توضیح الگوریتم های پروژه

در این پروژه از چند ساختمان داده مختلف استفاده شد که عبارتند از:

- لیست پیوندی (Linked List)
- مکس هیپ (Max Heap)
- لیست (`list` in python)
- صف اولویت دار (Priority List)

این پروژه درباره تعدای سرویس و زیرسرویس و نمایندگی های ارائه دهنده آنها بود که با توجه به مفهوم این موضوع میتوان دریافت که ساختمان داده ای مانند لیست پیوندی برای پیاده سازی چنین چیزی مناسب است.
برای سفارشات هر نمایندگی از صف اولویت دار استفاده شد چرا که هر سفارش دارای اولویتی نسبت به دیگر سفارشات دارد، یا از نظر زمان ارائه سفارش یا اولویت ذکر شده توسط مشتری.
با توجه با ساختار صف پیوندی، مکس هیپ برای پیاده سازی آن مناسب است.
از لیست هم در قسمت هایی از پروژه برای نگه داری اطلاعات استفاده شده است.

در کل، ۴ کلاس تعریف شده است:

- `class Node`
- `class LinkedList`
- `class HeapNode`
- `class MaxHeap`

کلاس `Node` صرفا یک node برای استفاده در لیست پیوندی ایجاد میکند.
کلاس `LinkedList` یک لیست پیوندی از node ها میسازد.
الگوریتم هایی که در این کلاس استفاده شده عبارتند از:

1. برای `add` کردن یک node بسته به این که یک سرویس اصلی یا یک زیرسرویس است، به لیست پیوندی اضافه میشود. به این صورت که:
   1.1. اگر سرویس اصلی باشد، صرفا با طی کردن طول لیست پیوندی به انتهای آن اضافه میشود. که این کار از مرتبه زمانی تعداد سرویس های اصلی است.
   1.2. اگر زیرسرویس باشد ابتدا توسط تابع `search` مکانی که باید به آن اضافه شود را بدست آورده و سپس آن را اضافه میکنیم. مرتبه زمانی این تابع برابر با مرتبه زمانی تابع `search` میباشد که جلوتر به آن پرداخته میشود.
2. برای `search` کردن از یک لیست کمک میگیریم. با هر اضافه کردن سرویس یا زیرسرویس، node اضافه شده همزمان در یک لیست هم ذخیره میشود. پس برای `search` تنها کافیست لیست مذکور را پیمایش کنیم که این کار از مرتبه زمانی پیمایش تعداد خانه های اشغال شده لیست است.
3. برای `delete` کردن یک node در لیست مذکور پیمایش کرده و وقتی به node قبل از node مورد نظر رسیدیم، اشاره گر بعدی آن را به اشاره گر بعدی آن node حذف شده تغییر میدهیم. این کار از مرتبه زمانی پیمایش تعداد خانه های اشغال شده لیست است.
4. برای `delete` کردن یک سرویس ارائه شده توسط یک نمایندگی از لیست سرویس های ارائه شده توسط آن، ابتدا در لیستی که سرویس های ارائه شده در آن ذخیره میشوند پیمایش کرده سپس با پیدا کردن سرویس مورد نظر آن را از لیست حذف میکنیم. سپس چک میکنیم که اگر

در دیگر نمایندگی ها نبود آن سرویس به کل حذف شود که این کار از مرتبه زمانی پیمایش تعداد خانه های اشغال شده لیست است.

5. تابع `add offer` یک سرویس اصلی را به یک نمایندگی اضافه میکند. این تابع ابتدا با تابع `search` وجود سرویس و نمایندگی را بررسی میکند. سپس در صورت وجود هردو، سرویس را به لیستی که در نمایندگی تعبیه شده اضافه میکند. مرتبه زمانی این کار با تابع `search` برابر است.
6. تابع `list agencies` با پیمایش در لیست نمایندگی ها همه آنها را چاپ میکند. این کار از مرتبه زمانی پیمایش تعداد خانه های اشغال شده لیست است.
7. تابع `list services` دو حالت دارد:
   7.1. اگر بدون پارامتر ورودی فراخوانی شود، همه سرویس های موجود را به صورت بازگشتی به همراه زیرسرویس های آنها چاپ مینماید. که این کار از مرتبه زمانی تعداد خانه های لیست پیوندی است.
   7.2. اگر با پارامتر ورودی فراخوانی شود، همه زیرسرویس های یک سرویس به همراه خود آن سرویس را چاپ میکند. مرتبه زمانی این کار برابر تعداد زیر سرویس های آن سرویس است.
8. تابع `child` همه زیرسرویس های یک سرویس را به صورت بازگشتی در یک لیست میریزد. مرتبه زمانی این کار برابر تعداد زیر سرویس های آن سرویس است.
9. تابع `order` یک سفارش ایجاد میکند. ابتدا نمایندگی موردنظر را پیدا کرده سپس سرویس را پیدا میکند. سپس یک `HeapNode` درست کرده و آن را به صف اولویت دار اضافه میکند. مرتبه زمانی این کار برابر تعداد سرویس های ارائه شده توسط نمایندگی ضربدر تعداد زیرسرویس های سرویس درخواستی است.
10. تابع `list orders` صف اولیت دار یک نمایندگی را پس از یافتن آن خالی میکند و نمایش میدهد. مرتبه زمانی این کار برابر تعداد خانه های اشغال شده لیست و صف اولویت دار است.

کلاس `HeapNode` صرفا یک نوع node برای استفاده در صف اولویت دار یا مکس هیپ میسازد.
کلاس `MaxHeap` دقیقا مانند مکس هیپ گفته شده در کلاس طراحی شده و توابع آن مرتبه زمانی یکسان با آن دارند.

توضیح نحوه اجرای برنامه

با اجرای برنامه شما میتوانید با وارد کردن فرمان `HELP` به طور کامل نحوه صحیح دستوردهی را ملاحظه کنید.

در صورت تخطی از این دستورالعمل برنامه به شما اخطار خواهد داد.

همچنین برای تست سریع برنامه دستورهای مورد نظر را متوالیا در جایی نوشته سپس آنها را در کنسول paste کنید تا نیازی به تایپ تک تک آنها نباشد.
دقت کنید در صورت انجام این کار حتما باید یک enter اضافه در انتهای دستورها قرار داده شود، یا در غیر این صورت پس از paste کردن یک بار کلید enter را فشار دهید.

یک نمونه از دستور های متوالی را در شکل روبرو مشاهده میکنید.
در اینجا خط 30 یک enter اضافی دارد.

همینطور از وارد کردن اسامی دو بخشی مانند:
`front door`
که با فاصله از هم جدا شده اند پرهیز کنید، این کلمات را با _ از یکدیگر جدا کنید:
`front_door`

```
add service a
add service b
add service c
add subservice a1 to a
add subservice a2 to a
add subservice a3 to a
add subservice a11 to a1
add subservice a12 to a1
add subservice a111 to a11
add subservice a21 to a2
add agency z
add agency y
add agency x
add offer a to x
add offer a to y
add offer a to z
add offer b to y
add offer b to z
add offer c to x
order a12 to x by parsa with low priority
order a1 to x by parsa with normal priority
order a2 to x by parsa with low priority
order b to x by parsa with high priority
order c to x by parsa with low priority
order a to x by parsa with normal priority
order a11 to x by parsa with low priority
order a111 to x by parsa with high priority
order a to x by parsa with low priority
order a1 to x by parsa with normal priority

```

با تشکر
پارسا انعامی