بافر چرخشی (حل در 3 ساعت آزمایشگاه)

روزهای ابتدایی سال 1402 بود که متاسفانه دنیای کامپیوتر شخص بزرگی به نام گوردون مور (Gordon Moore) یکی از بنیان‌گذاران شرکت بزرگ Intel و همچنین معرفی کننده قانون مور را از دست داد.

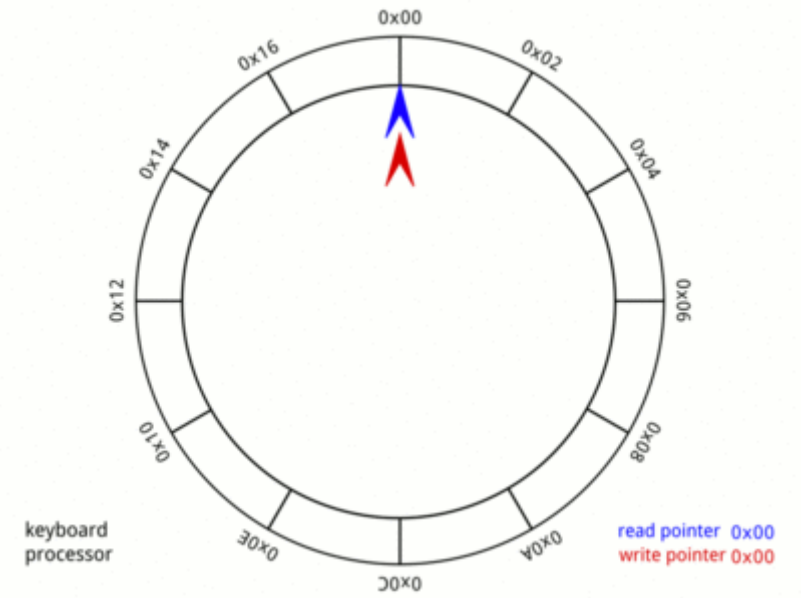
اما از اون روز به بعد اتفاق جالبی افتاد، گوردون هر شب به خواب تعدادی از بچه‌های حوزه کامپیوتر در سراسر دنیا میاد و موردی را مطرح میکنه (گوردون چون الان تو یه دنیایی دیگست از خیلی مسائل آگاهی داره!)

حالا گوردون به خواب یکی از TAهای AP، تو دانشگاه صنعتی اصفهان اومده و گفته که این هفته باید یکی از مسائل بسیار مهم حوزه کامپیوتر یعنی ***بافر*** را بچه‌ها پیاده‌سازی کنن، اونم ***بافر چرخشی***، چون این مسئله یکی از موارد اصلی تو زمینه معماری کامپیوتر و سیستم‌عامله.

خب اما بافر چرخشی چیه؟ - یه حافظه موقته که داده‌ها را از ورودی به صورت پیاپی دریافت میکنه تا موقعی که CPU آزاد شد، بتونه به اون سرویس بده.

حالا کجاها کاربرد داره؟ - برای ورودی Keyboard که کاراکترها به صورت پیاپی وارد میشن یا ورودی Sound سیستم

اگر با این توضیحات درکش نکردین، Gif زیر خیلی خوب بیانش میکنه



یادمه تو همون خواب از گوردون پرسیدم: خب درسته بچه‌ها تازه با مفهوم Template تو برنامه‌نویسی آشنا شدن و میتونن بافر چرخشی را از نوع کاراکتر برای Keyboard و هم از نوع عدد اعشاری برای Sound تعریف کنن اما یه موردی که هست اینه که اونا تازه با بحث Exception هم آشنا شدن و اونا چیکار کنیم؟

گوردون گفت: خب قبوله که بافر چرخشی تو حالت عادی به صورتی کار میکنه که وقتی پر شد، ورودی جدید را با اولین ورودی جایگزین میکنه ولی ما میتونیم استثنا به اونا بگیم که هنگام پر شدن بافر اگه خواست ورودی جدیدی وارد بشه اونا با Exception هندل کنن.

منم پریدم وسط حرفش و گفتم: ایول تازه میتونن هنگام حذف از بافر (سرویس توسط CPU)، اگه بافر خالی بود اونم با Exception هندل کنن.

گوردون ادامه داد: دقیقا و تازه دیگه الانم بچه‌ها به سطحی رسیدن که دیگه بهشون نمی‌خواد بگیم چه کلاس‌هایی میخوان یا دیگه چه Methodهایی را باید داشته باشن، خودشون دیگه تشخیص بدن چی می‌خواد.

در آخر گفت: میدونم بچه‌های دانشگاه صنعتی اصفهان قوی هستن و امیدوارم، ولی نه اطمینان دارم که بچه‌ها میتونن تو 3 ساعت آز به این سوال پاسخ بدن. من دیگه باید برم تو خواب یکی دیگه از بچه‌های حوزه کامپیوتر، سلام منا به تمام اساتید و بچه‌های دانشکده برق و کامپیوتر دانشگاه صنعتی اصفهان برسون. خدانگهدار و پیروز باشید

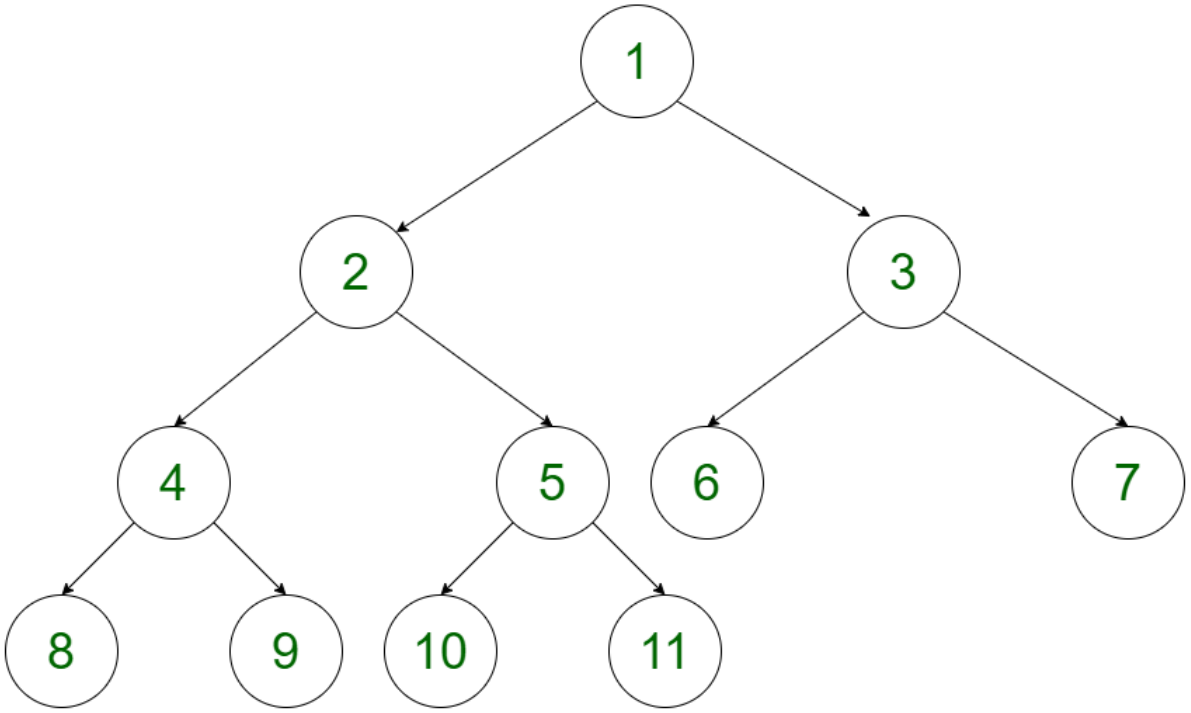
منم گفتم: ممنون گوردون، واقعا مونده بودیم برای این هفته آز چه سوالی بدیم، خدانگهدار و تشکر بابت تمام زحماتی که تو حوزه کامپیوتر کشیدی.

درخت باینری کامل

امروز قراره باهم بریم به جنگل!

اما نه هر جنگلی، یه جنگلی که بچه‌های کامپیوتر باهاش ارتباط میگرن. ما کامپیوتری‌ها یه سری چیزهامون با بقیه فرق میکنه، ما همه چیز را صفر و یکی میبینیم، حتی درخت‌های تو جنگل هم بهش صفر و یکی نگاه میکنیم (البته همیشه اینطور نیست). تو دنیای بقیه آدما وقتی دارن به یه درخت نگاه میکنن، ریشه اون درخت پایینه ولی ما وقتی به درخت نگاه میکنیم، ریشه‌ی درخت را بالا میبینیم!

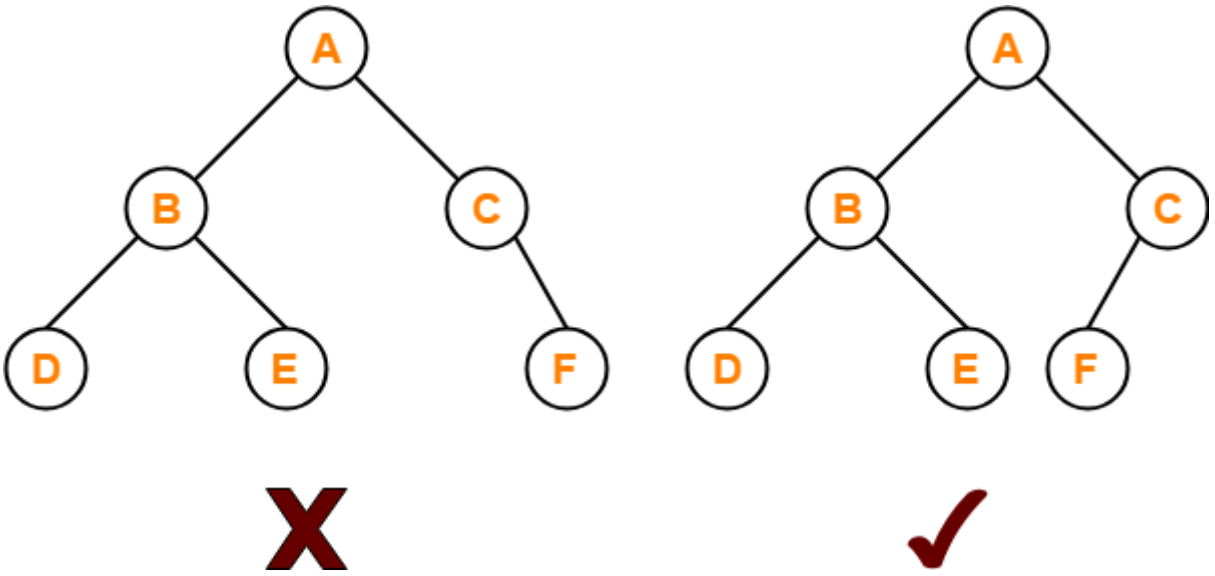
امروز قراره با درخت دودویی یا بهتر بگم Binary Tree آشنا بشین، اما از یک نوع خاصش اونم یعنی درخت باینری کامل. خب همونطور که تو شکل زیر می بینید یک ساختاری وجود داره که در ترم های آینده بیشتر از اون استفاده می کنید. این ساختمان داده اسمش هست درخت باینری!



اول اینکه منظور از باینری اینه که هر گره (Node) تو درخت شما **حداکثر** امکان این رو داره که دو تا فرزند داشته باشه و نه بیشتر! حالا قراره با درخت باینری توی این تمرین چیکار کنیم؟!

اول اینکه شما باید یک **کلاس** برای این ساختار کلی **درخت** طراحی کنید که قابلیت‌های اضافه و حذف کردن یک گره رو داشته باشه. قراره درخت باینری که پیاده سازی می کنید ***کامل*** باشه (دقت کنید، حتما باید ***کامل*** باشه). اگه سرچ کنید **درخت باینری کامل** یا complete binary tree توضیحات بیشتر راجب

این درخت رو می‌تونید پیدا کنید اما من بهتون میگم که درخت باینری کامل یعنی درخت باینری در حالتی که وقتی میخواین یه گره جدید بهش اضافه کنید حتما و لزوما باید اون رو در سمت چپ ترین جای خالی و در اولین ردیف ناکامل درخت قرار بدید. شکل زیر رو ببینید تا بهتر منظور من رو متوجه بشید.



نحوه ساخت درخت هم به این صورت باشه که با استفاده از همان قابلیت اضافه کردن Node بتوان درخت را به طور کامل ایجاد کرد. میشه تو سازنده کلاس، Node ریشه را به وجود آورد (البته تنها راهش نیست).

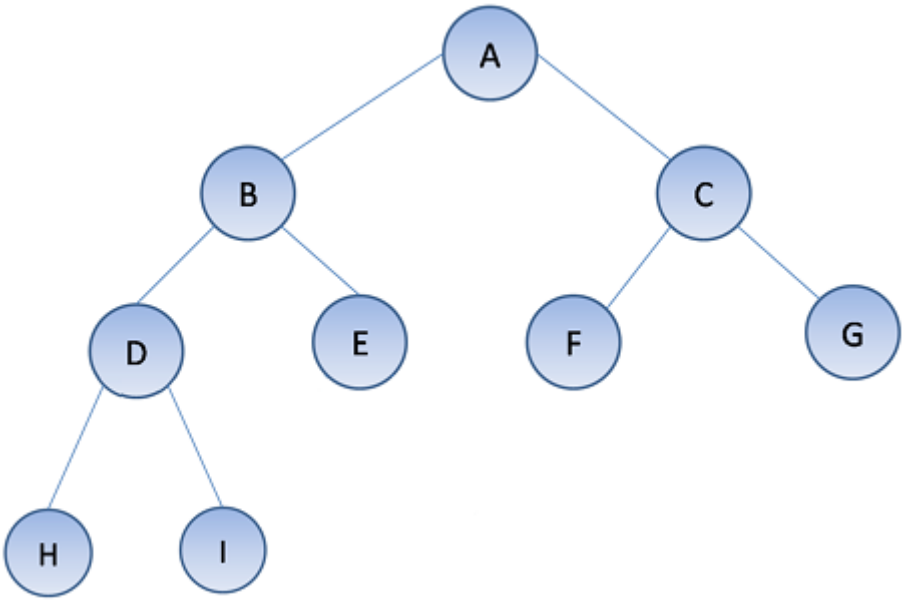
حالا قراره گره‌های درخت باینری ما قابلیت ذخیره کردن انواع مختلف داده‌ها رو داشته باشند یعنی می‌تونیم درخت باینری ای داشته باشیم که گره‌های اون اعداد صحیح یا اعداد اعشاری یا کاراکتر و ... رو نگه داری کنند. **پس نیاز دارید که از مفهوم Template برای ساخت کلاس درخت باینری کاملتون استفاده کنید.**

علاوه بر این باید در داخل کلاستون تابعی رو پیاده‌سازی کنید که یک رشته از کاراکترهای L و R بگیره و بر اساس رشته داده شده درخت باینری شما رو پیمایش و چاپ کنه. حالا منظورمون چی هست دقیقا؟! فرض کنید درخت باینری کامل شکل اول رو ایجاد کردید و تابع شما رشته LLR رو دریافت کرده حالا باید از ریشه درخت شروع کنه و هر بار کاراکتر L رو میبینه به چپ و هر بار کاراکتر R رو میبینه به راست حرکت کنه یعنی در این حالت تابع شما باید رشته 1249 رو چاپ کنه که حاصل پیمایش درخت بر اساس رشته ورودی هست! در کل با هر L به سمت چپ و با R به سمت راست حرکت میکنیم تا نهایت به سطح آخر درخت برسیم.

اگر تعداد رشته کاراکترهای R,L بیش از **ارتفاع درخت** بود (ارتفاع را حساب کنید!)، **بدون پیمایش** درخت بیان کنید که این رشته ورودی غلطه ولی در صورتی که تعداد رشته ورودی درسته، حال با پیمایش امکان چاپ رشته ورودی را بررسی کنید.

نکته دیگه ای که باید پیاده سازی کنید: **اگر رشته دریافتی شما معتبر نیست باید یک exception رو ایجاد کنید.** یعنی اگر مثلا من رشته RRL رو به تابع نظیر کلاس درخت باینری کامل بالا پاس بدم اصلا چنین پیمایشی معتبر نیست پس باید یک استثنا ایجاد کنه. همچنین برای exception یه **کلاس جدید** ایجاد کنید که از exceptionهای C++ استفاده کنه.

همانطور که در گسسته هم خوندید، درخت را می‌توان به نوع‌های متفاوتی پیمایش کرد، سه نوع پیمایش برای درخت وجود دارد (preorder, inorder, postorder) به این منظور که به فرض برای پیمایش preorder ابتدا نود ریشه، سپس فرزند چپ در نهایت فرزند راست را خواهیم دید. الان در درخت زیر میتوانید انواع پیمایش‌های مختلف را مشاهده نمایید.



- دنباله پیمایش پیش‌ترتیب: A, B, D, H, I, E, C, F, G
- دنباله پیمایش میان‌ترتیب: H, D, I, B, E, A, F, C, G
- دنباله پیمایش پس‌ترتیب: H, I, D, E, B, F, G, C, A

شما می‌بایست تمام این پیمایش‌ها را در کلاس خود پیاده کنید.

اما حیف نیست درخت به این خوبی درست کردیم ولی اعضا آن مرتب نباشن. میخواهیم تمام نودهای داخل درخت را مثل شکل اول مرتب کنیم (ساده‌ترین روش برای اینکار اینه که کل درخت را بررسی کنی

و min را پیدا کرده و با نود اول جابجا کنیم، حالا دوباره کل درخت را بررسی میکنی و دومین min را پیدا میکنی و با نود مربوطه جایگزین میکنید.) البته این روش ساده‌ترین راه برای مرتب سازی هست اما شما می‌توانید از هر روش دیگه‌ای که خواستید استفاده کنید.

بدانید و آگاه باشید که هر Node باید متغیری با تایپ تعریف شدهٔ خود نگه دارد و اگر دیتایی با تایپ متفاوت در آن ریخته شد، باید با کمک Exception ها، اروری نشان دهید که به کاربر بفهماند که تایپ داده‌ای که وارد کرده است غلط است. راهنمایی: throw را در کلاس Node باید بنویسید نه در کلاس Tree

در نهایت یک متد برای چاپ درخت طراحی نمایید، برای سادگی میتوانید هر سطر از درخت را در یک خط چاپ نمایید و هر عنصر را با فاصله از قبلی چاپ کنید (در صورتی که درخت را مانند درخت واقعی شکل بالا چاپ نمایید، نمره اضافی در بر خواهد داشت). الان به فرض برای شکل بالا حداقل چیزی که باید چاپ کنید به این صورته:

```
A
B C
D E F G
H I
```