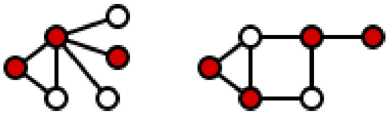
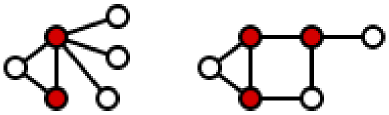
মিনিমাম ভারটেক্স কভার প্রবলেম(রিপোস্ট)

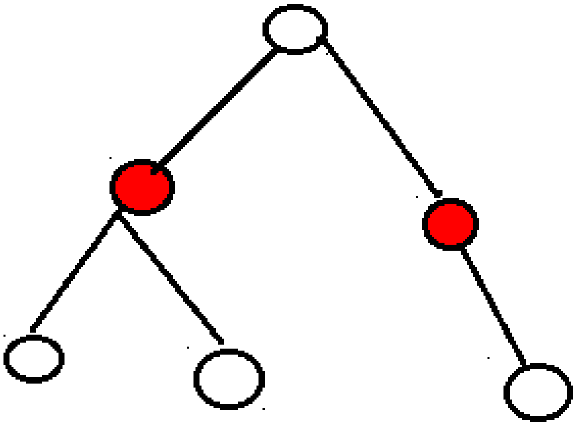
(মিনিমাম ভারটেক্স কভার নিয়ে অনেক আগে লিখেছিলাম,লেখাটা বুঝতে অনেকের সমস্যা হওয়াতে নতুন করে লেখাটি এডিট করলাম । আশা করি এখন বুঝতে সমস্যা হবেনা)



মিনিমাম ভারটেক্স কভার একটি ক্লাসিক গ্রাফ প্রবলেম। ধরা যাক একটি শহরে কিছু রাস্তা আছে,এখন প্রতি রাস্তায় মোড়ে আমরা পাহারাদার বসাতে চাই। কোনো নোডে পাহারাদার বসালে সে নোডের সাথে যুক্ত রাস্তাগুলো একাই পাহারা দিতে পারে। উপরের ছবিতে নোডগুলো হলো রাস্তার মোড়। এখন সব কয়টা রাস্তা পাহারা দিতে নূন্যতম কয়জন পাহারাদার দরকার? ছবিতে লাল নোডগুলোতো পাহারাদার বসানো হয়েছে। এটা অপটিমাল না,নিচের ছবির মত বসালে পাহারাদার কম লাগত:



এটি একটি NP-hard প্রবলেম,অর্থাৎ এই প্রবলেমের কোনো পলিনমিয়াল টাইম সলিউশন নেই। তবে গ্রাফটি যদি **Tree** হয় অর্থাত n-1 টা edge থাকে আর কোনো সাইকেল না থাকে তাহলে ডাইনামিক প্রোগ্রামিং বা ম্যাক্স ফ্লো/বাইপারটাইট ম্যাচিং এর সাহায্যে প্রবলেমটি সলভ করা সম্ভব।



Minimum vertext Cover in a Tree Graph

ডাইনামিক প্রোগ্রামিং সলিউশনটা আমি বিস্তারিত লিখছি,তারপর ম্যাক্স ফ্লো/বাইপারটাইট ম্যাচিং দিয়ে কিভাবে করতে হয় লিখবো।

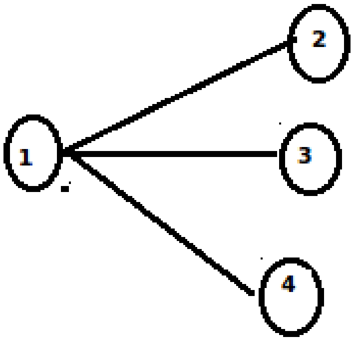
ডিপি সলিউশনে ২টি কেস আমাদের লক্ষ্য করতে হবে:

> *১. কোনো নোডে পাহারাদার না বসালে তার সাথে সংযুক্ত সব নোডে অবশ্যই পাহারাদার বসাতে হবে,এছাড়া সব রাস্তা কভার হবে না। অর্থাৎ যদি u আর v সংযুক্ত থাকে তাহলে u তে পাহারাদার না বসালে v তে অবশ্যই বসাতে হবে।*
> *২. কোনো নোডে পাহারাদার বসালে সংযুক্ত নোডগুলোতে পাহাদার বাসানো বাধ্যতামূলক না তবে বসালে লাভ হতে পারে। তাই u তে পাহারাদার বসালে v তে পাহারাদার একবার বসিয়ে এবং একবার না বসিয়ে দেখবো কোনটা লাভজনক*

সব ডিপির প্রবলেমের মতো এখানেও একটা রিকার্সিভ ফাংশন ডিফাইন করবো। আমাদের স্টেট হবে বর্তমানে কোন নোডে আছি,এবং সেই নোডে কোনো পাহারাদার বসানো হয়েছে নাকি।

> *F(u,1) = বর্তমানে u নম্বর নোডে আছি এবং এই নোডে **পাহারাদার আছে**। f(u,1) রিটার্ণ করবে বাকি নোডগুলোতে মোট পাহারাদার সংখ্যা।*
> *F(u,0) = বর্তমানে u নম্বর নোডে আছি এবং এই নোডে **পাহারাদার নাই**। f(u,0) রিটার্ণ করবে বাকি নোডগুলোতে মোট পাহারাদার সংখ্যা।*

ধরি ১ নম্বর নোডের সাথে ২,৩,৪ নম্বর নোড যুক্ত।



বুঝাই যাচ্ছে ১ নম্বর নোডে পাহারা না বসালে অবশ্যই ২,৩,৪ সবগুলোয় পাহারা বসাতে হবে। তাহলে আমরা বলতে পারি:

> *F(1,0)=F(2,1)+F(3,1)+F(4,1) + 0 , অর্থাত ১ এর সাথে সংযুক্ত সব নোডগুলোতে পাহারা বসালে প্রয়োজনীয় মোট পাহারাদার সংখ্যা।*

সবশেষে ০ যোগ করছি কারণ বর্তমান নোডে পাহারাদার বসাইনি।

এবার F(1,1) এর মান বের করি। ১ নম্বর নোডে পাহারা বসালে সংযুক্ত নোডগুলোতে পাহারা বসালেও চলে,না বসালেও চলে,তবে যেটা অপটিমাল রেজাল্ট দেয় সেটা আমরা নিব:

> *F(1,1)=1+min (F(2,1),F(2,0)) +min (F(3,1),F(3,0)) +min (F(4,1),F(4,0))*

১ নম্বর নোডে পাহারাদার বসাচ্ছি তাই সবশেষে ১ যোগ হচ্ছে,প্রতি নোডে একবার পাহারা বসিয়ে,আবার না বসিয়ে দেখছি কোনটা অপটিমাল।

একটা ব্যাপার লক্ষ রাখতে হবে যে প্যারেন্ট নোড নিয়ে কখনো হিসাব করবোনা। উপরের ছবিতে ১ থেকে ২ এ গেলে parent[2]=1,তাই ২ থেকে আবার ১ নম্বর নোডে যাবোনা।

এবার base case এ আসি। কোনো নোড থেকে নতুন কোনো নোডে যাওয়া না গেলে 1 বা 0 রিটার্ন করে দিতে হবে,পাহারাদার বসালে 1,না বসালে 0। কোনো ট্রি তে একটি মাত্র নোড থাকলে ১ রিটার্ণ করতে হবে(কিছু প্রবলেমে ০ ও রিটার্ণ করতে হতে পারে)।

Spoj এর PT07X(vertex cover) প্রবলেমটি straight forward প্রবলেম। এটার জন্য আমার কোডটা এরকম:

C++

```
1   #define MAXN 100002
2   int dp[MAXN][5];
3   int par[MAXN];
4   vector<int>edges[MAXN];
5
6   int f(int u,int isGuard)
7   {
8      if(edges[u].size()==0)return 0;
9      if(dp[u][isGuard]!=-1) return dp[u][isGuard];
10     int sum=0;
11     for(int i=0;i<(int)edges[u].size();i++)    {
12        int v=edges[u][i];
13        if(v!=par[u]){
14           par[v]=u;
15           if(isGuard==0) sum+=f(v,1);
16           else sum+=min(f(v,1),f(v,0));
17        }
18     }
19     return dp[u][isGuard]=sum+isGuard;
20  }
21
22  int main()
23  {
24        memset(dp,-1,sizeof(dp));
25     int n;
26     scanf("%d",&n);
27     for(int i=1;i<n;i++){
28        int u,v;
29        scanf("%d%d",&u,&v);
30        edges[u].push_back(v);
31        edges[v].push_back(u);
32     }
33     int ans=0;
34     ans=min(f(1,1),f(1,0));
35     printf("%d\n";,ans);
36     return 0;
37  }
```

আমি ট্রি এর root সবসময় ১ ধরে কোড লিখেছি। ৩৮ নম্বর লাইনে মেইন ফাংশনে root এ পাহারাদার একবার বসিয়ে আর একবার না বসিয়ে অপটিমাল রেজাল্ট টা নিচ্ছি।
ফাংশনে u হলো current node,isguard কারেন্ট নোডে পাহারাদার আছে নাকি নাই সেটা নির্দেশ করে।
১০ নম্বর লাইনে ট্রি এর সাইজ ১ হলে ১ রিটার্ণ করে দিয়েছি।
১৩ নম্বর লাইনে লুপের ভিতর current নোড থেকে সবগুলো child নোডে যাচ্ছি। কারেন্ট নোডে পাহারাদার না থাকলে পরেরটায় বসাচ্ছি,আর থাকলে ২ভাবেই চেষ্টা করছি। ১৫ নম্বর লাইনের কন্ডিশন দিয়ে প্যারেন্ট নোডে যেতে দিচ্ছিনা।
সবশেষে sum+isGuard রিটার্ণ করছি। অর্থাত কারেন্ট নোডে পাহারাদার থাকলে 1 যোগ করছি,নাহল 0।

মোটামুটি এই হলো ডিপি সলিউশন। ট্রি তে সাইকেল না থাকায় এটা অবশ্যই বাইপারটাইট গ্রাফ। ১৯৩১ সালে Dénes Kőnig প্রমাণ করেন কোনো বাইপারটাইট গ্রাফে maximum matching=minimum vertext cover। এটা গ্রাফ থিওরির অনেক min-max থিওরেমের একটা যেখানে কিছু একটা ম্যাক্সিমাইজ করলে অন্য আরেকটা কিছু মিনিমাইজ হয়। তুমি যদি ম্যাক্সিমাম ম্যাচিং এর অ্যালগোরিদম জানো তাহলে ট্রি টা বাইকালারিং করে ম্যাচিং বের করলেই ভারটেক্স কভার বের হয়ে যাবে। কোড সহজ হলেও complexity বেড়ে যাবে,তাই নোড বেশি থাকলে কাজ করবেনা। আবার ম্যাক্সিমাম ম্যাচিং যেহেতু ম্যাক্স-ফ্লো এর একটি ভ্যারিয়েশন তাই ফ্লো চালিয়েও সমাধান করা সম্ভব।

এরকম আরেকটা প্রবলেম uva-10243(fire fire fire)। আমি প্রথমে এটা সমাধান করে পরে spoj এর টা করেছি।