توضیحات سوال اول پروژه اول درس هوش مصنوعی

با توجه به اینکه این سوال با جاوا پیاده سازی شده است، توضیح مختصری از کلاس ها(با رنگ زرد) و برخی فیلدها(با رنگ سبز) و متدهای آن(با رنگ بنفش) آورده شده است:

- کلاس Cube: در اینجا، مکعب خود را با یک آرایه دو بعدی sides_colors تعریف می کنیم.
  همچنین، اینگونه تعریف می کنیم که سطح 1، جلو(F)، سطح 3، پایین(D)، سطح 2، چپ(L)، سطح 4، راست(R)، سطح 5، پشت(B) و سطح 6، بالا(U) باشد.
  برای چرخش ها نیز، متناسب با اینکه بخواهیم کدام وجه را ساعتگرد یا پادساتگرد بچرخانیم، تابع چرخش متناظر با آن صدا زده می شود.(به عنوان مثال، اگر میخواهیم وجه راست را پادساعتگرد بچرخانیم، کافی است کُدِ 21 را به تابع rotate دهیم تا تابع rotate_right_counter_clockwise فراخوانی شود)
  به طور کلی، در هر چرخش جای 12 خانه با هم عوض می شود. همچنین، می توان چرخش های پادساعتگرد یک وجه را به صورت چرخش ساعتگرد وجه روبرویش نوشت(یعنی مثلا کُدِ 30 معادل کد 61 خواهد بود) با این کار، می توان حداکثر ضریب انشعاب را به 6 کاهش داد.
- کلاس IDS_Search: در اینجا، یک گراف(graph) و یک عمق اولیه(depth) می گیریم و با شروع از عمق اولیه، روی آن، الگوریتم جستجوی اول عمق تکرار شونده را اعمال می کنیم. این کلاس، خود مدام کلاس DLS_Search را ایجاد می کند و الگوریتم جستجوی اول عمق محدود شده را اجرا می کند.

```
for (int i = depth;; i++) {
    DLS_Search dls_search = new DLS_Search(cube, i);
    String result = dls_search.simulate(cube, i);
    System.out.println("explored set is: ");
    for (int j = 0; j < dls_search.explored_set.size(); j++) {
        System.out.println(dls_search.explored_set.get(j));
    }
    System.out.println("explored set size is: " + dls_search.explored_set.size());
    if (!result.equals("cutoff")){
        System.out.println("IDS stopped at depth: " + i);
        return result;
    }
}
```

- کلاس DLS_Search: همانطور که گفته شد، در اینجا، دوباره یک گراف(graph) و یک عمق(depth) می گیریم و روی آن جستجوی اول عمق محدود شده را اعمال می کنیم.

- کلاس Bidirectional_Search: در اینجا، با در نظر گرفتن جستجوی دو طرفه اول سطح، مسئله را حل می کنیم. یک گراف(graph) در ورودی گرفته و با در نظر گرفتن حالت نهایی، از دو طرف حرکات مکعب را انجام می دهیم تا به یک حالت مشترک برسیم.

- کلاس UCS_Search: در اینجا، با گرفتن یک گراف(graph) می آییم و مادامی که frontier ما تهی نباشد، حالات را از frontier استخراج و پس از چک کردن goal_test، در صورت مردود شدن، حرکات را بر روی آن حالت انجام داده(فرزندان را تولید کرده) و آن ها را به frontier اضافه می کنیم.
  پ.ن: با توجه به اینکه هزینه حرکات یکسان می باشد، این الگوریتم همانند الگوریتم BFS عمل می کند.

همچنین، تعدادی حالت اولیه تست شده در فایل test.txt قرار داده شده است.

نکته مهم: الگوریتم ها با جستجوی گرافی پیاده سازی شده اند. اما می توان با کامنت کردن بررسی وجود داشتن حالت توی explored_set، به صورت درختی آن ها را اجرا کرد.