**به نام خدا**

**دانشگاه اصفهان ،دانشکده مهندسی کامپیوتر**

**نیم سال اول تحصیلی** 01 – 00

**مستند پروژه درس ساختمان داده ها**

**دکتر فاطمی**

گیلو فارغ التحصیل رشته‌ی آشپزی است و برای کسب تجربه در این حرفه‌ی خوشمزه، زمانی را نزد بهترین آشپزهای فرانسوی، ایتالیایی و انگلستانی سپری کرده‌است. او که معتقد است اخرین فوت و فن‌های آشپزی را فقط می تواند در ایران و نزد مادربزرگش یاد بگیرد، برای گذراندن آخرین خان به ایران برمیگردد. بعد از مدتی و در حالی که چند کیلویی وزن اضافه کرده، تصمیم میگیرد که ایران بماند و رستوران خودش را افتتاح کند. او بعد از تحقیقات فراوان به این نتیجه رسید که مثل همه ی بخش های دیگر دنیا، یک رستوران خوب علاوه بر غذاهای خوشمزه، برای رقابت با بقیه‌ی رستوران ها نیاز به تکنولوژی و به روز بودن هم دارد، بنابراین تصمیم گرفت به دوست قدیمی اش، ریضو، که فارغ التحصیل مهندسی کامپیوتراست، پیشنهاد شراکت بدهد. بعد از سر و کله زدن‌های فراوان، بالاخره ریضو این پیشنهاد رو قبول کرد و بدین ترتیب رستوران خرخون باشی افتتاح شد!

## بخش پذیرش مشتری:

از آنجایی که رستوران خرخون‌باشی در یکی از شلوغ‌ترین مراکز خرید شهر قرار داشت، اکثر اوقات مشتری‌ها برای ورود به رستوران باید در سالن انتظار منتظر بمانند تا به نوبت ورودشان به رستوران، صندوقدار از آنها سفارش گرفته و به سمت یک میز خالی، راهنمایی‌شان کند.

در ورودی در ابتدا نقشه‌ی هوایی رستوران به صورت یک آرایه‌ی دوبعدی از کاراکترها به شما داده میشود، هر میز با یک حرف الفبای کوچک یا بزرگ، وسایل زینتی با # ، جاهای خالی با + و آشپزخانه با $ مشخص می‌شوند. در ادامه در هر خط ورودی نام فردی که وارد رستوران میشود، غذایی که سفارش میدهد، مدت زمانی که طول میکشد تا سفارش او آماده شود و مدت زمانی که آن فرد صرف غذا خوردن میکند آمده‌است و شما باید به ترتیب و با توجه به میزهای خالی افراد را به سمت میز مناسب راهنمایی کنید.

```
#+#+$++++#
+#++#+###+
a+###++#++
+#++#+b+#+
#+c+#+++##
++#+##++++
##+#+##+++
++++#e#+##
++#++##+#+
d#####+++#+
```

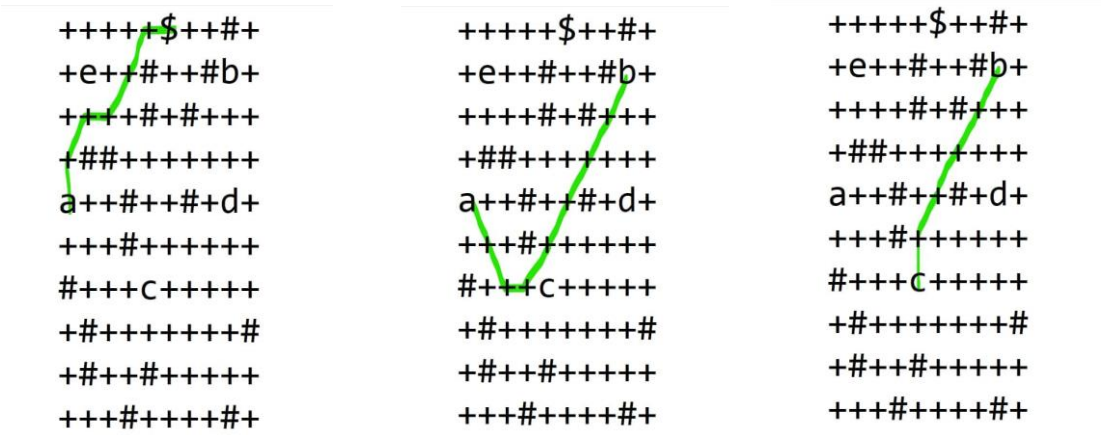
مثالی از چینش المان های رستوران

## تحویل غذا توسط گارسون:

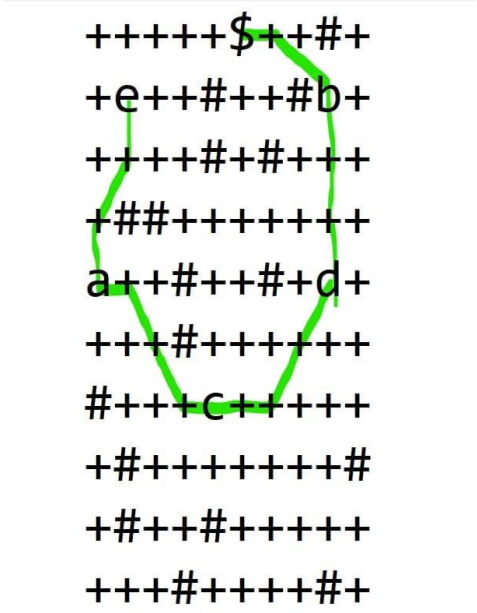
یکی از اولین چالش هایی که سر راه آنها قرار داشت، رساندن غذا به میزها بود.

آن‌ها تصمیم گرفتند چون اول راه هستند، فقط یک گارسون استخدام کنند. روند کار گارسون به این صورت است که در ابتدای روز، **به محض آماده شدن ۵ سفارش، کار خود را با تحویل گرفتن آنها شروع میکند** و بعد از آن، زمانی که همه ی سفارش های روی چرخش را **به ترتیب زمان آماده شدن غذاها** (برای جلوگیری از سرد شدن)، به مشتری ها رساند، برمیگردد و **تمام غذاهایی که آماده شده اند را تحویل گرفته** و تا پایان روز به این کار ادامه میدهد. حال با توجه به اینکه گارسون میتواند در ۸ جهت جغرافیایی حرکت کند به او کمک کنید تا مسیر رساندن هر غذا به مشتری را پیدا کند. بدیهی‌ست که گارسون از مسیرهایی که وسایل تزئینی در آن قرار دارد نمیتواند رد شود و اگر مسیری به میز موردنظر پیدا نکرد عبارت " CHE KHAKI BE SARAM KONAM GHAZAM YAKH KARD" را چاپ کنید.

بعد از مدتی، ریضو و گیلو متوجه شدند که بعضی از مشتری ها به خاطر سرد بودن غذا، ناراضی هستند. در جهت حل این مشکل، ریضو از گارسون خواست تا روش خود را برای توزیع غذا عوض کند و با اینکه همچنان **غذاها را به ترتیب زمان آماده شدن به میزها می رساند** اما، برای **هر غذا، کمینه مسیر ممکن را طی کند**. مثال:



**بخش امتیازی-** گارسون که عاشق حل کردن معما و مسائل مختلف است، با خود فکر میکند ایا میتوان راهی پیدا کرد، که بدون در نظر گرفتن زمان آماده شدن غذاهای روی میزش، **ترتیب رساندن غذاها را طوری انتخاب کند** که **مجموع مسیری که در کل طی میکند کمینه** شود.




## بخش آشپزخانه:

در این بخش به آشپزخونه‌ی رستوران میپردازیم. گیلو، سرآشپز رستوران به کمک ریضو روش عجیبی برای درست کردن غذاها دارد.

روش آنها به این صورت است که برای هر مرحله از مراحل درست کردن غذا باید همه‌ی نیازمندی‌های آن مرحله را انجام داده باشند.

به عنوان مثال اگر درست کردن قرمه سبزی وابسته به خرد کردن سبزی باشد و تنها نیازمندیِ خرد کردن سبزی هم شستن سبزی باشد، برای درست کردن قرمه سبزی، باید به ترتیب شستن سبزی و بعد خرد کردن سبزی و در نهایت پختن قرمه سبزی انجام بشوند. لازم به ذکر است که هر مرحله میتواند بیشتر از یک نیازمندی داشته باشد.

آشپز در هر خط به شما اسم یک مرحله و در ادامه مراحلی که این مرحله به آن ها وابسته است و تلاشی که برای انجام هر کدام لازم است را میدهد.

به عنوان مثال اگر انجام B به انجام شدن A و C نیاز دارد و رسیدن از A به B به ۵ تلاش و رسیدن از C به B به ۴ تلاش نیاز داشته باشد در ورودی خواهیم داشت:

**B A 5 C 4**

**و خروجی آن به شکل زیر خواهد بود :**

**A C B یا C A B**

حال آشپز از شما پیاده سازی مدلی با ویژگی ها و قابلیت های زیر را میخواهد:

- امکان حذف یک غذا و تمام نیازمندی‌های آن.
- دریافت غذایی که از لحاظ تعداد، بیشترین تعداد نیازمندی را دارد ( صرف نظر از زمان مورد نیاز آن‌ها )
- توانایی اضافه کردن یک رابطه‌ی دوتایی (وابستگی) دیگر به مجموعه‌ی موجود.

مثال : اضافه کردن وابستگی **A F 6** به مجموعه مثال بالا

**تغییر خروجی :**

**C F A B یا F A C B یا F C A B**

- توانایی خروجی گرفتن غذایی که در مجموع ( همراه با زمان مورد نیاز برای انجام وابستگی آن‌ها) کمترین زمان و غذایی که بیشترین زمان را بین همه‌ی غذاهای دیگر نیاز دارند.
- توانایی خروجی گرفتن n غذایی که میتوانند تکمیل شوند با دریافت n (تمام وابستگی‌های آن غذا آماده شده اند).

در صورتی که تعداد کمتری از n غذا قابل تکمیل هستند، همان تعداد را نمایش دهد. همچنین هر تعداد غذای خروجی داده شده را به ترتیب زمانی نمایش دهد. ( به ترتیب غذایی که زود تر همه‌ی وابستگیهای آن انجام شده)

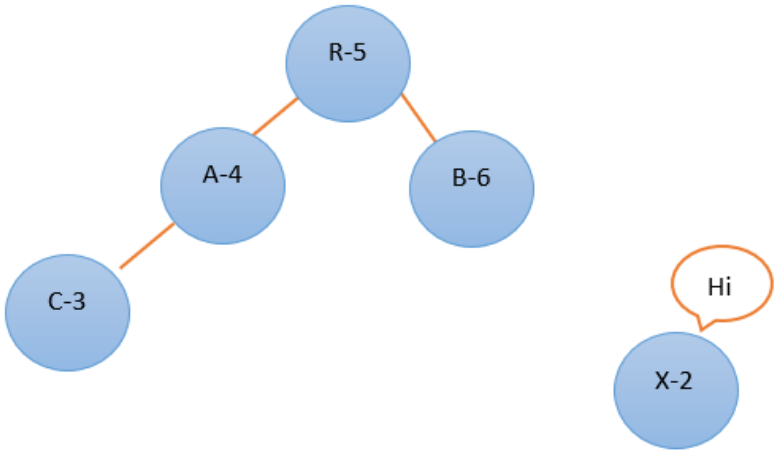
- دقت کنید هنگام اضافه کردن یک مرحله به دستور العمل، در دستورالعمل، دور به وجود نیاید. (درصورت بوجود آمدن دور آشپز نمیداند کدام مرحله را اول انجام دهد و crash میکند! )

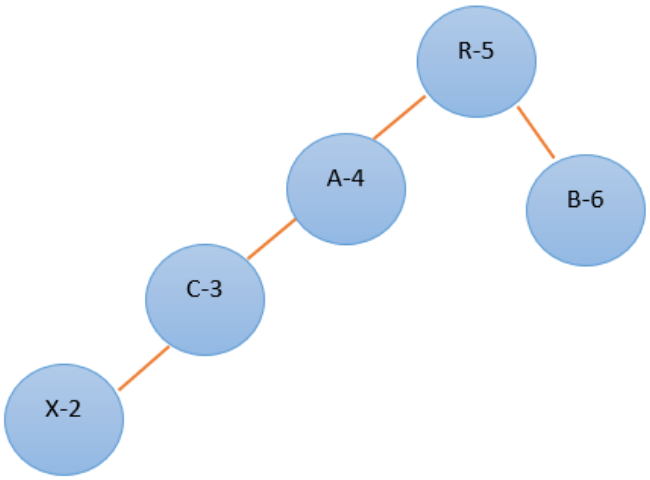
## بخش مهمانی:

بعد از مدتی که کار ریضو و گیلو حسابی گرفت، دانشگاه ها و شرکت های زیادی میخواستند رویدادهایی در رستوران آن‌ها برگذار کنند. گیلو برای سرویس دهی سریع تر به مهمان‌ها، چینش میزها را به شکل درخت دودویی قرار داد . همچنین در بدو ورود به هر مهمان یک عدد به عنوان نوبت میدهد. (تحقیق کنید این چینش نسبت به حالت عادی ، چه مرتبه زمانی ای خواهد داشت)

چینش مهمان‌ها به این صورت است که فردی که نوبتش کوچک‌تر است همیشه سمت چپ فردی‌ست که نوبت بزگتر دارد و برعکس (همان قوانین درخت جست و جوی دودویی) اما نکته اینجاست که وقتی کسی وارد و یا از مجلس خارج میشود، ظاهر مجلس نباید به هم بخورد و آرایش آن حفظ شود، یعنی اگر هر مهمان را یک نود درخت در نظر بگیریم نباید اختلاف جایگاه دو زیرشاخه (ارتفاع) نسبت به نود بالاییشان بیشتر از یک شود و همچنین قوانین نوبت ها را هم رعایت کنند و اگر کسی جای اشتباهی نشست که این نظم را از بین میبرد، سریعا به او تذکر میدهند که جایش را تغییر داده تا نظم جلسه حفظ شود.

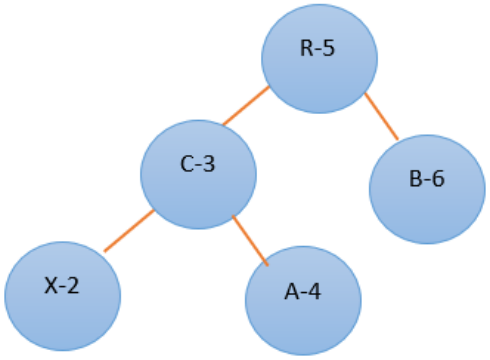
برای مثال اگه چینش مهمان‌ها در جلسه به شکل زیر باشد و خانم X وارد جلسه شود:






خانم X طبق قانون نوبت باید به صورت زیر بنشیند:

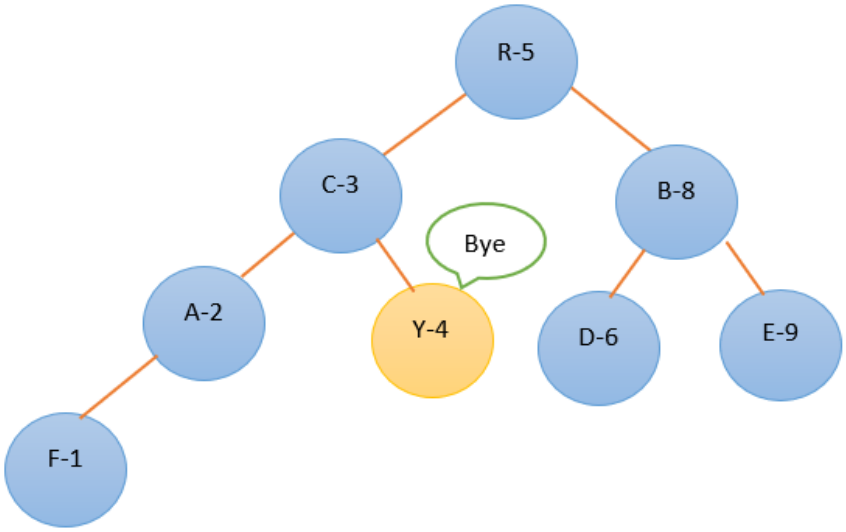
اما این باعث بهم خوردن نظم مجلس میشود و باید به او تذکر بدهند تا در مکانی که آرایش جلسه را حفظ کند



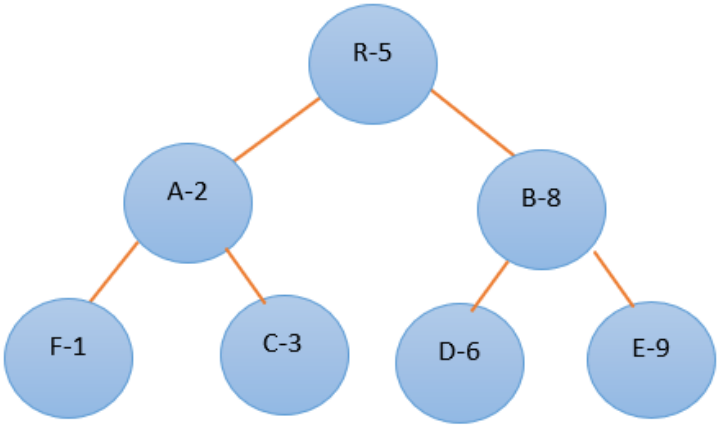

بنشیند، پس مهمان‌ها باید طوری بچرخند تا توازن برقرار شود:

مشکل بعدی وقتی بود که یکی از مهمان‌ها خسته میشد یا میخواست برای کاری به بیرون از جلسه برود و باعث بهم خوردن نظم جلسه میشد. ریضو برای این هم فکری کرد. وقتی یک مهمان از جایش بلند میشد و میرفت ، باید به مهمان‌هایی که در جایگاه های اطراف او هستند تذکر بدهند تا دوباره آرایش جلسه را حفظ کنند.

برای مثال در شکل زیر اگر آقای Y بخواهد از مجلس خارج شود، جای خالی او باعث نامتوازن شدن چینش مهمان‌ها در زیر شاخه‌ی سمت چپ میشود:




در نتیجه باید به بقیه مهمان‌ها تذکر داد تا جوری بچرخند و جایشان را عوض کنند که مجلس به نظم خودش بازگرد:




در حین اینکه که گیلو داشت قوانین مراسم را مینوشت، ریضو سر رسید و به او گفت که این قوانین قبلا توسط دو دانشمند به نام های G.M. Adelson-Velsky و E.M. Landis، ابداع شده و اسمش را هم طبق حروف اول اسم خودشان درخت AVL گذاشتند.

در واقع شما باید در این فاز برای مجالس ، چینش مهمان‌ها را طبق قوانین درخت **AVL** که در بالا مثال زده شد، پیاده سازی کنید. بطور مشخص توابع **insert, delete , search** را پیاده سازی کرده و مرتبه زمانی هرکدام را در بهترین و بدترین حالت تحلیل کنید.

هر نود درخت شما شامل دو ویژگی نام مهمان و شماره نوبت اون هست. ورودی این فاز به شما یک دسته عملیات شامل **insert, delete, search, showTree** است.

- تابع **showTree** همانطور که از اسمش پیداست باید وضعیت نود های درخت را نمایش بدهد. نحوه نمایش به سلیقه خودتان است اما ارتباط بین نودهای درخت باید کاملا مشخص باشد(اگر گرافیکی و زیباتر باشد نمره بهتری به شما تعلق میگیرد)
- تابع **search** شماره نوبت مهمان را میگیرد و با سرچ روی درخت اسم مهمان را برمیگرداند. در صورت عدم وجود عبارت مناسب را برمی‌گرداند.
- توابع **insert** و **delete** هم عملکردشان قبلا توضیح داده‌شد.

مثالی از نحوه ورودی مسئله :

**Insert Ali 2**

**Insert Hasan 3**

**Insert Reza 6**

**ShowTree**

**Insert Negar 10**

**Insert Sina 5**

**Delete Hasan**

**ShowTree**

**Search 5**

**Delete 5**

**Search 5**