کروسکال

هدف پیدا کردن درخت کمینه به وسیله ی روش کروسکال می باشد.

مطابق تابع زیر

```
void kruskal (int n, int m,
              set_of_edges E,
              set_of_edges& F)
{
  index i, j;
  set_pointer p, q;
  edge e;
  Sort the m edges in E by weight in nondecreasing order;
  F = ∅;
  initial(n);                    // Initialize n disjoint subsets.
  while (number of edges in F is less than n - 1){
    e = edge with least weight not yet considered;
    i, j = indices of vertices connected by e;
    p = find(i);
    q = find(j);
    if (! equal(p, q)){
       merge(p, q);
       add e to F;
    }
  }
}
```

برنامه شامل سه فایل Edge, Kruskal, Main می باشد

در فایل Edge کلاسی با همین نام وجود دارد که مشخصات یک یال زا ذخیره میکند شامل شماره راس های دو سر آن و وزن روی یال

در فایل کروسکال کلاسی با این نام تشکیل شده است که سازنده این کلاس به شکل داده شده در تصویر بالا می باشد

که مطابق متغیر های بالا میباشد و برای مرتب کردن mیال یک صف اولویت برای راحتی کار استفاده شده که در مقایسه کننده آن مشخص کرده ایم که به صورت غیر نزولی مرتب سازی کند.

سپس متد های initial وmerge و find مطابق کتاب پیاده سازی شده اند

و در انتهای سازنده از equals کلاس edge برای مقایسه ی p,q استفاده شده است.

```
11      Kruskal(int n, int m, ArrayList<Edge> edges) {
12          int i, j;
13          ArrayList<Integer> p, q;
14          Edge e;
15
16
17          Comparator<Edge> comparator = (e1, e2) -> {
18              return e1.getWeight() - e2.getWeight();
19          };
20          PriorityQueue<Edge> edgePriorityQueue = new PriorityQueue<>(comparator);
21          for (int k = 0; k < n; k++) {
22              edgePriorityQueue.add(edges.get(k));
23          }
24
25          F = new ArrayList<Edge>();
26
27          initial(n);
28
29          while (F.size() < n - 1) {
30              e = edgePriorityQueue.remove();
31              i = e.getI();
32              j = e.getJ();
33              p = find(i);
34              q = find(j);
35
36              assert p != null;
37              if(!p.equals(q)){
38                  merge(p,q);
39                  F.add(e);
40              }
41          }
42
43      }
```

در متد merge کل اعضای زیر مجموعه دومی را به اولی اضافه میکنیم و زیر مجمه دومی را از بین میبریم

```
45  @       private void merge(ArrayList<Integer> p, ArrayList<Integer> q) {
46              p.addAll(q);
47              q.clear();
48          }
```

در متد initial آرایه ای از لیست ها که همان زیرمجموعه هایمان هستند میسازیم و هر کدام از آنهارا مقدار دهی اولیه میکنیم و شماره همان راس را به زیر مجموعه اضافه میکنیم

```
51      private void initial(int n) {
52          subsets = new ArrayList[n];
53
54          for (int i = 0; i < n; i++) {
55              this.subsets[i] = new ArrayList<Integer>();
56              this.subsets[i].add(i);
57          }
58      }
```

در متد find شماره یک راس را میگیریم و در تمام آرایه زیر مجموعه هایمان میگردیم تا عضوی با آن شماره پیدا کنیم و آن زیر مجموعه را بر میگردانیم

```
60 @    private ArrayList<Integer> find(int i){
61          for (ArrayList<Integer> subset : subsets) {
62              if(subset == null) continue;
63              if (subset.contains(i)) return subsets[i];
64          }
65          return null;
66      }
```

بقیه ی توضیحات مربوط به کروسکال در کتاب موجود میباشد ☺

در فایل Main متد های start, input, draw را داریم که از با لانچ کردن main متد start صدا زده میشود

در متد start متغییر های مربوط به محیط set می شوند و سپس input صدا زده می شود

در ابتدای input یک عدد صحیح برای تعداد راس ها گرفته می شود سپس ماتریس مجاورت وزن دار از کاربر میگیریم

و یال های موجود در آن را می سازیم و در لیست edges ذخیره میکنیم

در انتها هم از کلاس کروسکال یک نمونه میسازیم و طول یال ها و راس ها به همراه مجموعه یال هارا به سازنده کلاس میدهیم و بعد از آن لیست F شامل لیست یال های بدست آمده با روش کروسکال را از نمونه کلاس ساخته شده میگیریم و به متد draw میدهیم

در متد draw سه حلقه برای کشیدن کل یالها و یالهای بدست آمده از کروسکال و راس ها وجود دارد

که به کمک روابط سینوس کسینوس و مختصات قطبی راس ها و یال هارا میکشیم و بر روی راس ها شماره آنها و بر روی یال ها وزنشان را قرار میدهیم

در انتها شکل گراف نشان داده می شود که یال های سیاه نشان دهنده کل یالها و یال های قرمز نشان دهنده یال های درخت کمینه بدست آمده از کروسکال میباشد

اجرا :

```
5
0 2 0 6 0
2 0 3 8 5
0 3 0 0 7
6 8 0 0 9
0 5 7 9 0
{i=0, j=1, weight=2}{i=1, j=2, weight=3}{i=1, j=4, weight=5}{i=0, j=3, weight=6}
```