

শাফায়েতের ব্লগ

প্রোগ্রামিং, অ্যালগরিদম, ব্যাকএন্ড ইঞ্জিনিয়ারিং

Home

অ্যালগরিদম নিয়ে যত লেখা!

আমার সম্পর্কে...

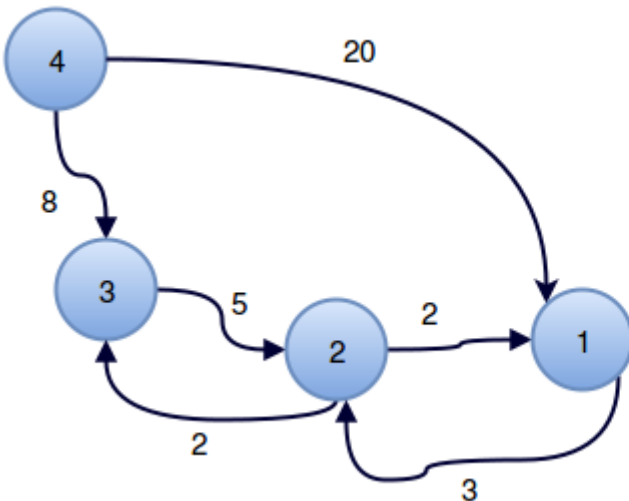
গ্রাফ থিওরিতে হাতেখড়ি ১০: ফ্লয়েড ওয়ার্শাল

📅 জুলাই ২২, ২০১৪ by *Shafaet Ashraf*

[f](#)
[Twitter](#)
[g+](#)
[in](#)

ফ্লয়েড ওয়ার্শাল সম্ভবত সব থেকে ছোট আকারের গ্রাফ অ্যালগোরিদম, মাত্র ৩লাইনে এটা লেখা যায়! তবে ৩ লাইনের এই অ্যালগোরিদমেই বোঝার অনেক কিছু আছে। ফ্লয়েড ওয়ার্শালের কাজ হলো গ্রাফের প্রতিটা নোড থেকে অন্য সবগুলো নোডের সংক্ষিপ্ততম দূরত্ব বের করা। এ ধরনের অ্যালগোরিদমকে বলা হয় “অল-পেয়ার শর্টেস্ট প্যাথ” অ্যালগোরিদম। এই লেখাটা পড়ার আগে [অ্যাডভেন্সেস ম্যাট্রিক্স](#) সম্পর্কে জানতে হবে।

আমরা একটা গ্রাফের উপর কিছু সিমুলেশন করে সহজেই অ্যালগোরিদমটা বুঝতে পারি। নিচের ছবিটা দেখ:

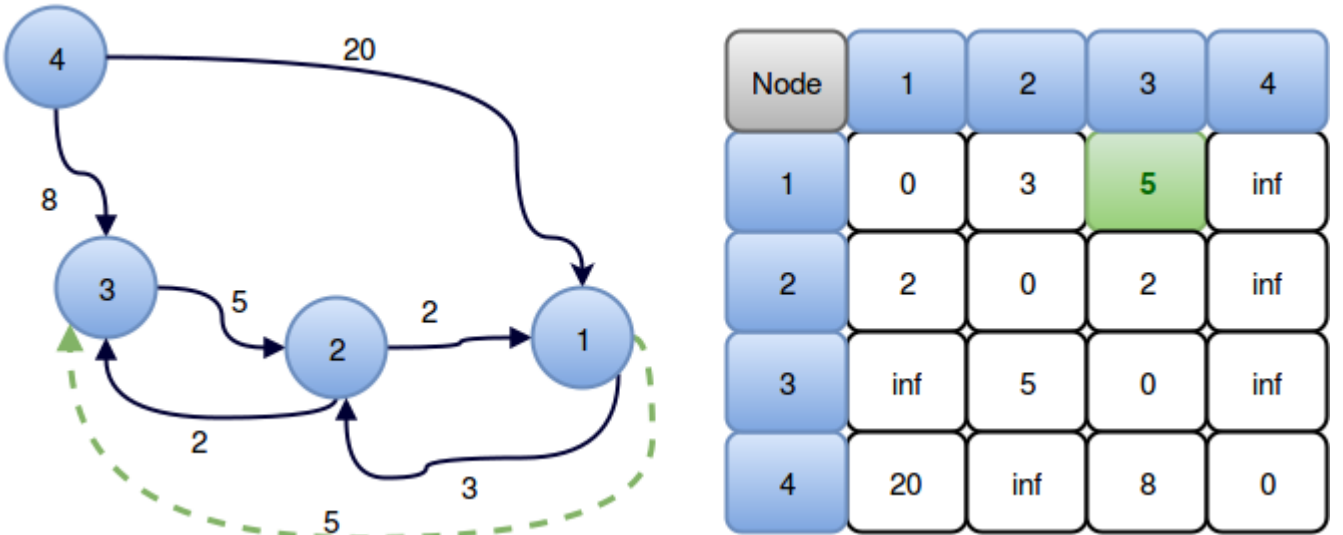


Node	1	2	3	4
1	0	3	inf	inf
2	2	0	2	inf
3	inf	5	0	inf
4	20	inf	8	0

ছবিতে চার নোডের একটা ওয়েটেড ডিরেক্টেড গ্রাফ দেখা যাচ্ছে। আর উপরে ডান কোনায় একটা ম্যাট্রিক্স। ম্যাট্রিক্সের u, v তম ঘরে বসানো হয়েছে $u-v$ এজ এর ওয়েট বা কস্ট। যাদের মধ্যে সরাসরি এজ নেই সেসব ঘরে অসীম বা ইনফিনিটি বসিয়ে দেয়া হয়েছে। আর কোনাকুনি ঘরগুলোতে মান ০ কারণ নিজের বাসা থেকে নিজের বাসাতেই যেতে কোন দূরত্ব অতিক্রম করতে হয় না!

এখন মনে করো “২” নম্বর নোডটাকে আমরা “মাঝের নোড” হিসাবে ধরলাম। মাঝের নোডকে আমরা বলবো k । তারমানে এখন $k=2$ । (আমরা একে একে সব নোডকেই মাঝের নোড হিসাবে ধরবো, **এটা যেকোন অর্ডারে করা যায়**)

এখন যেকোন এক জোড়া নোড (i, j) নাও। ধরি $i=1, v=3$ । আমরা চাই **u থেকে v তে যেতে, k নোডটাকে মাঝে রেখে।** তাহলে আমাদের i থেকে k তে যেতে হবে, তারপর k থেকে থেকে j তে যেতে হবে। কিন্তু লাভ না হলে আমরা এভাবে যাব কেন? আমরা k কে মাঝে রেখে যাবো কেবল যদি মোট কস্ট(cost) কমে যায়। ১ থেকে ৩ এর বর্তমান দূরত্ব $matrix[1][3]=$ ইনফিনিটি। আর যদি $k=২$ কে মাঝখানে রাখি তাহলে দূরত্ব দাড়াবে $matrix[1][2] + matrix[2][3] = 3 + 2 = 5$ । তারমানে কস্ট কমে যাচ্ছে! আমরা গ্রাফটা আপডেট করে দিতে পারি এভাবে:



আমরা ২ কে “মাঝের নোড” হিসাবে ব্যবহার করে ১ থেকে ৩ এ গিয়েছি মোট ৫ কস্ট এ। গ্রাফে তাহলে ১ থেকে ৩ এ সরাসরি একটা এজ দিয়ে দিতে পারি ৫ কস্ট এ।

এখন আবার ধর $i=2, j=4$ । আর আগের মতই $k=2$ । এবার $matrix[2][4]=$ ইনফিনিটি। এদিকে $matrix[2][2] + matrix[2][4] = ০ +$ ইনফিনিটি। এবার কিন্তু দূরত্ব কমলো না। তাই গ্রাফ আপডেট করার দরকার নেই। নিশ্চয়ই বুঝতে পারছো আপডেটের শর্তটা হবে এরকম:

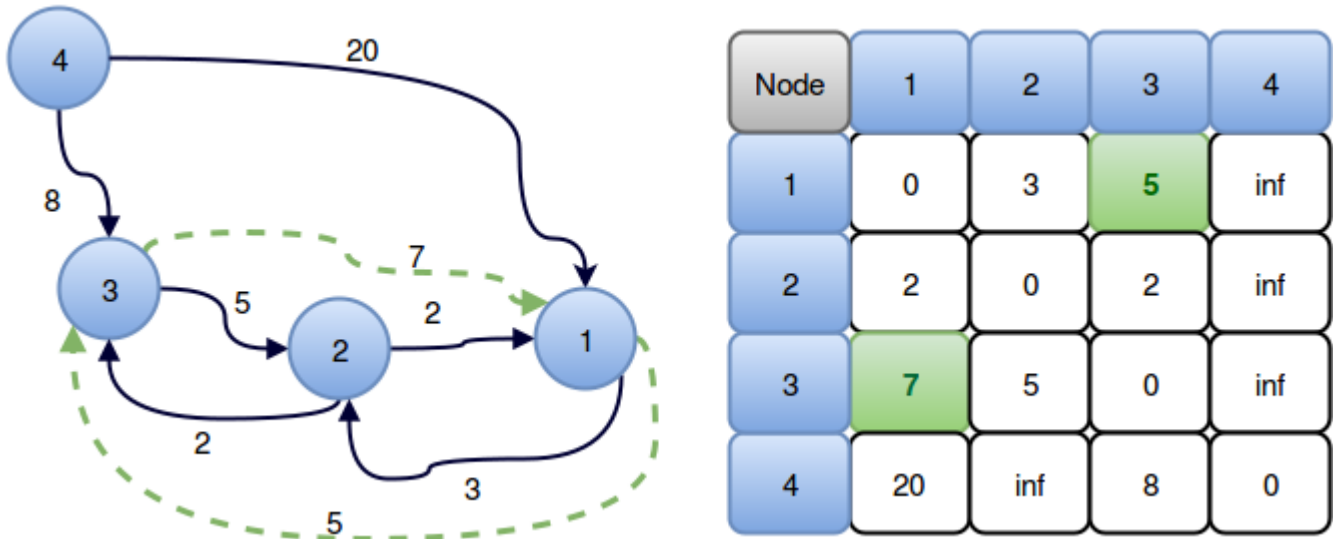
“ $if(matrix[i][k] + matrix[k][j] < matrix[i][j])$

$matrix[i][j] = matrix[i][k] + matrix[k][j]$

অথবা আমরা একলাইনে লিখতে পারি:

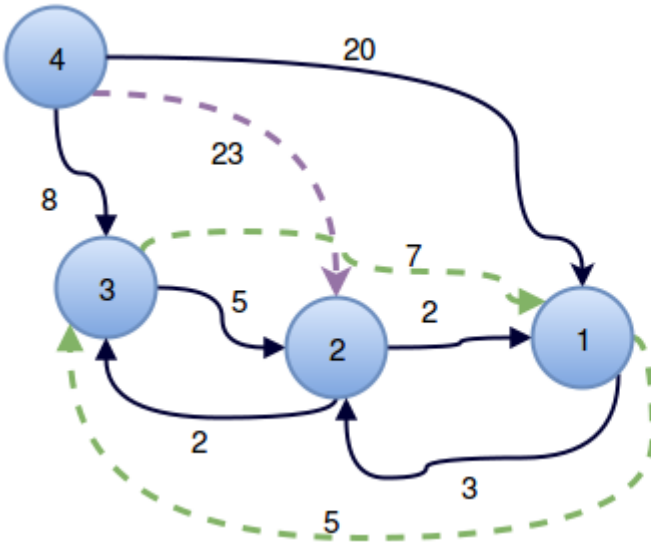
“ $matrix[i][j] = \min(matrix[i][j], matrix[i][k] + matrix[k][j])$

এখন $k=2$ স্থির রেখে আমরা i, j এর সবরকমের কম্বিনেশন নিবো। তুমি নিজেই চিন্তা করলে দেখবে $k=2$ এর জন্য $i=3, j=1$ এই কম্বিনেশনে আমরা আরেকটা নতুন এজ পাবো, বাকি গ্রাফ আগের মতই থাকবে।



এবার আমরা $k=1$ কে মাঝের নোড হিসাবে চিন্তা করি। মাথা খাটাতে চাইলে নিচে দেখার আগে নিজেই খাতায় একে ফেলতে পারো নতুন এজগুলো।

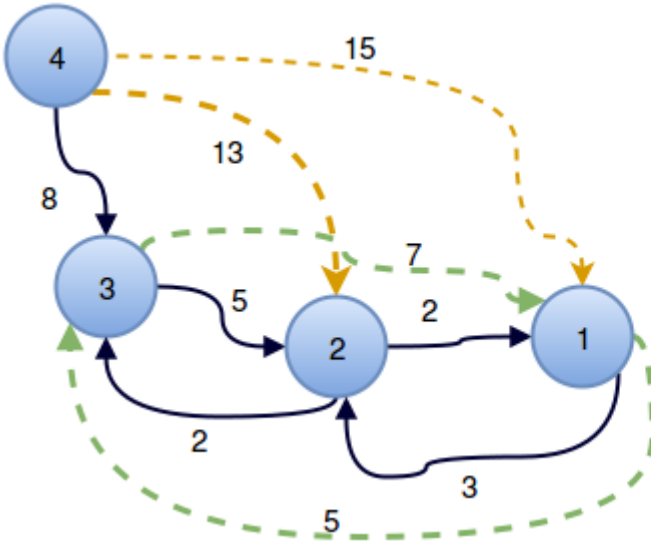
এবার একটা মাত্র এজ যোগ হবে। $i=4, j=2$ হলে আমরা ৪ থেকে ১ হয়ে ২ নম্বর নোডে যেতে পারি ২৩ কস্ট এ। তাহলে গ্রাফটা হবে এরকম:



Node	1	2	3	4
1	0	3	5	inf
2	2	0	2	inf
3	7	5	0	inf
4	20	23	8	0

এবার ৩ নম্বর নোডকে মাঝের নোড হিসাবে ধরবো। আবারো নিজে চেষ্টা করে তারপর নিচের অংশ দেখ।

এবার কিছু মজার জিনিস ঘটবে। ৪ থেকে ১ এ আগে লাগছিল ২০ কস্ট। এখন ৩ কে মাঝে রেখে ৪ থেকে ১ এ গেলে লাগবে $৮+৭=১৫$ কস্ট। লক্ষ্য কর একদম শুরুতে ৩ থেকে ১ এ আমাদের এজ ছিল না। কিন্তু আপডেট করার সময় আমরা এজ বসিয়ে দিয়েছি, এখন ৩ থেকে ১ এ যদিও সরাসরি চলে যাচ্ছি, মূল গ্রাফে আসলে $৩ \rightarrow ২ \rightarrow ১$ পথে যাচ্ছি। একই ভাবে ৪ থেকে ২ এর ২৩ কস্ট এর পথটা আপডেট হয়ে ১৩ হয়ে যাবে।



Node	1	2	3	4
1	0	3	5	inf
2	2	0	2	inf
3	7	5	0	inf
4	15	13	8	0

$k=৪$ এর জন্য আর কোন আপডেট হবে না কারণ কোনো নোড থেকে ৪ এ যাওয়া যায় না।

এখন আমরা ম্যাট্রিক্স দেখেই বলে দিতে পারছি কোন নোড থেকে কোন নোডে কত কস্ট এ যাওয়া যায়। ইনফিনিটি থাকা মানে সেই নোডে যাবার পথ নেই।

ইনপুট থেকে অ্যাডজেসেন্সি ম্যাট্রিক্স বানাবার পর তাহলে কাজ খুবই সহজ:

```
1 for k from 1 to |V|
2   for i from 1 to |V| \ \ |V| = number of nodes
3     for j from 1 to |V|
4       if matrix[i][j] > matrix[i][k] + matrix[k][j]
5         matrix[i][j] ← matrix[i][k] + matrix[k][j]
```

কোডে আমরা k এর লুপটা ১ থেকে চালাচ্ছি যদিও উদাহরণে আগে ২ নিয়েছি। এটা আসলে যেকোন অর্ডারেই করা যায়, তুমি আগে ১ নিলেও দেখবে অ্যালগোরিদম কাজ করবে।

এভাবেতো আমরা শুধু পাথের কস্ট পেলাম, পাথটা িকভাবে পাব?

ধরো আমাদের একটা ম্যাট্রিক্স আছে next[][]। এখন next[i][j] দিয়ে আমরা বুঝি i থেকে j তে যেতে হলে পরবর্তি যে নোড এ যেতে হবে সেই নোডটা। তাহলে একদম শুরুতে সব i,j এর জন্য next[i][j] = j হবে। কারণ শুরুতে কোন “মাঝের নোড” নেই এবং তখনও শর্টেস্ট পাথ বের করা শেষ হয় নি।

এখন আমরা যখন matrix[i][j] আপডেট করবো লুপের সেটার মানে হলো মাঝে একটা নোড k ব্যবহার করে আমরা যাবো। লক্ষ্য কর আমরা কিন্তু মূল গ্রাফে সরাসরি এজ দিয়ে i থেকে k তে নাও যেতে পারি, আমরা শুধু জানি i,j নোড দুটোর মাঝে একটা নোড k আছে যেখানে আমাদের যেতে হবে j তে যাবার আগে। i থেকে k তে যাবার পথে পরবর্তি যে নোডে যেতে হবে সেটা রাখা আছে next[i][k] তে! তাহলে next[i][j] = next[i][k] হয়ে যাবে।

```
1 for k from 1 to |V|
2   for i from 1 to |V|
3     for j from 1 to |V|
4       if matrix[i][k] + matrix[k][j] < matrix[i][j] then
5         matrix[i][j] ← matrix[i][k] + matrix[k][j]
6         next[i][j] ← next[i][k]
7
8
9 findPath(i, j)
10  path = [i]
11  while i ≠ j
12    i ← next[i][j]
13    path.append(i)
14  return path
```

findpath ফাংশনে আমরা j কে ফিক্সড রেখে next অ্যারে ধরে আগাচ্ছি যতক্ষণ না j তে পৌঁছাচ্ছি। তাহলে আমরা পেয়ে গেলাম পাথ!

ট্রান্সিটিভ ক্লোজার(Transitive Closure):

ধরো আমাদের অ্যাডজেন্সি ম্যাট্রিক্সটা এরকম:

```
matrix[i][j] = 1 যদি i থেকে j তে সরাসরি এজ থাকে
matrix[i][j] = 0 যদি এজ না থাকে
```

এখন আমরা এমন একটা ম্যাট্রিক্স তৈরি করতে চাই যেটা দেখে বলে দেয়া যাবে i থেকে j তে এক বা একাধিক এজ ব্যবহার করে যাওয়া যায় কিনা। আমরা চাইলে উপরের মত করে 0 এর জায়গায় ইনফিনিটি দিয়ে শর্টেস্ট পাথ বের করে কাজটা করতে পারতাম। কিন্তু এক্ষেত্রে “OR” আর “AND” অপারেশন ব্যবহার আরো দ্রুত কাজটা করা যায়। এখন আপডেটের শর্তটা হয়ে যাবে এরকম:

```
matrix[i][j] = matrix[i][j] || (matrix[i][k] && matrix[k][j])
```

এটার মানে matrix[i][j] তে তখনই ১ বসবে যখন হয় “matrix[i][j] তে ১ আছে” অথবা “matrix[i][k] এবং matrix[k][j]” দুটোতেই ১ আছে। তারমানে হয় সরাসরি যেতে হবে অথবা মাঝে একটা নোড k ব্যবহার করে যেতে হবে।

কমপ্লেক্সিটি:

৩টা নেস্টেড লুপ ঘুরছে নোড সংখ্যার উপর, টাইম কমপ্লেক্সিটি $O(n^3)$ । ২ডি ম্যাট্রিক্স ব্যবহার করায় স্পেস কমপ্লেক্সিটি $O(n^2)$ ।

কিছু প্রশ্ন:

১. k এর লুপটা কি i, j লুপের ভিতর দিলে অ্যালগোরিদম কাজ করত?
২. গ্রাফে নেগেটিভ কস্ট থাকলে ফ্লয়েড ওয়ার্শাল কাজ করবে কি?

রিলেটেড প্রবলেম:

Page Hopping

05-2 Rendezvous

Minimum Transport Cost

Asterix and Obelix

আরো অনেক প্রবলেম

হ্যাপি কোডিং!

ফেসবুকে মন্তব্য

0 comments