

## শাফায়েতের ব্লগ

### গ্রাফ থিওরিতে হাতেখড়ি ৬: মিনিমাম স্প্যানিং ট্রি(prim's mst)

২০১১-০৮-০৪ ১৯:০৮:০৫ শাফায়েত

(সিরিজের অন্যান্য পোস্ট)

মিনিমাম স্প্যানিং ট্রি গ্রাফ থিওরির খুব গুরুত্বপূর্ণ একটি অংশ। এই টিউটোরিয়ালে গ্রাফ থেকে মিনিমাম স্প্যানিং ট্রি নির্ণয়ের একটি অ্যালগোরিদম নিয়ে আলোচনা করবো।

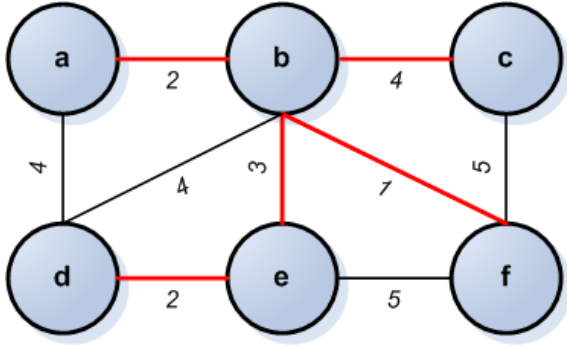
একটি গ্রাফ থেকে কয়েকটি নোড আর edge নিয়ে নতুন একটি গ্রাফ তৈরি করা হলে সেটাকে বলা হয় subgraph। স্প্যানিং ট্রি হলো এমন একটি সাবগ্রাফ যেটায়:

- \* মূল গ্রাফের সবগুলো নোড আছে।
- \* সাবগ্রাফটি একটি ট্রি। ট্রিতে কখনো সাইকেল থাকেনা, edge থাকে  $n-1$  টি যেখানে  $n$  হলো নোড সংখ্যা।

একটি গ্রাফের অনেকগুলো স্প্যানিং ট্রি থাকতে পারে, যে ট্রি এর edge গুলোর weight যোগফল সব থেকে কম সেটাই মিনিমাম স্প্যানিং ট্রি।

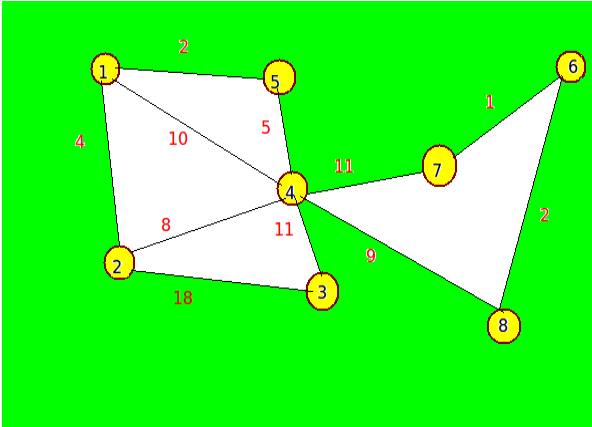
( $n$  টি নোডকে সর্বনিম্ন  $n-1$  টি edge দিয়ে একসাথে যুক্ত করা সম্ভব, তাই  $n-1$  টির বেশি এজ আমরা নিবোনা)

মনে করি নিচের গ্রাফের প্রতিটি নোড হলো একটি করে বাড়ি। আমাদের বাড়িগুলোর মধ্যে টেলিফোন লাইন বসাতে হবে। আমরা চাই সবথেকে কম খরচে লাইন বসাতে। edge গুলোর weight লাইন বসানোর খরচ নির্দেশ করে:



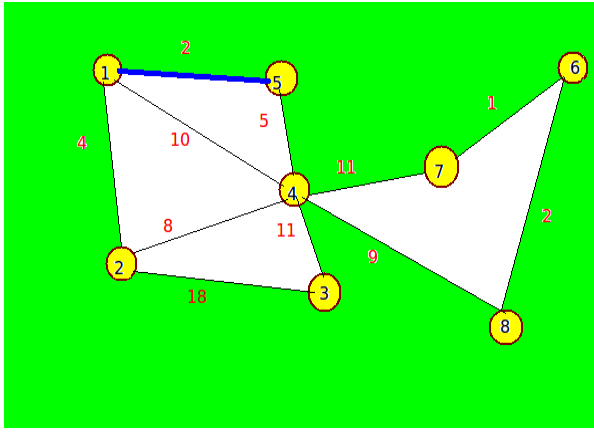
আমরা অনেক ভাবে লাইন বসাতে পারতাম। ছবিতে লাল এজ দিয়ে টেলিফোন লাইন বসানোর একটি উপায় দেখানো হয়েছে। টেলিফোন লাইনগুলো একটি সাবগ্রাফ তৈরি করেছে যেটায় অবশ্যই  $n-1$  টি এজ আছে, কোনো সাইকেল নেই কারণ অতিরিক্ত edge বসালে আমাদের খরচ বাড়বে, কোনো লাভ হবেনা। মিনিমাম স্প্যানিং ট্রি বের করার সময় আমরা এমন ভাবে এজগুলো নিবো যেন তাদের weight এর যোগফল মিনিমাইজ হয়।

এখন নিচের গ্রাফ থেকে কিভাবে আমরা mst বের করব?

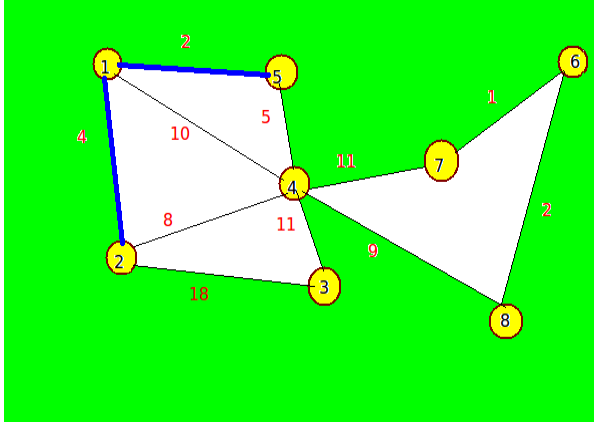


greedy অ্যাপ্রোচে খুব সহজে mst বের করা যায়। mst নির্ণয়ের প্রচলিত দুটি অ্যালগোরিদমই greedy। আমরা এখন prim's অ্যালগোরিদম কিভাবে কাজ করে দেখব।

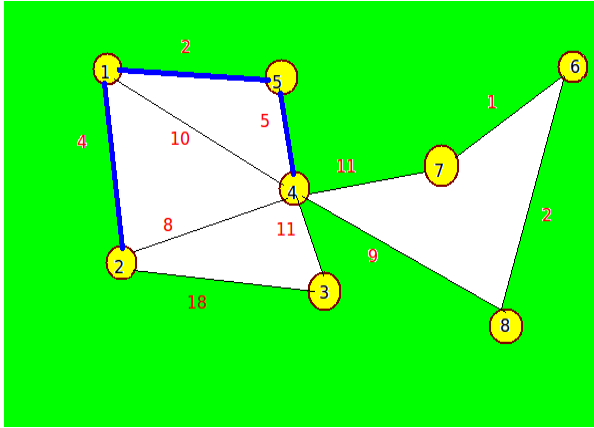
আমরা প্রথমে যেকোনো একটি সোর্স নোড নিব। ধরি সোর্স হলো ১। ১ থেকে যতগুলো এজ আছে সেগুলোর মিনিমাম টিকে আমরা সাবগ্রাফে যোগ করব। নিচের ছবিতে নীল এজ দিয়ে বুঝানো হচ্ছে এজটি সাবগ্রাফে যুক্ত করা হয়েছে:



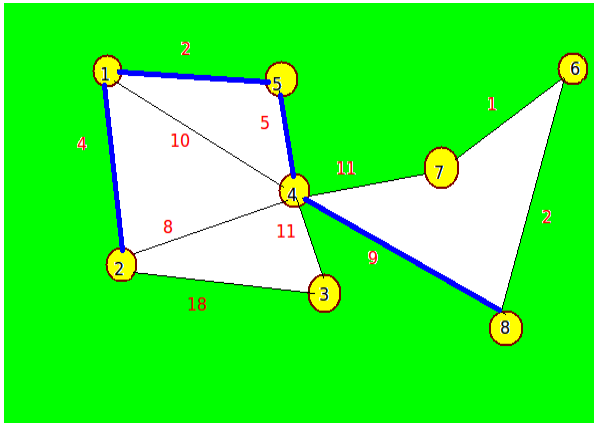
এবার সোর্স ১ এবং ৫ নম্বর নোড থেকে মোট যত এজ আছে(আগের এজগুলো সহ) তাদের মধ্যে মিনিমাম টি নিব:



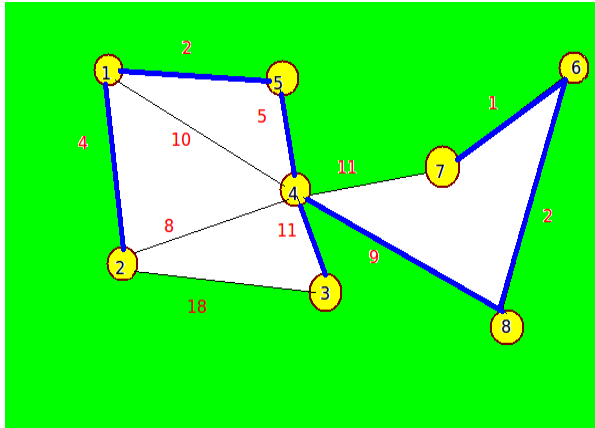
এবার নিব ১,২ এবং ৫ নম্বর নোড থেকে মোট যত এজ আছে(আগের এজগুলো সহ) তাদের মধ্যে মিনিমাম:



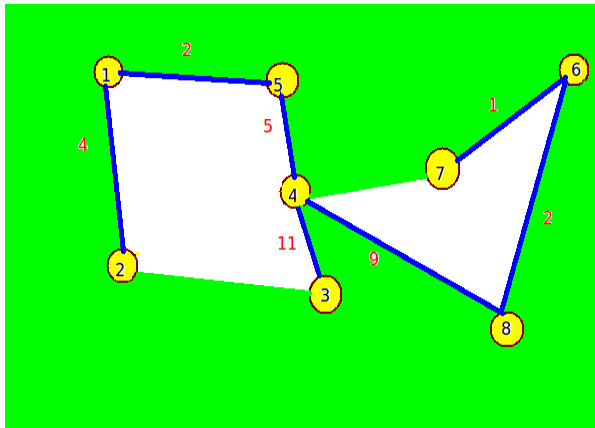
পরের ধাপটি গুরুত্বপূর্ণ। ১,২,৫,৪ থেকে যত এজ আছে তাদের মধ্য মিনিমাম হলো ২-৪, কিন্তু ২ নম্বর নোড এবং ৪ নম্বর নোড দুইটাই অলরেডি সাবগ্রাফের অংশ,তারা আগে থেকেই কানেক্টেড,এদের যোগ করলে সাবগ্রাফে সাইকেল তৈরি হবে,তাই ২-৪ এজটি নিয়ে আমাদের কোনো লাভ হবেনা। আমরা এমন প্রতিবার এজ নিব যেন নতুন আরেকটি নোড সাবগ্রাফে যুক্ত হয়। তাহলে ৪-৮ হবে আমাদের পরের চয়েস।



এভাবে শেষ পর্যন্ত আমরা পাবো:



নীলরং এর এই সাবগ্রাফটাই আমাদের মিনিমাম স্প্যানিং ট্রি। বাকি এজগুলো মুছে দিলে থাকে:



তাহলে টেলিফোন লাইন বসাতো মোট খরচ:  $8+2+5+11+9+2+1=38$ । একটি গ্রাফে অনেকগুলো মিনিমাম স্প্যানিং ট্রি থাকতে পারে তাই অনলাইন জাজে mst সংক্রান্ত প্রবলেম অনেক ক্ষেত্রেই special judge দিয়ে কাজ করে, অর্থাৎ সঠিক আউটপুটগুলোর যেকোনো একটা প্রিন্ট করলেই চলে।

আমাদের সুডোকোড হবে এরকম:

```
* Input: A non-empty connected weighted graph with vertices V and edges E (the weights can be negative).
* Initialize: Vnew = {x}, where x is an arbitrary node (starting point) from V, Enew = {}
* Repeat until Vnew = V:
    o Choose an edge (u, v) with minimal weight such that u is in Vnew and v is not
      (if there are multiple edges with the same weight, any of them may be picked)
    o Add v to Vnew, and (u, v) to Enew
* Output: Vnew and Enew describe a minimal spanning tree
(Wiki)
```

এখন মাথায় প্রশ্ন আসতে পারে কি ভাবে prim's mst ইম্প্লিমেন্ট করব? বারবার লুপ চালিয়ে naive অ্যাপ্রোচে কোড লিখলে তোমার কোড টাইম লিমিটের মধ্যে রান না করার সম্ভাবনাই বেশি।

আমাদের সমস্যার সমাধান হলো “priority queue”। আসা করি এই ডাটা স্ট্রাকচারটি সম্পর্কে সবাই জানে।

প্রতিটি নোড ডিজিট করার সাথে সাথে নতুন এজগুলোকে priority queue তে ঢুকিয়ে রাখবে। priority queue তে এজগুলোকে weight অনুযায়ী সোর্ট রাখবে। তাহলে প্রতিবার queue এর top এ পাবে এখন পর্যন্ত নেয়া নোডগুলো থেকে বের হওয়া সবথেকে কম weight এর এজ, সেটাকে সাবগ্রাফে যোগ করবে।

নিজে priority queue না বানিয়ে stl বা java collections ব্যবহার কর। সি++ এ অপারেটর ওভারলোডিং করে priority queue সর্ট করা সব থেকে সহজ। আমরা প্রথমে ডাটা নামক একটি structure ডিক্লেয়ার করবো:

```
struct data
{
    int u,v,cost;
    bool operator < ( const data& p ) const { //overloading operator
        return cost > p.cost;
    }
};
```

এরপর priority queue ডিক্লেয়ার করে সেটায় 'data' টাইপের ভ্যারিয়েবল পুশ করলে Q তে অটোমেটিক সর্ট হয়ে যাবে। যেহেতু এটা অ্যালগোরিদমের টিউটোরিয়াল, সি++ এর না, তাই এসব নিয়ে আর বেশি লিখবোনা, কোথায় সমস্যা হলে মন্তব্য অংশে বা আমার সাথে যোগাযোগ করে জানাতে পারো।

mst নির্ণয়ের জন্য আরেকটি অ্যালগোরিদম আছে যা kruskal's mst নামে পরিচিত যা কন্ট্রোলিংয়ের মধ্যে বেশি জনপ্রিয়। kruskal শিখতে হলে তোমাকে disjoint set(union find) নামের একটি চমতকার data structure এর সাথে অকশ্যই পরিচিত হতে হবে। তারপর kruskal's mst শেখা খুব সহজ হবে, prim's শিখলেও kruskal অকশ্যই শিখতে হবে। disjoint set শেখার জন্য নিচের টিউটোরিয়ালগুলো

দেখো:

আমার লেখা টিউটোরিয়াল

topcoder tutorial

Wikipedia

Coreman's introduction to algorithm

mst নিয়ে uva টে অনেক প্রবলেম আছে। অ্যালগোরিদমটা implement করার পর অবশ্যই নিচের সমস্যাগুলো সমাধানের চেষ্টা করবে, আবারো বলবো প্রবলেম সল্ড না করলে অ্যালগোরিদম শেখা অর্থহীন কারণ মাথা খাটিয়ে প্রবলেম সলভিং হলো স্কিল ডেভেলপ করার সব থেকে ভালো উপায়।

<http://uva.onlinejudge.org/external/5/544.html>(Straight forward)

<http://uva.onlinejudge.org/external/9/908.html>

<http://uva.onlinejudge.org/external/100/10034.html>(Straight forward)

<http://uva.onlinejudge.org/external/112/11228.html>

<http://uva.onlinejudge.org/external/104/10462.html>(2nd best mst)

spoj:


<http://www.spoj.pl/problems/MST/>(Straight forward)

গ্রাফ থিওরিতে হাতেখড়ি সিরিজটা কেমন লেগেছে?

- ☐ খুবই ভালো, আমি এই সিরিজ পড়েই গ্রাফ থিওরি প্রথম শিখেছি/শিখছি।
- ☐ আমি গ্রাফ থিওরি সম্পর্কে জানি, তাও এই সিরিজটি পড়ে উপকার হয়েছে।
- ☐ লেখাগুলো মোটামুটি চলনসই, তবে আরো বিস্তারিত লেখা উচিত ছিলো।
- ☐ পুরো অকাজের সিরিজ, কিছুই শিখতে পারিনি এটা থেকে।

Vote

[View Results](#)

 Loading ...

ফেসবুকে মন্তব্য

[Facebook Comments](#) ওয়ার্ডপ্রেস প্লাগইন।