به نام خداوند بخشنده ی مهربان

راه حل دوره ۲۱ روز اول

به همت حامد صالح

سوال ۱:

برای حل این سوال تابع process را بررسی می کنیم. for داخلی که الکی است و در واقع k برابر y می شود و هدف این for صرفا زمانبر کردن تابع process است :) دو خط بعدی هم ارز با دوبرابر کردن y و اضافه کردن b به آن است که b برعکس (not) باقی مانده x به ۲ است. پس اگر x و y را در مبنای ۲ در نظر بگیریم، در هر مرحله not بیت کم ارزش x به سر y اضافه می شود. یعنی y هم از نظر ترتیب بیت ها برعکس x خواهد بود و هم خود بیت ها. تنها نکته باقیمانده این است که for بیرونی از x/2 شروع می شود و شرط پایانش x > 1 است، یعنی بیت اول و آخر x در نظر گرفته نمی شوند. در کل اگر x در مبنای ۲ به صورت x_0, x_1, ..., x_n باشد، y برابر با not(x_n-1), ..., not(x_1) خواهد شد.

irprogramming.ir

حال ما که می خواهیم delta^2 را تولید کنیم، کافیست ترتیب بیت هایش در مبنای ۲ را برعکس کرده و آن ها را not کنیم، یک ۱ به سرش و یک ۰ به تهش اضافه کنیم. یعنی:

[y_0, y_1, ..., y_n --> 1, not(y_n), ... not(y_0), 0 [y = delta^2

سوال ۲:

با کمی دقت می توان فهمید که ضرب ۱۱ عدد اول نخست، ۲۰۴۸ مقسوم علیه دارد، که این عدد برابر است با ۲۰۰۵۶۰۴۹۰۱۳۰. پس جواب حداکثر برابر این عدد است و بزرگترین مقسوم علیه اولش ۳۱ است (۱۱ امین عدد اول). نکته مهم دیگر این است که توان اعداد اول در جواب بهینه نزولی است. برای مثال اگر توان ۳ از توان ۷ کمتر باشد، می توان توان هایشان را جابجا کرد و جواب کمتر می شود، در صورتی که تعداد مقسوم علیه ها تغییر نمی کند. با توجه به این نکات می توان به طور بازگشتی و پس گرد (backtrack) جواب های بهینه را ساخت و مقایسه نمود و بهترین جواب را پیدا کرد. به این صورت که از ۱۱ امین عدد اول به عقب می رویم و توان هر کدام را تایین می کنیم، تا به عدد نخست برسیم که در آن جا جوابی به دست می آید که اگر حداقل ۲۰۱۱ مقسوم علیه داشته باشد، می تواند جواب بهینه باشد.

دقت کنید در بهینه سازی زمانی الگوریتم بعضی از نکات ریز می توانند خیلی مهم باشند. برای مثال در اینجا، بازگشت از تابع هنگامی که عدد فعلی از بهترین جواب به دست آمده تا کنون بیشتر می شود (زیرا جوابی که بدست خواهد آمد، کمتر نخواهد شد و بهینه نخواهد بود)، خیلی مؤثر است. بهینه سازی دیگر، افزایش توان تمام اعداد اول ۱ تا iام در هنگامی است که می خواهیم توان i را زیاد کنیم. (به علت نزولی بودن توان ها) بدون این دو بهینه

سازی نمی توان جواب را با سرعت خوبی به دست آورد ولی این دو بهینه سازی باهم، تعداد حالت هایی که قرار است بررسی شوند را به مقدار خوبی کم می کنند و جواب سریع به دست می آید.

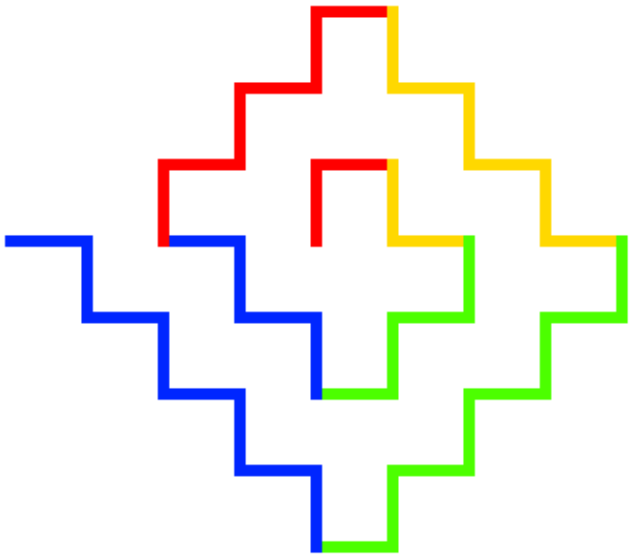
سوال ۳:

این شیوه ی شماره گزاری راس های درخت، به ما یک درخت دودویی می دهد که به آن هرم می گویند. (برای ذخیره سازی هرم ها نیز معمولا از این روش استفاده می شود.) این شیوه خواص خوبی دارد که با نگاه کردن به اعداد روی درخت در مبنای دو می توان متوجه آنها شد. یکی از این خواص این است که نمایش مبنای دو اعداد نشان دهنده ی مسیر ریشه تا آن راس است. چون درخت دودویی است و هر راس دو بچه چپ و راست دارد (راس k حداکثر دو بچه دارد:۲k و 2k + 1) می توان مسیر ریشه تا هر راسی به صورت دنباله ای از ۰ و ۱ ها نشان داد. (اگر بیت iام ۰ باشد یعنی در حرکت iام به چپ رفتیم و اگر ۱ باشد به راست) اعداد روی راس های درخت هم دقیقا به همین شکل مسیر را نگه داشته اند. حال اگر بخواهیم فاصله دو عدد خاص در این درخت را بدست آوریم، کافیست فاصله آن دو تا ریشه را با هم جمع کرده و دو برابر طول اشتراک مسیر آن دو از ریشه را از آن کم کنیم (چون این قسمت اشتراک دو بار حساب شده ولی نباید حساب شود). فاصله یک عدد تا ریشه که برابر تعداد ارقام عدد در مبنای ۲ است (همانطور که گفتیم نمایش مبنای دو نشان دهنده مسیر از ریشه است، پس فاصله از ریشه برابر طول عدد در مبنای دو است)، طول اشتراک مسیر از ریشه دو عدد نیز برابر با طول بزرگترین پیشوند (از سمت چپ) مشترک دو عدد در مبنای دو است، که هر دوی این ها را به راحتی می توان محاسبه نمود.

برای به دست آوردن جواب نهایی، ابتدا با استفاده از غربال اراتستن تمام اعداد اول ۱ تا 2delta (چون ممکن است جمع دو تا از این اعداد اول از delta بیشتر شود) را به دست می آوریم که تعدادشان ۳۱۶۴ تاست. (به ازای delta = 29123) سپس به ازای هر جفت که شرایط مساله را دارد، فاصله شان، که توضیح داده شد چگونه به دست می آید، را در هم ضرب می کنیم تا جواب به دست آید.

سوال ۴:

با توجه به الگوی نشان داده در شکل به راحتی می توان جای آیدین بعد ۲۰۰۰۰۰۰۰۰۰ مرحله را به دست آورد.



سوال ۵:

در ابتدا با استفاده غربال اراتستن شماره چوب هایی که داریم را به دست می آوریم. طبق شروط مساله اگر بخواهیم دنباله ای از چوب ها را در چوب با شماره i فرو کنیم، باید هم شماره تمام چوب های این دنباله از i کمتر باشد، هم رقم یکان راست ترین عدد دنباله (یکان کل دنباله) برابر چپ ترین رقم i باشد. برای ارضای شرط اول به ترتیب روی چوب ها حرکت می کنیم و به هر عدد i که می رسیم، بلند ترین میله ای که راست ترین چوبش i باشد را به دست می آوریم. این بلند ترین میله دو حالت خواهد داشت: ۱- فقط شامل چوب با شماره i باشد. ۲- شامل چوب i و دنباله ای از چوب های ۱ تا i - 1 باشد که قابل فرو رفتن در i باشد. نکته مهم این است که دو دنباله با یکان یکسان از نظر چوبی که قرار است در آن فرو روند، هیچ فرقی با هم ندارند. پس اگر ما به ازای هر یکان مثل j، طول بزرگترین دنباله ای که یکان اش j را در d_j نگه داریم، جواب حالت ۲ برای چوب شماره i برابر d_x + l - 1 است که در آن x چپ ترین رقم عدد i و l طول عدد i است. هم چنین این عدد ممکن است d_i%10 را هم زیاد کند که آن را هم لحاظ می کنیم. جواب نهایی بزرگترین مقدار بین d_j هاست.