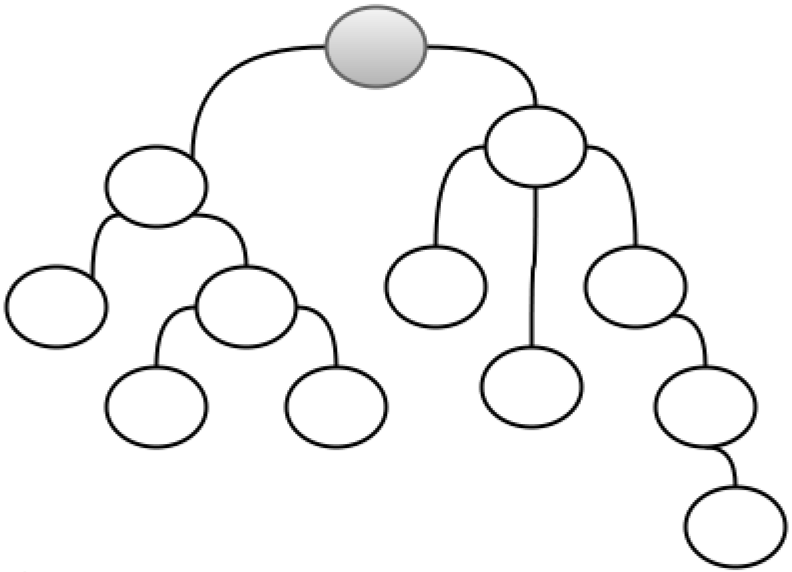
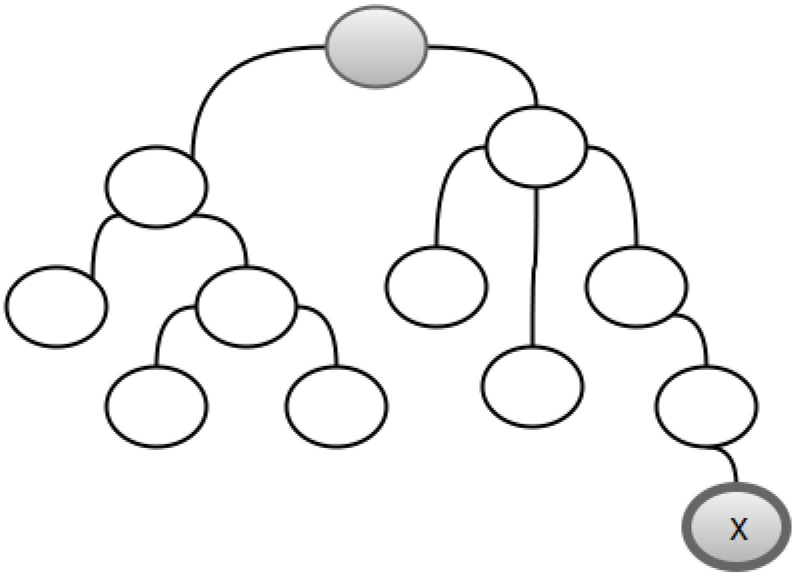
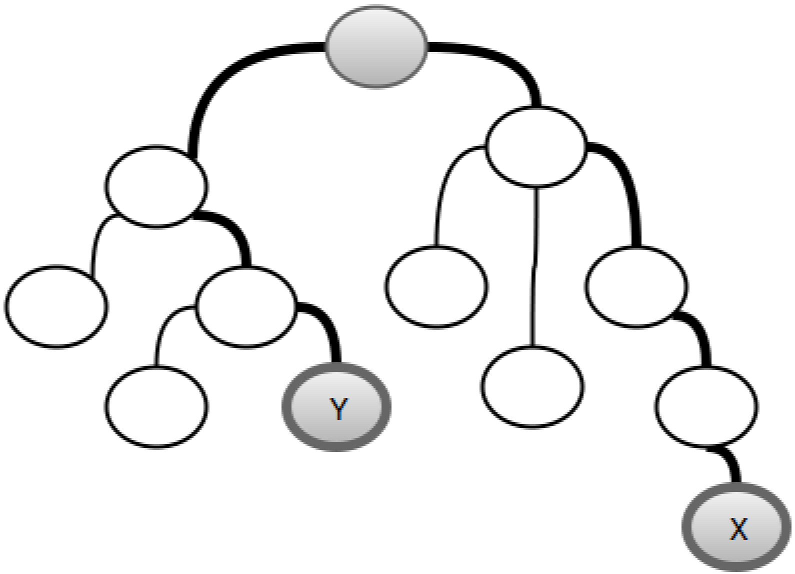
ট্রি হলো এমন একটা আনডিরেক্টেড গ্রাফ যেটার সব নোড থেকে সব নোডে যাওয়া যায় এবং কোনো সাইকেল নেই। এখন আমাদের ট্রি এর সবথেকে দূরের দুটা নোড খুজে বের করতে হবে, একেই বলা হয় ট্রি এর ডায়ামিটার।
মনে করো কিছু কম্পিউটারের মধ্যে নেটওয়ার্ক কেবল লাগানো হয়েছে নিচের ছবির মতো করে। এখন তুমি জানতে চাইতেই পারো কোন দুটি কম্পিউটার সবথেকে দূরে আছে।



এটা বের করা খুব সহজ, এজন্য তোমার জানতে হবে বিএফএস বা ডিএফএস এর যে কোন একটা। আনডিরেক্টেড ট্রি তে **যেকোন নোড**কেই রুট ধরা যায়, আমরা মনে করি উপরের ধূসর নোডটা ট্রি এর রুট।
আমাদের প্রথম কাজ কাজ হলো রুট হতে সবথেকে দূরের নোডটা খুজে বের করা। সেই নোডটাকে মনে করি x । একাধিক নোডের দূরত্ব সবথেকে দূরের নোডের দূরত্বের সমান হলেও সমস্যা নেই, যেকোন একটাকে সিলেক্ট করতে হবে। এই কাজটা আমরা ডিএফএস/বিএফএস চালিয়ে বের করতে পারি।



এখন ২য় কাজ হলো X নোড থেকে শুরু আরেকটি ডিএফএস/বিএফএস চালিয়ে X থেকে সবথেকে দূরের নোড খুজে বের করা। মনে করি নোডটা হলো Y।

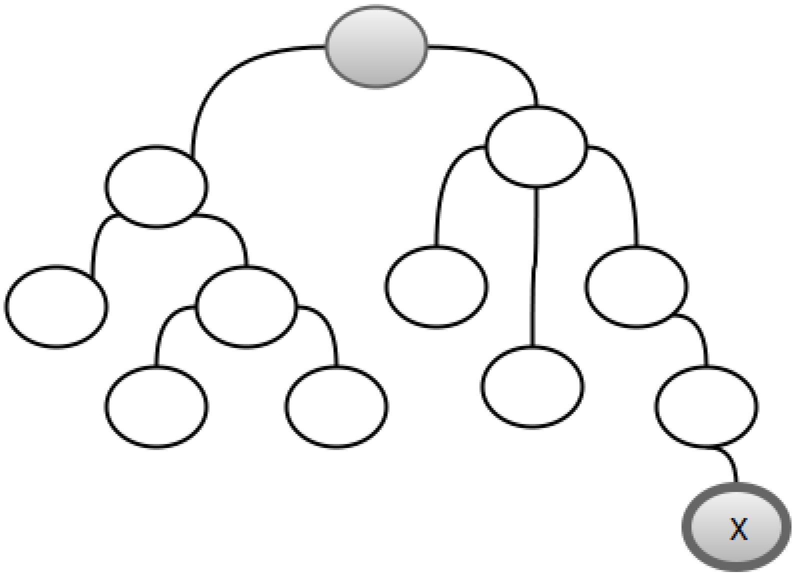


X আর Y এর মধ্যকার দূরত্বই ট্রি এর ডায়ামিটার! উপরের ছবিতে ডায়ামিটার ৭।

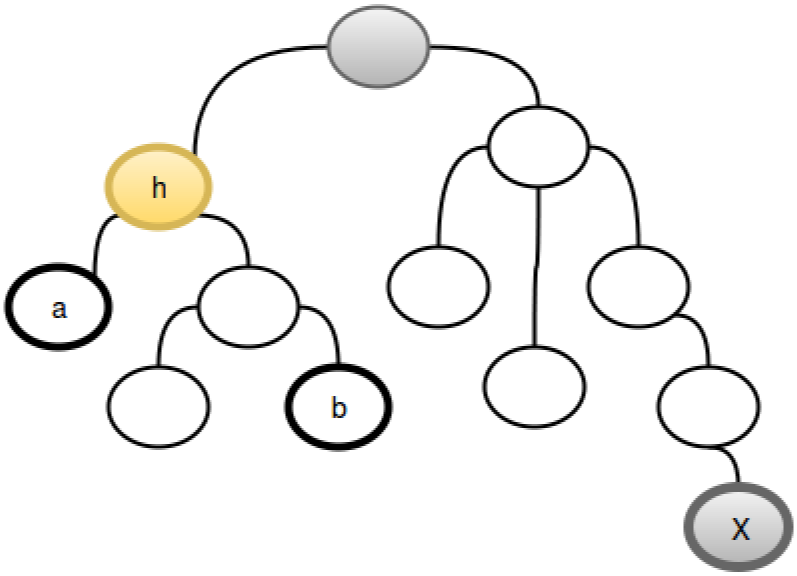
প্রমাণ:

আমরা যদি প্রমাণ করতে পারি ১ম ধাপে খুজে পাওয়া X সবসময়ই ডায়ামিটারের একটা প্রান্ত হবে তাহলেই অ্যালগোরিদমটা প্রমাণ হয়ে যায়। কারণ X যদি নিশ্চিতভাবে ডায়ামিটারের একটা প্রান্ত হয় তাহলে ২য় ধাপটা অবশ্যই সঠিক, X থেকে সবথেকে দূরের নোডই হবে ট্রি এর ডায়ামিটার।

এটা আমরা প্রমাণ করবো "প্রুফ বাই কনট্রাডিকশন" এর সাহায্যে। এটা বহুল ব্যবহৃত একটা পদ্ধতি। আমরা প্রমাণ করতে চাই X নিশ্চয়ই কোনো ডায়ামিটারের প্রান্ত। কিন্তু তা যদি না হয়, অর্থাৎ X কোনো ডায়ামিটারের প্রান্ত না হলে এমন একটা ডায়ামিটার থাকবে যার দুই প্রান্ত (ধরি) a এবং b, এখানে a এবং b যেকোনো দুটি নোড হতে পারে(X ছাড়া) । a-b ডায়ামিটার ব্যবহার করে আমরা দেখাবো এমন আরেকটা ডায়ামিটার পাওয়া সম্ভব যেটা a-b এর চেয়ে বড় বা সমান এবং যার এক প্রান্তে X আছে। (উল্লেখ্য, X হলো রুট থেকে সবচেয়ে দূরের একটা নোড)।



এখন **আর্বিট্রালি(arbitrarily)** দুটি নোড a আর b সিলেক্ট করি। নোড দুটির মধ্যকার পাথ রুটের কাছে যে নোডে এসে মিলবে সেটার নাম দেই h, অর্থাৎ h হলো নোড দুইটার কমন অ্যানসেস্টর। (অনেকেই হয়তো বুঝে গেছো আমরা এখানে lowest common ancestor এর কথা বলছি)।



কিন্তু রুট থেকে b এর দূরত্ব, রুট থেকে X এর দূরত্বের কম বা সমান হতে বাধ্য, বেশি হবেনা কারণ রুট থেকে সবথেকে দূরের নোড হলো X।

অর্থাৎ:

$$distance(root,b)<=distance(root,X)$$

আবার এটাও নিশ্চিত যে রুট থেকে b যত দূরে, h থেকে b এর দূরত্ব তার থেকে কম বা সমান। (সমান হবে যদি h আর রুট একই নোড হয় অর্থাৎ a আর b যদি রুটের দুইপাশে থাকে)

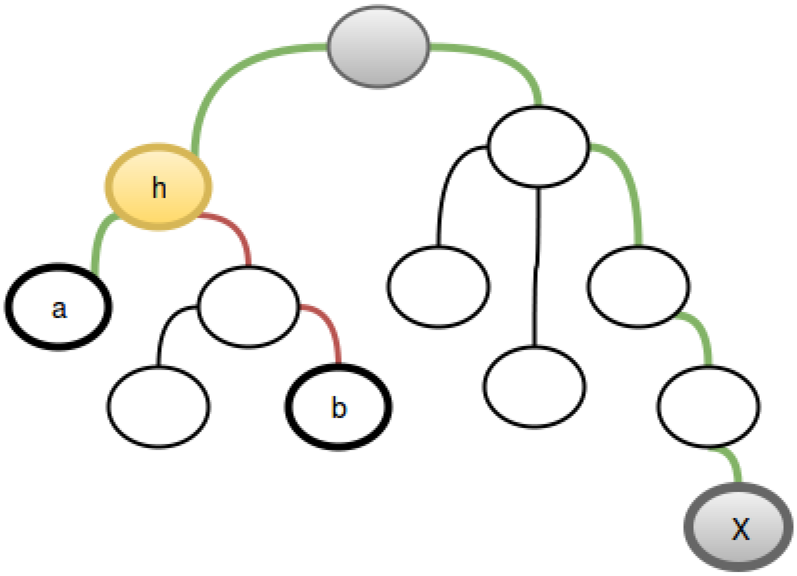
$distance(h,b) <= distance(root,b)$

তাহলে,

$distance(h,b) <= distance(root,b) <= distance(root,X)$

$distance(h,b) <= distance(root,X)$

তাহলে আমরা h-b পাথটা root-X পাথ দিয়ে রিপ্লেস করে দিলে এবং h-x এর মধ্যে পাথ বসিয়ে দিলে(যদি h আর x সমান না হয়) অবশ্যই আগের সমান বা আগের থেকে বড় একটা পাথ পাবো!



তারমানে আমরা যে X কে বাদ দিয়ে যেই দুইটা নোডই সিলেক্ট করিনা কেন, X কে নিয়ে তার থেকে বড় বা সমান একটা পাথ পাওয়া যাবে। তারমানে X অবশ্যই ডায়ামিটারের একটা প্রান্ত!

কমপ্লেক্সিটি:
বিএফএস/ডিএফএস এর কমপ্লেক্সিটির সমান: O(V+E) যেখানে V হলো নোডসংখ্যা এবং E হলো এজ সংখ্যা।

এই অ্যালগোরিদমটা জেনারেল গ্রাফে কাজ করবেনা, শুধু ট্রি এর ক্ষেত্রে করবে। জেনারেল গ্রাফে লংগেস্ট পাথ বের করার প্রবলেম এন-পি হার্ড, আর ডায়ামিটার বের করার প্রবলেম লংগেস্ট পাথ প্রবলেমেরই স্পেশাল কেস।

Farthest Nodes in a Tree
Farthest Nodes in a Tree (II)
হ্যাপি কোডিং!

কৃতজ্ঞতা স্বীকার:

http://stackoverflow.com/questions/20010472/proof-of-correctness-algorithm-for-diameter-of-a-tree-in-graph-theory
http://apps.topcoder.com/forums/?module=Thread&threadID=668470&start=0&mc=12#1216875