Московский Государственный Университет

Минимальное остовное дерево

Выполнил: Курцев Д.В.
Группа: 517

Факультет Вычислительной математики и кибернетики
Кафедра Математических методов прогнозирования

Октябрь 2023

1 Постановка задачи

Имеется связный неориентированный граф $G = (V, E)$, где V – множество городов, E – множество рёбер, образованных всеми парами городов. Для каждого ребра $(u, v) \in E$ известна длина в километрах.

В графе G нужно выделить связный подграф $T = (V, E'), E' \subset E$, общая длина которого минимальна. Поскольку подграф T ациклический и связный, он является деревом. Это дерево называется минимальным остовным деревом (МОД) графа G.

Нужно разработать алгоритм и реализовать программу вычисления минимального остовного дерева для графа с вершинами в городах и оценить общую длину ребер этого дерева.

Программа должна обеспечить:

- Ввод заданной матрицы расстояний между 196 городами;
- Вывод общей длины ребер покрывающего дерева и списка ребер, входящих в него.

2 Описание данных

Матрица длин ребер представлена в виде таблицы, фрагмент которой показан ниже.

	Абакан	Альметьевск	Анапа	Ангарск	Арзамас	Армавир	Арсеньев	Артём
Абакан		3391	5229	1421	4106	4979	5280	5400
Альметьевск	3391		1958	4061	717	1708	7920	8040
Анапа	5229	1958		5899	1559	378	9758	9878
Ангарск	1421	4061	5899		4808	5682	3853	3973
Арзамас	4106	717	1559	4808		1456	8633	8752
Армавир	4979	1708	378	5682	1456		9510	9630
Арсеньев	5280	7920	9758	3853	8633	9510		219
Артём	5400	8040	9878	3973	8752	9630	219	
Архангельск	4647	1688	2707	5349	1463	2600	9209	9330
Астрахань	4784	1513	982	5487	1303	703	9347	9468
Ачинск	471	2887	4725	1174	3600	4477	5034	5155
Балаково	3838	567	1423	4541	683	1175	8401	8522
Балашиха	4450	1057	1482	5152	498	1376	9012	9133
Барнаул	1241	2501	4339	1943	3214	4091	5803	5924

Рис. 1: Граф

Исходные данные задаются в файле Table.cities.coord в формате таблицы Microsoft Excel. Расстояния между городами задаются положительными числами. Считается, что если между городами нет пути (нет ребра между вершинами), то эта длина равна 10^8, что является больше максимального веса ребра.

3 Ход работы и программная реализация

В данной работе был разобран и реализован алгоритм Прима для поиска минимального остовного дерева. Сам алгоритм имеет очень простой вид. Искомый минимальный остов строится постепенно, добавлением в него рёбер по одному. Изначально остов полагается состоящим из единственной вершины (её можно выбрать произвольно). Затем выбирается ребро минимального веса, исходящее из этой вершины, и добавляется в минимальный остов. После этого остов содержит уже две вершины, и теперь ищется и добавляется ребро минимального веса, имеющее один конец в одной из двух выбранных вершин, а другой — наоборот, во всех остальных, кроме этих двух. И так далее, т.е. всякий раз ищется минимальное по весу ребро, один конец которого — уже взятая в остов вершина, а другой конец — ещё не взятая, и это ребро добавляется в остов (если таких рёбер несколько, можно взять любое). Этот процесс повторяется до тех пор, пока остов не станет содержать все вершины (или, что то же самое, $n - 1$ ребро).

В итоге будет построено остовное дерево, являющееся минимальным. Если граф был изначально не связен, то остов найден не будет (количество выбранных рёбер останется меньше $n - 1$).

3.1 Доказательство

Пусть граф G был связным, т.е. ответ существует. Обозначим через T остовное дерево, найденное алгоритмом Прима, а через S — минимальное остовное дерево. Очевидно, что T действительно

является остовом (т.е. поддеревом графа G). Покажем, что веса S и T совпадают.

Рассмотрим первый момент времени, когда в T происходило добавление ребра, невходящего в оптимальный остов S. Обозначим это ребро через e, концы его — через a и b, а множество входящих на тот момент в остов вершин — через V (согласно алгоритму, $a \in V, b \notin V$, либо наоборот). В оптимальном остове S вершины a и b соединяются каким-то путём P; найдём в этом пути любое ребро g, один конец которого лежит в V, а другой — нет. Поскольку алгоритм Прима выбрал ребро e вместо ребра g, то это значит, что вес ребра g больше либо равен весу ребра e.

Удалим теперь из S ребро g, и добавим ребро e. По только что сказанному, вес остова в результате не мог увеличиться (уменьшиться он тоже не мог, поскольку S было оптимальным). Кроме того, S не перестало быть остовом (в том, что связность не нарушилась, нетрудно убедиться: мы замкнули путь P в цикл, и потом удалили из этого цикла одно ребро).

Итак, мы показали, что можно выбрать оптимальный остов S таким образом, что он будет включать ребро e. Повторяя эту процедуру необходимое число раз, мы получаем, что можно выбрать оптимальный остов S так, чтобы он совпадал с T. Следовательно, вес построенного алгоритмом Прима T минимален, что и требовалось доказать.

Ниже на рисунке представлен поэтапный пример работы данного алгоритма.

3.2 Оценка сложности

Если искать каждый раз ребро простым просмотром среди всех возможных вариантов, то асимптотически будет требоваться просмотр $O(m)$ рёбер, чтобы найти среди всех допустимых ребро с наименьшим весом. Суммарная асимптотика алгоритма составит в таком случае $O(nm)$, что в худшем случае есть $O(n^3)$, — слишком медленный алгоритм.

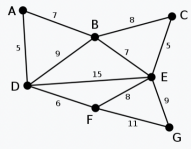
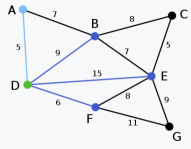
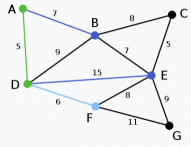
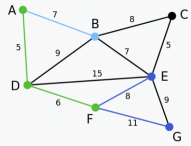
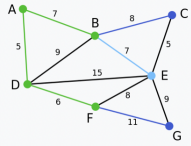
Изображение	Множество выбранных вершин U	Ребро (u, v)	Множество невыбранных вершин V \ U	Описание
	{}		{A,B,C,D,E,F,G}	Исходный взвешенный граф. Числа возле ребер показывают их веса, которые можно рассматривать как расстояния между вершинами.
	{D}	(D,A) = 5 **V** (D,B) = 9 (D,E) = 15 (D,F) = 6	{A,B,C,E,F,G}	В качестве начальной произвольно выбирается вершина **D**. Каждая из вершин **A**, **B**, **E** и **F** соединена с **D** единственным ребром. Вершина **A** — ближайшая к **D**, и выбирается как вторая вершина вместе с ребром **AD**.
	{A,D}	(D,B) = 9 (D,E) = 15 (D,F) = 6 **V** (A,B) = 7	{B,C,E,F,G}	Следующая вершина — ближайшая к любой из выбранных вершин **D** или **A**. **B** удалена от **D** на 9 и от **A** — на 7. Расстояние до **E** равно 15, а до **F** — 6. **F** является ближайшей вершиной, поэтому она включается в дерево **F** вместе с ребром **DF**.
	{A,D,F}	(D,B) = 9 (D,E) = 15 (A,B) = 7 **V** (F,E) = 8 (F,G) = 11	{B,C,E,G}	Аналогичным образом выбирается вершина **B**, удаленная от **A** на 7.
	{A,B,D,F}	(B,C) = 8 (B,E) = 7 **V** (D,B) = 9 цикл (D,E) = 15 (F,E) = 8 (F,G) = 11	{C,E,G}	В этом случае есть возможность выбрать либо **C**, либо **E**, либо **G**. **C** удалена от **B** на 8, **E** удалена от **B** на 7, а **G** удалена от **F** на 11. **E** — ближайшая вершина, поэтому выбирается **E** и ребро **BE**.

Рис. 2: Алгоритм Прима

Этот алгоритм можно улучшить, если просматривать каждый раз не все рёбра, а только по одному ребру из каждой уже выбранной вершины. Для этого, например, можно отсортировать рёбра из каждой вершины в порядке возрастания весов, и хранить указатель на первое допустимое ребро (напомним, допустимы только те рёбра, которые ведут в множество ещё не выбранных вершин). Тогда, если пересчитывать эти указатели при каждом добавлении ребра в остов, суммарная асимптотика алгоритма будет $O(n^2+m)$, но предварительно потребуется выполнить сортировку всех рёбер за $O(m \log n)$, что в худшем случае (для плотных графов) даёт асимптотику $O(n^2 \log n)$.

3.3 Реализация

Код данного алгоритма представлен в юпитер ноутбуке. Считается, что файл с данными (Table.cities.coord.xls) лежит в той же директории, что данный ноутбук. Был создан класс для поиска ми-

нимального остовного дерева MinimumSpanningTree, который на вход принимает граф - список рёбер и матрицу смежности с расстояниями между городами (веса рёбер). В функции prim реализован алгоритм Прима для поиска минимального остова. Данная функция на выходе выдаёт суммарную длину рёбер МОД и список рёбер МОД в виде пар в формате «город (номер) – город (номер)»

4 Эксперименты

Был проведён эксперимент на данных, представленных в задании. Представленный алгоритм работал около 2 секунд и нашёл минимальное остовное дерево длиной 39089.0. Список рёбер представлен ниже на рисунках.

```
Абакан (0) – Минусинск (92)
Минусинск (92) – Кызыл (82)
Абакан (0) – Красноярск (76)
Красноярск (76) – Железногорск (46)
Красноярск (76) – Ачинск (10)
Красноярск (76) – Канск (61)
Ачинск (10) – Кемерово (62)
Кемерово (62) – Ленинск–Кузнецкий (83)
Ленинск–Кузнецкий (83) – Киселёвск (65)
Киселёвск (65) – Прокопьевск (134)
Прокопьевск (134) – Новокузнецк (110)
Новокузнецк (110) – Междуреченск (90)
Кемерово (62) – Томск (171)
Томск (171) – Северск (152)
Ленинск–Кузнецкий (83) – Новосибирск (114)
Новосибирск (114) – Бердск (17)
Бердск (17) – Барнаул (13)
Барнаул (13) – Бийск (19)
Барнаул (13) – Рубцовск (140)
Новосибирск (114) – Омск (124)
Омск (124) – Ишим (54)
Ишим (54) – Тюмень (173)
Тюмень (173) – Курган (80)
Тюмень (173) – Тобольск (169)
Курган (80) – Каменск–Уральский (59)
Каменск–Уральский (59) – Екатеринбург (43)
Екатеринбург (43) – Первоуральск (130)
Екатеринбург (43) – Нижний Тагил (109)
Каменск–Уральский (59) – Челябинск (186)
Челябинск (186) – Миасс (91)
Миасс (91) – Златоуст (50)
Нижний Тагил (109) – Серов (154)
Первоуральск (130) – Кунгур (79)
Кунгур (79) – Пермь (131)
Пермь (131) – Березники (18)
Березники (18) – Соликамск (158)
Березники (18) – Кудымкар (77)
Пермь (131) – Воткинск (36)
Воткинск (36) – Чайковский (184)
Воткинск (36) – Ижевск (52)
Ижевск (52) – Сарапул (148)
Сарапул (148) – Нефтекамск (104)
Ижевск (52) – Глазов (37)
Нефтекамск (104) – Набережные Челны (97)
Набережные Челны (97) – Нижнекамск (107)
Нижнекамск (107) – Альметьевск (1)
Альметьевск (1) – Октябрьский (123)
Октябрьский (123) – Уфа (178)
Уфа (178) – Стерлитамак (162)
```

Граф

```
Назрань (98) – Грозный (38)
Грозный (38) – Хасавюрт (182)
Хасавюрт (182) – Махачкала (89)
Махачкала (89) – Дербент (39)
Краснодар (75) – Новороссийск (113)
Новороссийск (113) – Анапа (2)
Анапа (2) – Керчь (63)
Керчь (63) – Симферополь (156)
Симферополь (156) – Севастополь (150)
Симферополь (156) – Евпатория (42)
Майкоп (88) – Сочи (159)
Киров (64) – Сыктывкар (165)
Колпино (70) – Петрозаводск (132)
Петрозаводск (132) – Беломорск (16)
Тобольск (169) – Ханты–Мансийск (180)
Ханты–Мансийск (180) – Нефтеюганск (105)
Нефтеюганск (105) – Сургут (163)
Сургут (163) – Нижневартовск (106)
Сургут (163) – Ноябрьск (120)
Ноябрьск (120) – Новый Уренгой (118)
Беломорск (16) – Мурманск (94)
Канск (61) – Братск (22)
Братск (22) – Ангарск (3)
Ангарск (3) – Иркутск (53)
Иркутск (53) – Улан–Удэ (175)
Улан–Удэ (175) – Чита (188)
Псков (135) – Калининград (57)
Вологда (32) – Архангельск (8)
Архангельск (8) – Северодвинск (151)
Сыктывкар (165) – Воркута (33)
Воркута (33) – Салехард (144)
Воркута (33) – Нарьян–Мар (101)
Чита (188) – Благовещенск (21)
Благовещенск (21) – Биробиджан (20)
Биробиджан (20) – Хабаровск (179)
Хабаровск (179) – Комсомольск–на–Амуре (71)
Хабаровск (179) – Арсеньев (6)
Арсеньев (6) – Уссурийск (177)
Уссурийск (177) – Артём (7)
Артём (7) – Владивосток (26)
Артём (7) – Находка (102)
Комсомольск–на–Амуре (71) – Южно–Сахалинск (193)
Благовещенск (21) – Якутск (194)
Якутск (194) – Магадан (86)
```

Граф

```
Стерлитамак (162) – Салават (143)
Салават (143) – Оренбург (126)
Глазов (37) – Киров (64)
Нижнекамск (107) – Казань (56)
Казань (56) – Йошкар-Ола (55)
Йошкар-Ола (55) – Новочебоксарск (115)
Новочебоксарск (115) – Чебоксары (185)
Казань (56) – Ульяновск (176)
Ульяновск (176) – Димитровград (41)
Димитровград (41) – Тольятти (170)
Тольятти (170) – Самара (145)
Самара (145) – Новокуйбышевск (111)
Тольятти (170) – Сызрань (164)
Сызрань (164) – Кузнецк (78)
Кузнецк (78) – Пенза (129)
Пенза (129) – Саранск (147)
Сызрань (164) – Балаково (11)
Саранск (147) – Арзамас (4)
Арзамас (4) – Нижний Новгород (108)
Нижний Новгород (108) – Дзержинск (40)
Дзержинск (40) – Муром (95)
Муром (95) – Ковров (68)
Ковров (68) – Владимир (28)
Ковров (68) – Иваново (51)
Иваново (51) – Кострома (73)
Кострома (73) – Ярославль (195)
Ярославль (195) – Рыбинск (141)
Владимир (28) – Орехово-Зуево (127)
Орехово-Зуево (127) – Ногинск (119)
Ногинск (119) – Электросталь (191)
Электросталь (191) – Раменское (138)
Раменское (138) – Жуковский (48)
Жуковский (48) – Люберцы (85)
Люберцы (85) – Железнодорожный (47)
Железнодорожный (47) – Балашиха (12)
Балашиха (12) – Щёлково (190)
Щёлково (190) – Королёв (72)
Королёв (72) – Мытищи (96)
Королёв (72) – Пушкино (136)
Мытищи (96) – Москва (93)
Москва (93) – Химки (183)
Химки (183) – Красногорск (74)
Химки (183) – Зеленоград (49)
Красногорск (74) – Одинцово (122)
Пушкино (136) – Сергиев Посад (153)
Москва (93) – Подольск (133)
Раменское (138) – Воскресенск (35)
Воскресенск (35) – Коломна (69)
Зеленоград (49) – Клин (67)
Подольск (133) – Серпухов (155)
Серпухов (155) – Обнинск (121)
```

Граф

```
Обнинск (121) – Калуга (58)
Клин (67) – Тверь (168)
Коломна (69) – Рязань (142)
Серпухов (155) – Тула (172)
Тула (172) – Узловая (174)
Узловая (174) – Новомосковск (112)
Узловая (174) – Елец (44)
Елец (44) – Липецк (84)
Липецк (84) – Воронеж (34)
Воронеж (34) – Старый Оскол (161)
Липецк (84) – Тамбов (167)
Старый Оскол (161) – Белгород (15)
Белгород (15) – Курск (81)
Курск (81) – Орёл (125)
Орёл (125) – Брянск (23)
Балаково (11) – Саратов (149)
Рыбинск (141) – Череповец (187)
Череповец (187) – Вологда (32)
Саратов (149) – Камышин (60)
Камышин (60) – Волжский (31)
Волжский (31) – Волгоград (29)
Брянск (23) – Смоленск (157)
Смоленск (157) – Великие Луки (24)
Миасс (91) – Магнитогорск (87)
Великие Луки (24) – Псков (135)
Псков (135) – Великий Новгород (25)
Великий Новгород (25) – Колпино (70)
Колпино (70) – Санкт-Петербург (146)
Волжский (31) – Харабали (181)
Харабали (181) – Астрахань (9)
Оренбург (126) – Орск (128)
Волгоград (29) – Элиста (192)
Элиста (192) – Волгодонск (30)
Волгодонск (30) – Шахты (189)
Шахты (189) – Новошахтинск (117)
Шахты (189) – Новочеркасск (116)
Новочеркасск (116) – Ростов-на-Дону (139)
Ростов-на-Дону (139) – Батайск (14)
Ростов-на-Дону (139) – Таганрог (166)
Батайск (14) – Краснодар (75)
Краснодар (75) – Майкоп (88)
Майкоп (88) – Армавир (5)
Армавир (5) – Невинномысск (103)
Невинномысск (103) – Ставрополь (160)
Невинномысск (103) – Пятигорск (137)
Пятигорск (137) – Ессентуки (45)
Ессентуки (45) – Кисловодск (66)
Пятигорск (137) – Нальчик (99)
Нальчик (99) – Нарткала (100)
Нарткала (100) – Назрань (98)
Назрань (98) – Владикавказ (27)
```

Граф

5 Выводы

В данной работе был разобран и реализован алгоритма Прима для поиска минимального остовного дерева. Его работа была проверена для задачи поиска пути между всеми городами. Представленный алгоритм, смог хорошо и довольно быстро решить данную задачу.

6 Литература

[1]: https://ru.wikipedia.org/wiki/РθРњРγР«СЃРчСЂРё_Р§СЃРчРёРө