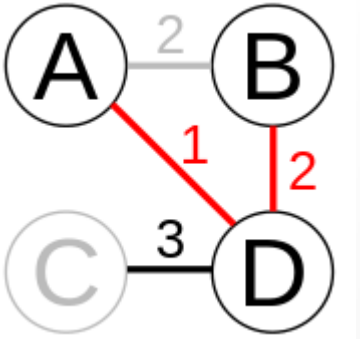
if $u \in V \backslash V'$ & $C[u] > w(v, u)$:

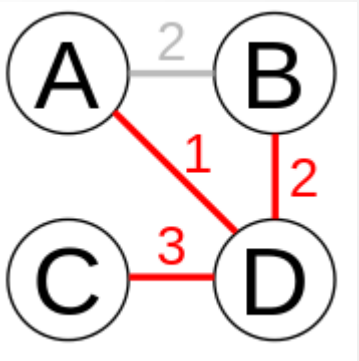
$C[u] = w(v, u)$

$P[u] = v$

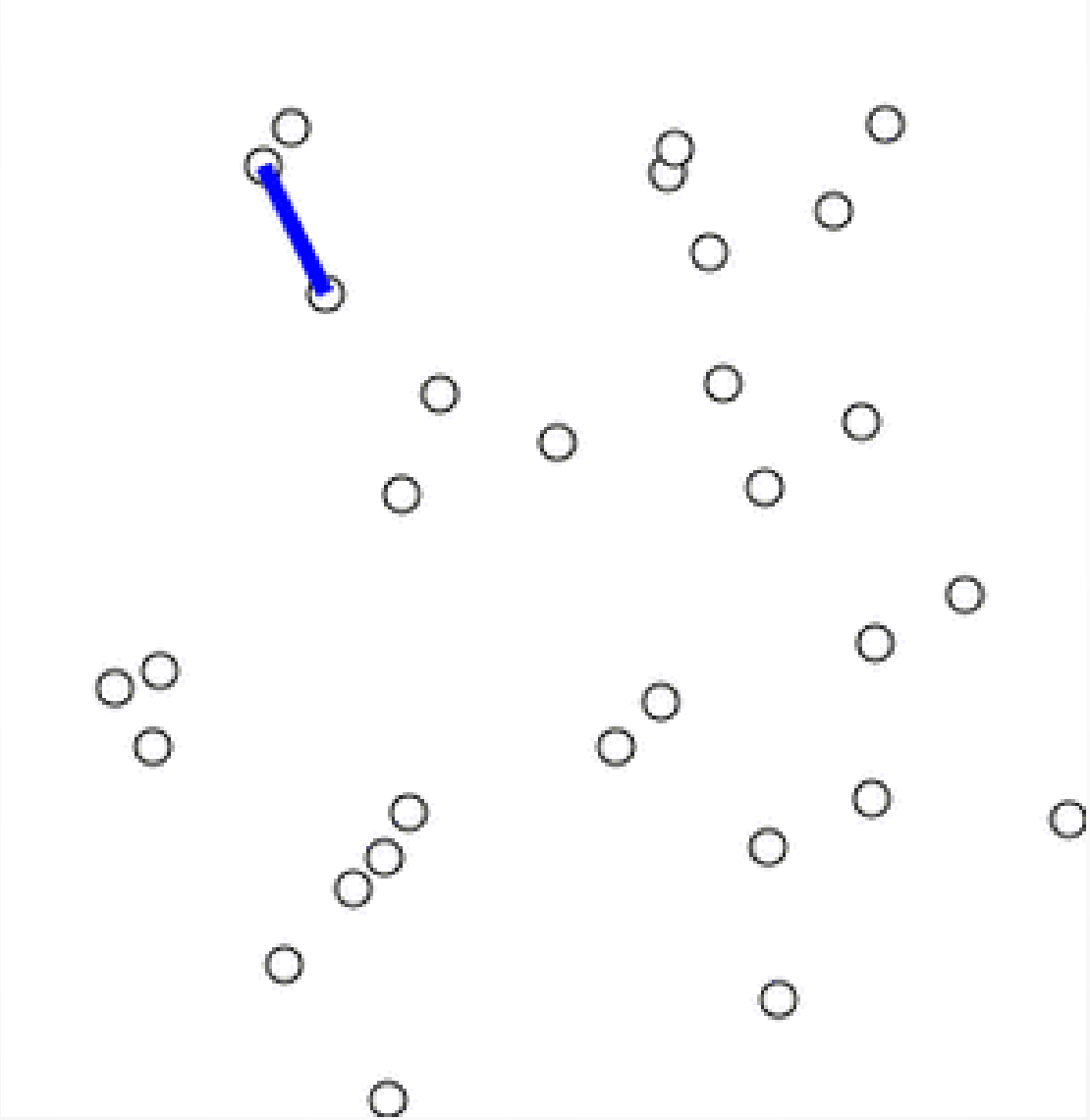








Алгоритм Прима



Алгоритм Прима

1. $G'(V', E')$: $V' = \{s\}, E' = \emptyset$.
2. Создать массивы C[1..*n*], P[1..*n*].
   - $C[s] = 0$; P[s]=*s*.
   - For each $v \in V \backslash V'$: $C[v] = w(s, v)$; $P[v] = s$
3. While $V' \neq V$: *n*-1 итераций
   - Найти $v \in V \backslash V'$: для v значение $C[v]$ минимально. ???
   - Добавить вершину v в V'; добавить ребро $(P[v], v)$ в E'. O(1)
   - Update_C&P(v). ???

Алгоритм Прима

Необходимо оценить суммарную временную сложность всех вызовов процедуры Update_C&P.

Алгоритм обновляет массивы C[] and P[] максимум один раз для каждого ребра => общее количество обновлений составляет $O(m)$.

Сложность выбора ближайшей вершины $v \in V \backslash V'$ зависит от программной реализации.

Алгоритм Прима

1) Наивная реализация: перебирать $V \backslash \mathrm{V}'$ и искать минимальное значение $C[v]$. Каждый проход требует времени $O(n) \Rightarrow$ совокупная сложность получается $O(m + n^2) = O(n^2)$.
2) Использовать *очередь с приоритетами* для хранения $C[v]$ и получения минимального значения на каждой итерации. Совокупная временная сложность зависит от реализации очереди с приоритетами:
   a) Двоичная очередь: $O(m \log n)$
   b) Очередь Фибоначчи: $O(m + n \log n)$

Алгоритм Прима

Очередь с приоритетами – абстрактная структура данных, которая позволяет эффективно выполнять операции добавления новых элементов и получения элемента с минимальным/максимальным *ключом*.

Цепной критерий оптимальности

Определение.

Пусть G' - остовное дерево графа G, ребро (u, v) принадлежит дереву G'. При удалении (u, v) из G' дерево G' распадается на 2 компоненты (п.4 из теоремы о деревьях). Через $\delta(G', (u, v))$ обозначим полученный ***разрез***, то есть множество рёбер, у которых концы лежат в разных компонентах полученного леса.

Разрезный критерий оптимальности

Теорема (Разрезный критерий оптимальности остовного дерева).

Остовное дерево $G'(V', E')$ минимально $\Leftrightarrow$ для любого *древесного* ребра $(u, v) \in E'$ и любого *недревесного* ребра $(x, y) \in \delta(G', (u, v))$ выполняется: $w(u, v) \leq w(x, y)$.

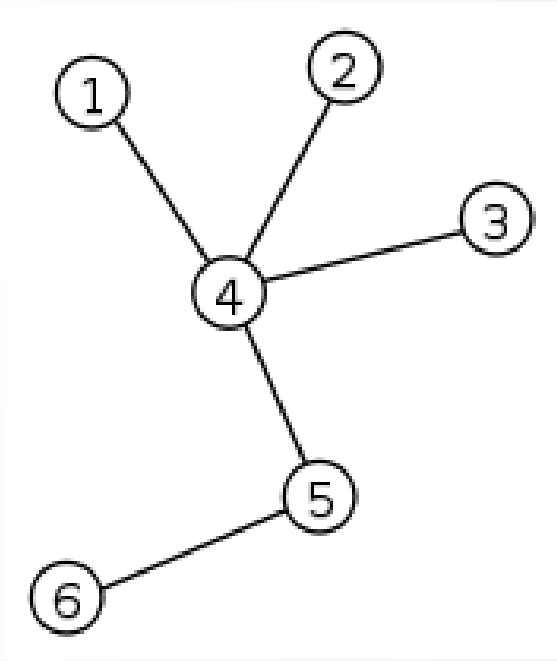
Доказательство.

Деревья

Теорема о деревьях.

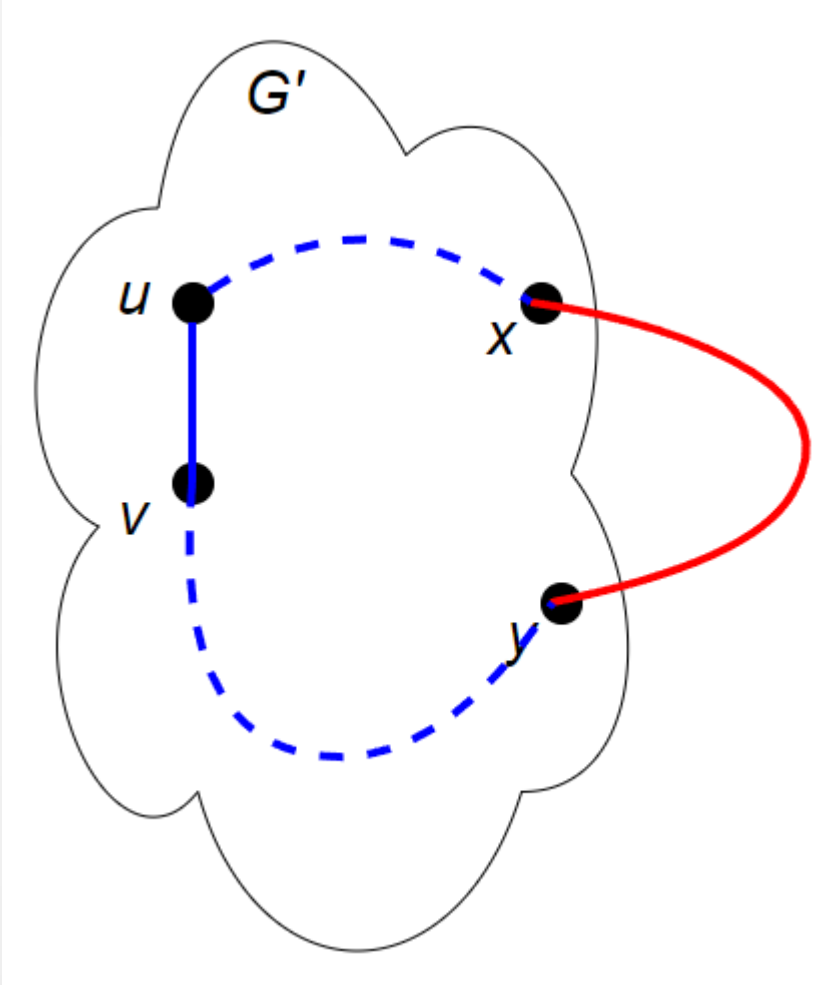
Для конечного графа $G(V, E)$ следующие утверждения эквивалентны:

1) G — дерево.
2) G — не содержит циклов и $|E| = |V| - 1$.
3) G — связен и $|E| = |V| - 1$.
4) G — связен и каждая дуга является мостом.
5) Любые две вершины можно соединить единственной простой цепью.
6) G не содержит циклов, и добавление к нему любой новой дуги приводит к образованию единственного простого цикла.



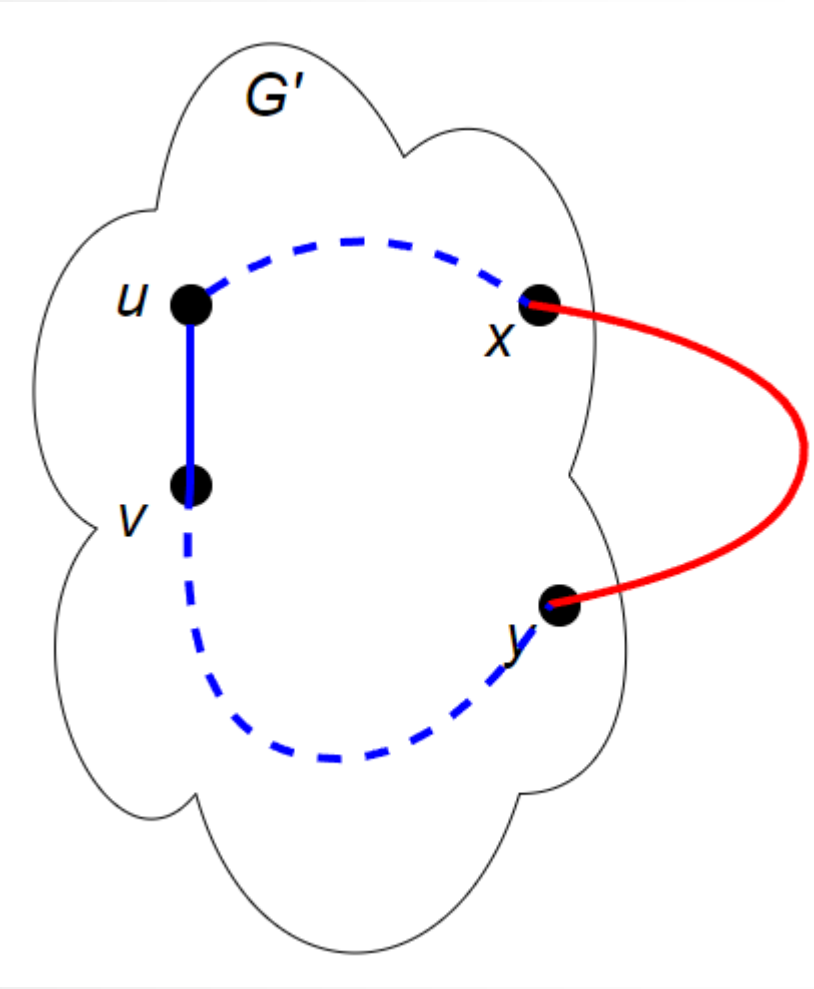
Разрезный критерий оптимальности

Доказательство.



$\Rightarrow$ Пусть G' - минимальное остовное дерево. Граф $H = (G'\backslash\{(u,v)\}) \cup \{(x,y)\}$ – тоже остовное дерево (применяем п. 4 из теоремы о деревьях). Поэтому $w(G') \leq w(H)$, и, следовательно, $w(u,v) \leq w(x,y)$.

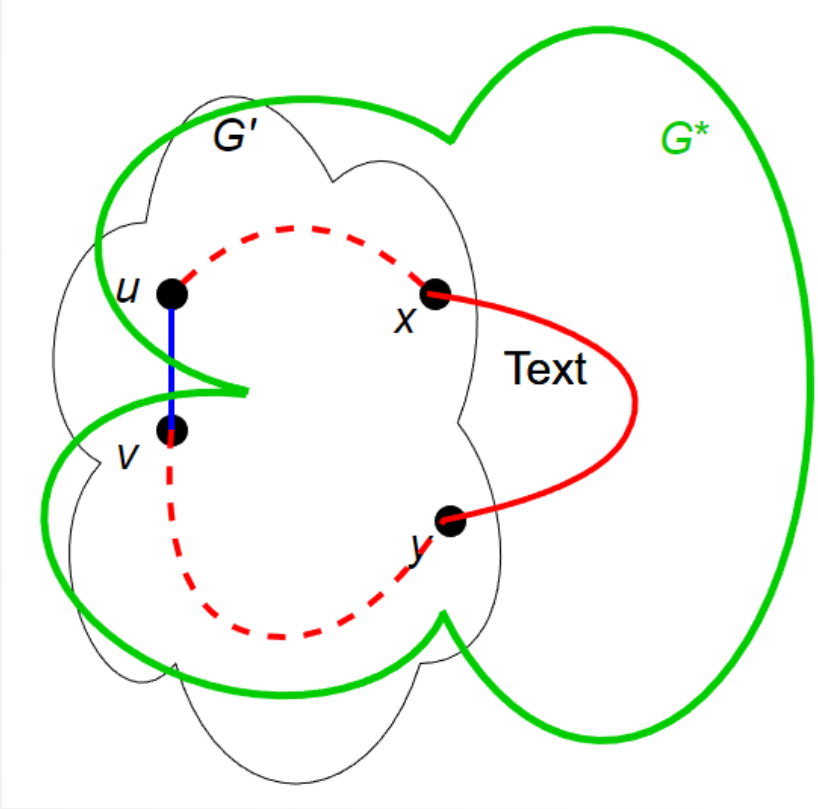
Разрезный критерий оптимальности



$\Leftarrow$ Пусть для G' выполняется условие теоремы, но G' - не минимальное остовное дерево.

Пусть G^* - то из минимальных остовных деревьев, которое имеет максимально возможное количество общих рёбер с G'. Выберем в качестве (u, v) такое ребро, которое принадлежит G' и не принадлежит $G^*((u, v) \in G' \setminus G^*)$.

Разрезный критерий оптимальности



Рассмотрим в G^* единственную цепь, соединяющую вершины u и v. На этой цепи как минимум одно ребро (x, y) принадлежит разрезу $\delta(G', (u, v))$. По условию теоремы, $w(u, v) < w(x, y)$.

Построим граф $H = (G^* \backslash \{(x, y)\} \cup \{(u, v)\}$. Это остовное дерево и $w(H) < w(G^*)$ – противоречие с тем, что G^* - минимальное остовное дерево.

Алгоритм Прима

Теорема. Алгоритм Прима строит минимальное остовное дерево.

Доказательство.

Пусть $G'(V, E')$ - результат работы алгоритма Прима. По построению алгоритма, G' - остовное дерево. Докажем, что это минимальное остовное дерево.

Покажем, что G' удовлетворяет разрезному критерию оптимальности. Пусть (u, v) - произвольное древесное ребро. Предположим, что для него нарушено условие разрезного критерия, то есть существует недревесное ребро $(x, y) \in \delta\big(G', (u, v)\big)$, для которого $w(u, v) > w(x, y)$. Но в этом случае на итерации, на которой было добавлено (u, v), вместо этого ребра должно было быть добавлено (x, y). Теорема доказана.

Сравнение алгоритмов Краскала и Прима

Алгоритм Краскала

- Сложность: $O(m \cdot \log n)$.
- Структура для хранения: список рёбер (с весами).
- Структура для работы: Union_Find.

Алгоритм Прима:

- Сложность: $O(m \cdot \log n)$.
- Структура для хранения: список смежности.
- Структура для работы: очередь с приоритетами.

Для разреженных графов эффективнее алгоритм Краскала, для плотных – алгоритм Прима (есть реализация со сложностью $O(n^2)$).

Задание 4

Задание 4

Задача «Этажи НИИ ЧаВо».

Лифт ожил! Привалов и Амперян согласились ехать на 76 этаж НИИ ЧАВО в город Тьмускорпионь. Но возникла проблема, которую надо срочно решить: упомянутым добровольцам, на всякий случай, нужны планы этажей, начиная с 76 и выше. У Привалова есть карманный компьютер, но необходимо максимально сэкономить его память для хранения этих карт.

Каждый этаж НИИ ЧАВО представляется прямоугольным полем $n \times m$. В некоторых местах на этажах расположены важные элементы инфраструктуры. В соответствующих клетках поля должны располагаться буквы латинского алфавита. Одинаковыми буквами обозначены одинаковые элементы инфраструктуры, а разными – разные. При этом заглавные и строчные буквы различаются. На карманный компьютер Привалова необходимо отправить k планов этажей последовательно. Есть два способа передать план этажа:

1. Передать целиком. Тогда план будет занимать $n \cdot m$ байт.
2. Передать отличие от одного из предшественников. Допустим, мы хотим передать уровень x, и при этом уровень y уже был передан. Предположим также, что уровень x отличается от уровня y в z клетках. Тогда количество байт для передачи этим способом будет равно $z \cdot w$, где w – некоторая константа.

Задание 4

Требуется написать программу, которая определит последовательность передачи планов этажей с наименьшими затратами памяти.

Формат входных данных

В первой строке указаны числа n, m, k, w ($1 \le n, m \le 10$, $1 \le k, w \le 1000$). Далее следуют описания этажей. Каждый этаж описывается n строками по m символов в каждой. Каждый символ это либо буква латинского алфавита, либо символ точка («.»), означающий пустую клетку на плане этажа.

Формат выходных данных

В первой строке должно быть одно целое число, означающие минимальное количество байт необходимых для хранения планов всех этажей. В следующих k строках нужно вывести описания способа передачи последовательности этажей. Каждая из этих строк состоит из двух чисел a_i и b_i. Число a_i означает номер этажа, в соответствии с порядком, определенным входными данными, план которого будет передаваться следующим. Число b_i равно 0, если указанный план будет передаваться первым способом (т.е. полностью). Если план для этажа a_i будет передаваться вторым способом, то число b_i равно порядковому номеру этажа, относительно которого будет передано отличие.

Задание 4

Пример

input.txt	output.txt
2 3 3 2 A.A ... A.a ..C X.Y ...	14 1 0 2 1 3 1
1 1 4 1 A . B .	3 1 0 2 0 4 2 3 0
1 3 5 2 ABA BBB BBA BAB ABB	11 1 0 3 1 2 3 4 2 5 1