Алгоритм Краскала

1)**Алгоритм Краскала**(или алгоритм Крускала) - алгоритм построения минимального остовного дерева взвешенного связного неориентированного графа. Алгоритм впервые описан Джозефом Крускалом в 1956 году.

Его особенность заключается в том, что алгоритм способен строить одновременно несколько деревьев для разных компонент связности, которые впоследствии объединяются в одно.

2) **Принцип работы**

Вначале текущее множество рёбер устанавливается пустым. Затем, пока это возможно, проводится следующая операция: из всех рёбер, добавление которых к уже имеющемуся множеству не вызовет появление в нём цикла, выбирается ребро минимального веса и добавляется к уже имеющемуся множеству. Когда таких рёбер больше нет, алгоритм завершён. Подграф данного графа, содержащий все его вершины и найденное множество рёбер, является его остовным деревом минимального веса.

Алгоритм состоит из следующей последовательности действий:

* 1. Создается список ребер E, содержащий длину ребра, номер исходной вершины ребра (i), номер конечной вершины ребра (j), признак включения данного ребра в дерево.
* 2. Данный список упорядочивается в порядке возрастания длин ребер.
* 3. Просматривается список E и выбирается из него ребро с максимальной длиной, еще не включенное в результирующее дерево и не образующее цикла с уже построенными ребрами.
* 4. Если все вершины включены в дерево и количество ребер на единицу меньше количества вершин, то алгоритм свою работу закончил. В противном случае осуществляется возврат к пункту 3.

3) **Оценка временной сложности**

Сложность этого алгоритма совпадает со сложностью используемоего алгоритма сортировки, поскольку число операций в цикле while линейно по числу ребер. Поэтому сложность алгоритма Краскала поиска МОД равна *E\*log Е.*

**5)Графики**

6) **Вывод**

Алгоритм Краскала является одним из оптимальных способов нахождения минимального остовного дерева. Сложность алгоритма совпадает со сложностью сортировки, что намного упрощает его. Так же алгоритм может одновременно строить деревья для нескольких компонент связности которые в процессе объединяются в одно.

Задача о нахождении минимального остовного дерева часто встречается в подобной постановке: есть *n* городов, через которые можно проложить маршрут так, чтобы можно было добраться из любого города в любой другой (напрямую или через другие города). Требуется найти такой маршрут, чтобы стоимость проезда была максимальной. На практике алгоритм Краскала применятся в работе авиалиний при нахождении наименьшего воздушного пути из одного пункта назначения в другой.

7) Список использованной литературы

<https://ru.wikipedia.org/wiki/Алгоритм_Краскала>

<https://knowledge.allbest.ru/programming/3c0b65635b2bd68b4c53b89521306d36_0.html>

<https://revolution.allbest.ru/programming/d00542656.html>

[Дискретная математика, 2006](https://ru.wikipedia.org/wiki/%D0%90%D0%BB%D0%B3%D0%BE%D1%80%D0%B8%D1%82%D0%BC_%D0%9A%D1%80%D0%B0%D1%81%D0%BA%D0%B0%D0%BB%D0%B0#CITEREFДискретная_математика2006)

8) **Код алгоритма**

Класс Edge

// Класс ребра  
public class Edge implements Comparable<Edge> {  
 int u ;  
 int v ;  
 int w ;  
  
 Edge(int u, int v, int w) {  
 this.u = u;  
 this.v = v;  
 this.w = w;  
 }  
  
 @Override  
 public int compareTo(Edge o) {  
 if (w != o.w) return Integer.*compare*(o.w,w);  
 return 0;  
 }  
}

Класс DSF

// Класс СНМ  
public class DSF {  
 int[] set; // номер множества  
 int[] rnk; // ранг  
  
 DSF(int size) {  
 set = new int[size];  
 rnk = new int[size];  
 for (int i = 0; i < size; i++)  
 set[i] = i;  
 }  
  
 // Возвращает множество, которому принадлежит x  
 int set(int x) {  
 return x == set[x] ? x : (set[x] = set(set[x]));  
 }  
  
 // Если u и v лежат в разных множествах, то сливаем их и возвращаем true  
 boolean union(int u, int v) {  
 if ((u = set(u)) == (v = set(v)))  
 return false;  
 if (rnk[u] < rnk[v]) {  
 set[u] = v;  
 } else {  
 set[v] = u;  
 if (rnk[u] == rnk[v])  
 rnk[u]++;  
 }  
 return true;  
 }  
}

Класс Main

import java.io.File;  
import java.io.FileNotFoundException;  
import java.io.FileOutputStream;  
import java.io.PrintWriter;  
import java.util.Arrays;  
import java.util.Collections;  
import java.util.LinkedList;  
import java.util.Scanner;  
  
public class Main {

public static Edge[] addEdges(String[] s) throws FileNotFoundException {  
 Edge[] edges = new Edge[s.length];  
 for (int i = 0; i < s.length; i++) {  
 String[] k = s[i].trim().split("\t");  
 edges[i] = new Edge(Integer.*parseInt*(k[0]), Integer.*parseInt*(k[1]), Integer.*parseInt*(k[2]));  
 }  
 return edges;  
 }  
  
 public static LinkedList<Edge> addEdgesList(String[] s){  
 LinkedList<Edge> edges = new LinkedList<>();  
 int i=0;  
 while( i< s.length){  
 String[] k = s[i].trim().split("\t");  
 edges.add(new Edge(Integer.*parseInt*(k[0]),Integer.*parseInt*(k[1]),Integer.*parseInt*(k[2])));  
 i++;  
 }  
 return edges;  
 }  
  
 public static void mstKruskalaArray(Edge[] edges, int size) throws FileNotFoundException {  
 DSF dsf = new DSF(size); // СНМ  
 long o = System.*nanoTime*( );  
 Arrays.*sort*(edges); // Сортируем ребра  
 int ret = 0; // результат  
 int count = 0;  
 for (Edge e : edges) { // перебираем ребра в порядке возрастания  
 count++;  
 if (dsf.union(e.u, e.v)) {// если ребра принадлежат разным компонентам  
 ret += e.w; // добавляем вес ребра к стоимости MST  
 }  
 }  
 long p = System.*nanoTime*();  
 PrintWriter pw = new PrintWriter(new FileOutputStream(new File("files/OutputArray.txt"),true));  
 pw.println(p-o);  
 pw.close();  
 }  
  
 public static void mstKruskalaList(LinkedList<Edge> edges, int size) throws FileNotFoundException {  
 DSF dsf = new DSF(size); // СНМ  
 long o = System.*nanoTime*();  
 Collections.*sort*(edges); // Сортируем ребра  
 int ret = 0; // результат  
 int count = 0;  
 for (Edge e : edges) { // перебираем ребра в порядке возрастания  
 count++;  
 if (dsf.union(e.u, e.v)) {// если ребра принадлежат разным компонентам  
 ret += e.w; // добавляем вес ребра к стоимости MST  
 }  
 }  
 long p = System.*nanoTime*();  
 PrintWriter pw = new PrintWriter(new FileOutputStream(new File("files/OutputList.txt"),true));  
 pw.println(p-o);  
 pw.close();  
 }  
  
 public static void main(String[] args) throws FileNotFoundException {  
 int size = *makeInput*();  
 Scanner scans = new Scanner(new File("files/Input.txt"));  
 while (scans.hasNextLine()) {  
 String line = scans.nextLine();  
 String[] s = line.trim().split(",");  
 *mstKruskalaArray*(*addEdges*(s), size);  
 }  
 Scanner scanq = new Scanner(new File("files/Input.txt"));  
 while (scanq.hasNextLine()){  
 String line2 = scanq.nextLine();  
 String[] s2 = line2.trim().split(",");  
 *mstKruskalaList*(*addEdgesList*(s2),size);  
 }  
 }  
}

9) **Код генерации входных файлов для сортировки**

public static int makeInput() throws FileNotFoundException {  
 PrintWriter pw = new PrintWriter(new File("files/Input.txt"));  
 int l = (int) (50 + Math.*random*() \* 50);  
 System.*out*.println(l);  
 System.*out*.println("");  
 int k = (int) ( 100 + Math.*random*());  
 for (int i = 0; i < l; i++) {  
 System.*out*.println(k );  
 for (int j = 0; j < k; j += 1) {  
 pw.print((int) (1 + Math.*random*() \* 24));  
 pw.print("\t");  
 pw.print((int) (1 + Math.*random*() \* 24));  
 pw.print("\t");  
 pw.print((int) (50 + Math.*random*() \* 50));  
 pw.print(",");  
 }  
 k+=100;  
 pw.println("");  
 }  
 pw.close();  
 return 10001;  
}

4) **Таблицы полученных значений количества итераций и времени работы в зависимости от размера и характера данных**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Количество элементов | Время(массив) | Время(лист) | Итерации(массив) | Итерации(лист) | Сложность |
| 100 | 338667 | 293333 | 561 | 561 | 461 |
| 200 | 295110 | 392000 | 1260 | 1260 | 1060 |
| 300 | 372888 | 234666 | 2011 | 2011 | 1711 |
| 400 | 303111 | 241777 | 2797 | 2797 | 2397 |
| 500 | 296444 | 307111 | 3607 | 3607 | 3107 |
| 600 | 671110 | 368444 | 4438 | 4438 | 3838 |
| 700 | 532000 | 696444 | 5286 | 5286 | 4586 |
| 800 | 387555 | 569777 | 6148 | 6148 | 5348 |
| 900 | 444000 | 944888 | 7022 | 7022 | 6122 |
| 1000 | 782221 | 713776 | 7908 | 7908 | 6908 |
| 1100 | 394666 | 746221 | 8803 | 8803 | 7703 |
| 1200 | 1307554 | 856888 | 9708 | 9708 | 8508 |
| 1300 | 639111 | 822222 | 10621 | 10621 | 9321 |
| 1400 | 412444 | 1026665 | 11542 | 11542 | 10142 |
| 1500 | 614222 | 973332 | 12470 | 12470 | 10970 |
| 1600 | 398666 | 465777 | 13404 | 13404 | 11804 |
| 1700 | 421332 | 960888 | 14345 | 14345 | 12645 |
| 1800 | 429777 | 1542664 | 15292 | 15292 | 13492 |
| 1900 | 569777 | 1001332 | 16244 | 16244 | 14344 |
| 2000 | 784888 | 1034665 | 17202 | 17202 | 15202 |
| 2100 | 931999 | 797332 | 18164 | 18164 | 16064 |
| 2200 | 1524442 | 741333 | 19132 | 19132 | 16932 |
| 2300 | 792888 | 807999 | 20104 | 20104 | 17804 |
| 2400 | 613777 | 1261776 | 21080 | 21080 | 18680 |
| 2500 | 601777 | 1051110 | 22060 | 22060 | 19560 |
| 2600 | 985777 | 1571554 | 23044 | 23044 | 20444 |
| 2700 | 730666 | 1719109 | 24033 | 24033 | 21333 |
| 2800 | 991999 | 1661332 | 25025 | 25025 | 22225 |
| 2900 | 1049332 | 1723998 | 26020 | 26020 | 23120 |
| 3000 | 1763998 | 1420442 | 27019 | 27019 | 24019 |
| 3100 | 1764442 | 1077776 | 28021 | 28021 | 24921 |
| 3200 | 2363108 | 1629331 | 29027 | 29027 | 25827 |
| 3300 | 2147553 | 1277776 | 30036 | 30036 | 26736 |
| 3400 | 1599998 | 1227998 | 31047 | 31047 | 27647 |
| 3500 | 1307998 | 1091999 | 32062 | 32062 | 28562 |
| 3600 | 1279554 | 928444 | 33079 | 33079 | 29479 |
| 3700 | 1328442 | 949776 | 34100 | 34100 | 30400 |
| 3800 | 2398219 | 841777 | 35122 | 35122 | 31322 |
| 3900 | 1081776 | 875110 | 36148 | 36148 | 32248 |
| 4000 | 991999 | 804443 | 37176 | 37176 | 33176 |
| 4100 | 1211554 | 1853330 | 38207 | 38207 | 34107 |
| 4200 | 1375998 | 1130665 | 39240 | 39240 | 35040 |
| 4300 | 1214221 | 1294220 | 40275 | 40275 | 35975 |
| 4400 | 1153776 | 1403554 | 41313 | 41313 | 36913 |
| 4500 | 1236443 | 1204887 | 42353 | 42353 | 37853 |
| 4600 | 2505774 | 2520441 | 43396 | 43396 | 38796 |
| 4700 | 1514221 | 2079998 | 44440 | 44440 | 39740 |
| 4800 | 2831552 | 1307109 | 45487 | 45487 | 40687 |
| 4900 | 2608441 | 2188886 | 46535 | 46535 | 41635 |
| 5000 | 1782664 | 2329774 | 47586 | 47586 | 42586 |
| 5100 | 1399109 | 2230664 | 48639 | 48639 | 43539 |
| 5200 | 1535109 | 1396887 | 49693 | 49693 | 44493 |
| 5300 | 1375554 | 1531109 | 50750 | 50750 | 45450 |
| 5400 | 971998 | 1373331 | 51808 | 51808 | 46408 |
| 5500 | 938665 | 1560442 | 52869 | 52869 | 47369 |
| 5600 | 1007554 | 1689776 | 53931 | 53931 | 48331 |
| 5700 | 1563998 | 1393331 | 54995 | 54995 | 49295 |
| 5800 | 1035999 | 1369332 | 56061 | 56061 | 50261 |
| 5900 | 1306221 | 1701331 | 57128 | 57128 | 51228 |
| 6000 | 1439998 | 1444443 | 58197 | 58197 | 52197 |
| 6100 | 1267553 | 2545330 | 59268 | 59268 | 53168 |
| 6200 | 1346665 | 2313331 | 60340 | 60340 | 54140 |
| 6300 | 824444 | 2588886 | 61414 | 61414 | 55114 |
| 6400 | 833332 | 2380442 | 62490 | 62490 | 56090 |
| 6500 | 864443 | 2395996 | 63567 | 63567 | 57067 |
| 6600 | 1010665 | 2192886 | 64646 | 64646 | 58046 |
| 6700 | 1479998 | 2461330 | 65726 | 65726 | 59026 |
| 6800 | 980443 | 2524441 | 66808 | 66808 | 60008 |
| 6900 | 1449331 | 2564886 | 67891 | 67891 | 60991 |
| 7000 | 1384887 | 2601330 | 68976 | 68976 | 61976 |
| 7100 | 866221 | 2599996 | 70062 | 70062 | 62962 |
| 7200 | 1081332 | 2878218 | 71149 | 71149 | 63949 |
| 7300 | 1423554 | 2737329 | 72238 | 72238 | 64938 |
| 7400 | 1518665 | 2723107 | 73328 | 73328 | 65928 |
| 7500 | 1483998 | 2909774 | 74420 | 74420 | 66920 |
| 7600 | 1461332 | 2840885 | 75513 | 75513 | 67913 |
| 7700 | 1132887 | 2868441 | 76607 | 76607 | 68907 |
| 7800 | 1156443 | 2858663 | 77703 | 77703 | 69903 |
| 7900 | 1153332 | 2897330 | 78799 | 78799 | 70899 |
| 8000 | 1123110 | 2977774 | 79898 | 79898 | 71898 |
| 8100 | 1256887 | 2958218 | 80997 | 80997 | 72897 |
| 8200 | 1935109 | 3057329 | 82097 | 82097 | 73897 |
| 8300 | 1627997 | 2048442 | 83199 | 83199 | 74899 |
| 8400 | 1256887 | 3175996 | 84302 | 84302 | 75902 |
| 8500 | 1365776 | 2535108 | 85406 | 85406 | 76906 |
| 8600 | 1161331 | 2542218 | 86512 | 86512 | 77912 |
| 8700 | 2237330 | 2253330 | 87618 | 87618 | 78918 |
| 8800 | 1208443 | 2223108 | 88726 | 88726 | 79926 |
| 8900 | 1358665 | 3416884 | 89835 | 89835 | 80935 |
| 9000 | 1227998 | 4716438 | 90945 | 90945 | 81945 |
| 9100 | 1254220 | 5119993 | 92056 | 92056 | 82956 |
| 9200 | 1172443 | 3788884 | 93168 | 93168 | 83968 |
| 9300 | 1293332 | 3211551 | 94281 | 94281 | 84981 |
| 9400 | 1201776 | 3969328 | 95396 | 95396 | 85996 |
| 9500 | 1317776 | 3592440 | 96511 | 96511 | 87011 |
| 9600 | 1288887 | 3280440 | 97627 | 97627 | 88027 |

10) Входные данные

Файл со входными данными в папке