WiseBidders: مزایده گران عاقل

هدف این پروژه، پیاده سازی پروتکل مزایده گری توضیح داد شده است. گرچه انجام اینکار با روشی ساده­تر و بدون استفاده از اصول شی­گرایی هم ممکن بود، اما برای اینکه کار را اصولی انجام دهیم و همچنین کد قابل فهم و قابل بسط یا دوباره استفاده کردن باشد، در این پروژه از اصول شی گرایی بهره جستیم.

از این رو، پس از دریافت ورودی­های مذکور طبق صورت سوال، اعم از ماتریس حداکثر سرمایه خریداران (V) و لیست حداقل قیمت­های خانه­ها و ...، آن­ها را طی فرایند preprocessing تحت قالب شی گرای طراحی شده می­بریم و پس از آن فرایند مزایده و خرید و فروش را پیاده­سازی می­کنیم که الگوریتم آن بسیار ساده تر و قابل فهم تر از حالت کلاسیک خواهد بود.

# ساختار کلی و الگوریتم

پروژه شامل چند کلاس اصلی می­باشد که طبق نامشان کاربردشان هم قابل پیش­بینی است اما در اینجا باز به توضیح آن­ها در بخش مربوطه می­پردازیم.

الگوریتم کلی به این صورت است که پس از دریافت داده خام ورودی، اطلاعات حداقل قیمت خانه ها را به لیستی از عناصر کلاس House تبدیل می­کنیم، همچنین داده­های خریدارها را هم به همین منوال استخراج کرده و تحت قالب لیستی از عناصر کلاس Bidder ذخیره می­کنیم.

سپس یک شی Auction خواهیم ساخت که وظیفه پروتکل مزایده را برعهده دارد. این کلاس با فراخوانی متد start خود، طبق اطلاعات House ها و Bidder ها و پیشنهادهای طبق پروتکلی تحت قالب دیکشنری ای از عناصر کلاس Bid ایجاد می­کند و با استفاده از این عناصر هر راند مزایده را مدیریت می­کند. نهایتا وقتی قیمت­ها به سکون رسیدند خریدارهای نهایی و صاحبان نهایی خانه­ها اعلام می­شوند.

# کلاس­ها

حال به تشریح کلاس ها بصورت مستقل می­پردازیم:

## 2-1 کلاس House

این کلاس اطلاعات هر خانه داخل مزایده را نگه می­دارد؛ اکرچه در این مورد داده اصلی صرفا حداکثر قیمت موردنظر فروشنده است، با این وجود ما دو متغیر دیگر با نام­های id و name هم برای این کلاس جهت ایجاد تمایز میان خانه­ها تعریف کرده­ایم.

این کلاس تابع ممبر خاصی ندارد، صرفا یک مشخصه به نام is\_sold که نمایانگر فروش رفتن یا نرفته خانه است دارد. همچنین یک تابع استاتیک در این کلاس با نام ArrangeHouseInstances تعریف شده که وظیفه­اش دریافت ورودی به فرم خواسته شده سوال (لیستی از حداقل قیمت خانه­ها) و تبدیل آن به لیستی از اشیاء House می­باشد.

همچنین این کلاس دو فیلد مهم با نام­های best\_bid و sold\_to دارد. Best\_bid در هردور مزایده مشخصه کننده بهترین پیشنهاد حال حاضر آن خانه و sold\_to -که درآخر مزایده مقدار می­گیرد- مشخص کننده آبجکت خریدار این خانه است.

## 2-2 کلاس Bidder

این کلاس همانطور که از نامش برمی­آید، مشخص کننده یک مزایده­گر یا همان خریدار است. همانند کلاس House این کلاس هم مشخصه­های id و name را جهت متمایز کردن اشخاص داراست. فیلد دیگری با نام purchased دارد که در انتهای بازی و وقتی یک خریدار پیشنهاد نهایی­اش ثبت شده و خریدش نهایی می­شود مقدار می­گیرد و برابر با آبجکت خانه­ی خریداری شده توسط خریدار می­گردد. همچنین فیلد max\_invest\_per\_house نیز لیستی از حداکثر سرمایه هر خریدار به ازای خانه­های موجود در مزایده است.

درواقع پس از فراخوانی متد ArrangeBidderInstances -که متدی مشابه با متد ArrangeHouseInstances می­باشد- با دریافت ماتریس Vn x m -که درایه vi x j نشانگر حداکثر سرمایه خریدار j ام برای خرید خانه i ام است- لیستی از خریدارها را تولید می­کند. مشخصات هر خریدار در هر یک از ستون­های این ماتریس وجود دارد که هر سطر i در این ستون، معادل حداکثر سرمایه­ی خرید یک خریدار، برای خرید خانه i ام است. درواقع فیلد max\_invest\_per\_house کلاس Bidder معادل با یک ستون خاص در این ماتریس است.

## 3-2 کلاس Bid

این کلاس نمایانگر یک پیشنهاد مزایده در پروتکل ماست. در واقع هر پیشنهاد یک مزاید­ه­گر برای یک خانه­ی خاص تحت قالب این کلاس می­باشد. فیلدهای اصلی این کلاس فیلد item که مشخص کننده خانه­ای که پیشنهاد خریدش داده ­می­شود و فیلد bidder که مشخص کننده خریداری که پیشنهاد را می­دهد است، می­باشند. همچنین فیلدهای دیگری هم هستند که به صورت زیر می­باشند:

* max\_investment: که مشخص کننده حداکثر هزینه­ای که bidder این پیشنهاد حاضر است برای item اش بکند. این مقدار درواقع از لیست max\_investment\_per\_house یک bidder مقدار دهی می­شود.
* suggested\_price: مشخص کننده هزینه پیشنهادی حال حاضر (درهردور مزایده) می باشد.
* profit: یکی از فیلدهای مهم این کلاس می­باشد که در ادامه مبنای انتخاب پیشنهاد اصلی bidder ها خواهد بود. طبق فرمول ذکر شده در پروتکل این فیلد مشخصه کننده مقدار سود حال حاضر این پیشنهاد برای خریدار می­باشد. این سود با محاسبه فاصله suggested\_price از max\_investment خریدار برای item انتخابی این bid بدست می­آید.
* lose\_count: مشخص کننده تعداد دفعات شکست پیشنهاد است. منظور از شکست مشخص کننده تعداد دفعاتی است که خریدار نتواند برای item پیشنهاد دهد، یعنی میزان پیشنهاد ها از سقف max\_investment این bid عبور کرده باشد.
* failed: طبق تعریف پروتکل، هنگامی که یک خریدار روی خانه ای مزایده می­کند، اگر به تعداد دو دور متوالی نتواند پیشنهادی بدهد، دیگر نمیتواند به آن خانه پیشنهادی دهد. فیلد failed دقیقا مشخص کننده همین وضعیت است که وقتی True باشد دیگر آیتم bid را فاقد اعتبار میکند و در الگوریتم باعث کنار گذاشته شدن آن می­گردد.

همچنین متدهای اصلی این کلاس به شرح زیر می­باشند:

* can\_raise: این تابع با دریافت ورودی new\_price از قیمت بعدی که در مزایده باید برای یکم item پیشنهاد داده شود مطلع شده و با مقایسه آن با max\_investment تعیین می­کند آیا bidder قادر به افزایش پقیمت پیشنهادی خود هست یا کم کم باید کنار بکشد!
* lose: این تابع زمانی که خریدار نتواند پیشنهاد دهد، یعنی درواقع زمانی که خروجی can\_raise منفی باشد، فراخوانی می­کند و lose\_count را افزایش می­دهد. هم­چنین طبق تعریف صورت سوال، اگر مقدار این فیلد 2 یا بیشتر شد، bid را failed تلقی خواهد کرد.
* raise\_price: این تابع با دریافت قیمت هدف، قیمت پیشنهادی این bid را افزایش می­دهد و سپس سود جدید را طبق تعرفه های جدید محاسبه می­کند.
* temp\_win: این تابع در هر راند برای پیشنهادی که مناسبترین تلقی می­شود فراخوانی می­شود و درواقع همان فرایند برنده شدن موقتی را پیاده سازی می­کند. این تابع با به روز رسانی فیلد best\_bid خانه هدف با خود آبجکت bid این کار را انجام می­دهد.
* win: این تابع یک تابع نهایی تلقی می­شود که پس از مشخص شدن وضعیت نهایی bid ها و به ثبات رسیدن قیمت ها، در راند آخر فراخوانی شده و برنده و خریدار نهایی هر خانه را مشخص می­کند.

## 4-2 کلاس Auction

این کلاس، پیاده کننده پروتکل مزایده می­باشد و درواقع کلاس اصلی و مدیر برنامه می­باشد. پس از گرفتن ورودی کاربرها در ماژول app و تبدیل آنها به فرمت شی­گرا، یک آبجکت Auction ساخته می­شود که به آن لیست bidder ها و house ها و شرایط مزایده از قبیل گام مزایده € و تعداد ماکزیمم راند (درصورتی که برنامه در تعداد راندهای بیش از حد زیاد گیر کرد، این مقدار مانع هنگ کردن برنامه میشود) پاس داده شده و با فراخوانی متد start مزایده طبق آنچه که تعریف شده شروع و راندهای آن مدیریت میگرند. درادامه الگوریتم کار start شرح داده میشود.

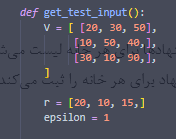
روند کلی کار این تابع، به این صورت است که با استفاده از لیست­های خریدارها و خانه ها (تحت عنوان bidders و items) ابتدا برای هر bidder تابع analyze خود را فراخوانی می­کند؛ این تابع تمامی پیشنهادهای مزایده آبجکت های bid) را به ازای هر خریدار می­سازد و سپس به ترتیب مقدار profit آن­ها برای خریدارش مرتب می­کند. سپس هر کدام از این لیست­های مرتب شده­ی جدید را تحت قالب یک آیتم دیکشنری با کلید از نوع Bidder و مقدار List[Bid]، برای هر خریدار در می­آورد. درواقع در نهایت، این دیکشنری شامل تمامی پیشنهاد­هایی که یک خریدار می­تواند برای هر خانه بدهد می­باشد. مرتب شدن پیشنهادهای یک bidder بر حسب profit به این معناست که در هر دور مزایده، به ازای هر خریدار، عنصر اول bid هایش مشخص کننده بهترین پیشنهاد وی در آن دور می­باشد.

نکته: درصورتی که یک خریدار در ابتدای لیست خود چند bid با مقدار سود یکسان داشته باشد، با فراخوانی تابع randomize\_first\_bid\_by\_profit\_if\_needed پیشنهاد اصلی خریدار بصورت تصادفی در می­آید ]طبق همان چیزی که در پروتکل خواسته شده است[. حالا به ازای هر خریدار عنصر اول لیست bid هایش انتخاب شده، و به یک دیکشنری دیگر تحت عنوان bids\_per\_item اضافه می­شود. دیکشنری جدید برای هر خانه لیست bid های داده شده به آن در آن راند را شامل است. سپس لیست های موجود در این دیکشنری سپس بر حسب suggested\_price بصورت نزولی مرتب می­شوند که این یعنی گران­ترین پیشنهاد برای یک خانه همان عنصر اول این لیست خواهد بود. همانند قبل درصورتی که چند پیشنهاد با مقدار suggested\_price مساوی موجود باشد از طریق فراخوانی تابع randomize\_first\_bid\_by\_price\_if\_needed بهترین bid راند کنونی بصورت تصادفی انتخاب می­گردد.

طبق این توزیع داده که در بالا شرح داده شد، در راند اول پیشنهادها برای هر خانه لیست می­شوند، سپس این لیست از طریق فراخوانی accept\_best\_bid بهترین پیشنهاد برای هر خانه را ثبت می­کند (از طریق فراخوانی تابع temp\_win بهترین bid)

سپس در راندهای بعدی، برای هر آیتمی که بهترین bid دارد، درصورتی که خریداری دیگر دارای bid ای باشد که هنوز قادر به افزایش قیمت باشد ( can\_raise او برابر با True شود) با فراخوانی raise\_price به میزان € نسبت به بهترین قیمت قبلی پیشنهاد جدیدی ثبت می­کند (البته اگر این bid بهترین پیشنهاد انتخابی یک bidder باشد.) دوباره طبق روش مذکور، دیکشنری ها آپدیت می­شوند و temp\_win ها فراخوانی می­گردد و پیشنهادها و best\_bid ها آپدیت می­شوند؛ این روند تا زمانی ادامه پیدا می­کند که قیمت­ها در 3 راند اخر کاملا ثابت بمانند؛ در آن صورت در هر item با فراخوانی متد win فیلد best\_bid خود خریدار برنده را اعلام می­کند. تمامی این اطلاعات هم در هر راند مزایده در کنسول چاپ می­شوند و کاربر را مطلع ­می­کنند.

در ادامه نمونه خروجی برنامه را برای ورودی زیر قرار داده ایم:



نمونه ای از خروجی برنامه:

