সূচিপত্র

অধ্যায় ১

ভূমিকা

You can being your chapters with this quote box :D

– Me

গণিত সম্পর্কিত কোনো বিষয়ের কিছু লিখতে গেলে ল্যাটেকের কোনো বিকল্প নেই। তবে ল্যাটেক দিয়ে বাংলায় সরাসরি কিছু লিখতে গেলে তেমন ভালো সাপোর্ট পাওয়া যায় না। সেই সমস্যাকে ট্যাকেল করতে গণিত অলিম্পিয়াডের আদীব হাসানের বানানো ল্যাটেকবাংলা প্যাকেজটি অত্যন্ত গুরুত্বপূর্ণ। পরবর্তীতে যাওয়াদ আহমেদ চৌধুরী ও এম আহসান আল মাহীর সেই প্যাকেজটিকে তাদের বইয়ে ব্যবহারের জন্য আরো কিছু ফিচার যুক্ত করেছেন।

এই টেমপ্লেট এ প্রায় সব environment ডিফাইন করা আছে। সেগুলোর টাইটেল বাংলায় আসবে। যেমন

সমস্যা ১.১: এটি একটি সমস্যা

এছাড়াও আর কিছু environment বানানো আছে, সেগুলো environments.sty ফাইলে পাওয়া যাবে।

অধ্যায় ২

ম্যাট্রিক্স এক্সপোনেন্সিয়েশন

§ ২.১ শুরুর কথা

নামটা শুনতে কঠিন মনে হলেও ম্যাট্রিক্স এক্সপোনেন্সিয়েশন আসলে তেমন কঠিন কিছু না। ম্যাট্রিক্স সম্পর্কে কমবেশি সবারই জানা থাকার কথা। তারপরেও যারা এ সম্পর্কে জানো না তারা ম্যাট্রিক্সকে 2D অ্যারের মত চিন্তা করতে পার। বাইরে থেকে দুটি একইরকমই দেখতে। যদি কোন ম্যাট্রিক্সের n টি সারি আর m টি কলাম থাকে তাহলে ম্যাট্রিক্সটিকে $n \times m$ ম্যাট্রিক্স বলা হয়। যেমন নিচের ম্যাট্রিক্সটি একটি 2×3 ম্যাট্রিক্স।

$$\begin{pmatrix} 1 & 3 & 2 \\ 9 & 0 & 7 \end{pmatrix}$$

ঠিক অ্যারের মতই কোন ম্যাট্রিক্স A এর i তম সারির j তম সংখ্যাকে $A_{i,j}$ দিয়ে প্রকাশ করা হয়। যেমন উপরের ম্যাট্রিক্সের জন্য $A_{1,1} = 1$, আবার $A_{2,3} = 7$। ম্যাট্রিক্সের যোগ, বিয়োগও সম্ভব, তবে তুমি একটি $n \times m$ ম্যাট্রিক্সের সাথে আরেকটি $n \times m$ ম্যাট্রিক্সই যোগ বা বিয়োগ করতে পারবে। এক্ষেত্রে A এবং B যোগ করে C পাওয়া গেলে $C_{i,j} = A_{i,j} + B_{i,j}$ হতে হবে। যেমন

$$\begin{pmatrix} 1 & 3 \\ 9 & 0 \end{pmatrix} + \begin{pmatrix} 2 & -1 \\ 3 & 1 \end{pmatrix} = \begin{pmatrix} 1+2 & 3-1 \\ 9+3 & 0+1 \end{pmatrix}$$

তবে সবচেয়ে অদ্ভুত হচ্ছে ম্যাট্রিক্সের গুন। গুনের ক্ষেত্রে একটি $n \times m$ ম্যাট্রিক্সের সাথে কেবল একটা $m \times k$ ম্যাট্রিক্স গুন করতে পারবে এবং গুণফল

হবে একটা $n \times k$ ম্যাট্রিক্স। অর্থাৎ প্রথম ম্যাট্রিক্সের কলাম সংখ্যা আর দ্বিতীয় ম্যাট্রিক্সের সারি সংখ্যা সমান হতে হবে। C যদি A এবং B ম্যাট্রিক্সের গুণফল হয় তাহলে

$$C_{i,j} = \sum_{x=1}^{m} A_{i,x} \times B_{x,j}$$

যেমন ধর,

$$\begin{pmatrix} 1 & 3 & 2 \\ 9 & 0 & 7 \end{pmatrix} \begin{pmatrix} 5 & 6 & 0 & 3 \\ 0 & 2 & -1 & 1 \\ 1 & 1 & 4 & -1 \end{pmatrix} = \begin{pmatrix} 5 & 6 & 7 & 8 \\ 9 & 10 & 12 & 13 \end{pmatrix}$$

এখানে 2×3 ম্যাট্রিক্সের সাথে 3×4 ম্যাট্রিক্স গুন করে 2×4 ম্যাট্রিক্স পাওয়া গিয়েছে। তবে গুণফলটা আসলে কীভাবে বের হল সেটা বুঝতে একটু ছোট উদাহরণ দেখা যাক। নিচের ২টি 2×2 ম্যাট্রিক্সের গুণ করা যাক

$$\begin{pmatrix} a & b \\ c & d \end{pmatrix} \begin{pmatrix} p & q \\ r & s \end{pmatrix} = \begin{pmatrix} ap + br & aq + bs \\ cp + dr & cq + ds \end{pmatrix}$$

$C_{2,1}$ এর কথা ধর। প্রথম ম্যাট্রিক্সের ২য় সারির সংখ্যাগুলো হচ্ছে c এবং d, আবার দ্বিতীয় ম্যাট্রিক্সের ১ম কলামের সংখ্যাগুলো হচ্ছে p এবং r। তাই c এর সাথে p গুন করেছি আর d এর সাথে q গুন করেছি, এরপর গুণফল দুটিকে যোগ করে দিয়েছি। এজন্যই $C_{2,1}$ এর মান $cp + dr$। অন্য পদগুলোও এভাবেই বের করা যাবে। (তোমরা হয়ত ভাবছ এমন অদ্ভুত ভাবে ম্যাট্রিক্স গুন করা হয় কেন। এর উত্তর জানতে লিনিয়ার আলজেব্রা পড়তে হবে। চাইলে 3blue1brown এর ভিডিও সিরিজটি দেখতে পারো)।

ম্যাট্রিক্স গুণফলের সবচেয়ে চমদপ্রদক দিক হল অ্যাসোসিয়েটিভিটি। যেমন ধর তুমি তিনটি ম্যাট্রিক্স A, B, C গুন করতে চাও, অর্থাৎ ABC এর মান বের করতে চাও। তাহলে তুমি AB এর সাথে C কে গুন করলে যে ম্যাট্রিক্স পাওয়া যাবে, A এর সাথে BC কে গুন করলে একই ম্যাট্রিক্স পাওয়া যাবে। সহজ ভাষায় $A(BC) = (AB)C$। সোজা কথায় আমরা যেভাবেই ব্রাকেট বসাই

না কেন একই উত্তর আসবে। এই বৈশিষ্ট্য আমাদের পরে কাজে লাগবে। তবে সাবধান! AB কিন্তু কখনই BA এর সমান নয়। কোনটিকে আগে কোনটিকে পরে গুন করতে হবে তা লক্ষ্য রাখতে হবে।

§ ২.২ ডাইনামিক প্রোগ্রামিং এর সাথে সম্পর্ক

আবার ফিবোনাচ্চি সমস্যায় ফেরত যাওয়া যাক। রিকারেন্সটি নিশ্চয় মনে আছে,

$$\begin{aligned} f_0 &= 0 \\ f_1 &= 1 \\ f_n &= f_{n-1} + f_{n-2} \end{aligned}$$

আমরা এমন একটি 2×2 ম্যাট্রিক্স A বের করতে চাই যেন,

$$\begin{pmatrix} a & b \\ c & d \end{pmatrix} \begin{pmatrix} f_n \\ f_{n-1} \end{pmatrix} = \begin{pmatrix} f_{n+1} \\ f_n \end{pmatrix}$$

অর্থাৎ f_n ও f_{n-1} এর ভেক্টরের ($n \times 1$ ম্যাট্রিক্স গুলোকে ভেক্টর বলা হয়) সাথে এমন একটি ম্যাট্রিক্স গুন করতে যেন f_{n+1} ও f_n এর ভেক্টর পাওয়া যায়। কাজটা কিন্তু খুব কঠিন না। একটু চেষ্টা করলেই বুঝবে $A = \begin{pmatrix} 1 & 1 \\ 1 & 0 \end{pmatrix}$ ম্যাট্রিক্সটি কাজ করে

$$\begin{pmatrix} 1 & 1 \\ 1 & 0 \end{pmatrix} \begin{pmatrix} f_n \\ f_{n-1} \end{pmatrix} = \begin{pmatrix} 1f_n + 1f_{n-1} \\ 1f_n + 0f_{n-1} \end{pmatrix} = \begin{pmatrix} f_{n+1} \\ f_n \end{pmatrix}$$

এখন লক্ষ্য কর, A ম্যাট্রিক্সটি যদি দুইবার গুন করি তাহলে কিন্তু $\begin{pmatrix} f_n \\ f_{n-1} \end{pmatrix}$ থেকেই $\begin{pmatrix} f_{n+2} \\ f_{n+1} \end{pmatrix}$ পেয়ে যাবো। কারণ

$$A \times A \times \begin{pmatrix} f_n \\ f_{n-1} \end{pmatrix} = A \times \begin{pmatrix} f_{n+1} \\ f_n \end{pmatrix} = \begin{pmatrix} f_{n+2} \\ f_{n+1} \end{pmatrix}$$

লক্ষ্য কর এখানে আমরা ম্যাট্রিক্সের অ্যাসোসিয়েটিভিটি ধর্মটি ব্যবহার করেছি। আবার যদি আমরা দুইবারের বদলে m বার A ম্যাট্রিক্সটি গুন করতাম, তাহলে একইভাবে আমরা পাব

$$A^m \begin{pmatrix} f_n \\ f_{n-1} \end{pmatrix} = A^{m-1} \begin{pmatrix} f_{n+1} \\ f_n \end{pmatrix} = \cdots = \begin{pmatrix} f_{n+m} \\ f_{n+m-1} \end{pmatrix}$$

উপরের সমীকরণে $n = 1$ বসালে আমরা পাব

$$\begin{pmatrix} 1 & 1 \\ 1 & 0 \end{pmatrix}^m \begin{pmatrix} f_1 \\ f_0 \end{pmatrix} = \begin{pmatrix} f_{m+1} \\ f_m \end{pmatrix}$$

তোমরা হয়ত ভাবছ, এত কিছু বের করে আসলে কী লাভ হল। আমরা শুরুতে যখন n তম ফিবোনাচ্চি নাম্বার বের করা শিখেছিলাম সেটার কমপ্লেক্সিটি ছিল $\mathcal{O}(n)$। কিন্তু ম্যাট্রিক্স এক্সপনেন্সিয়েশন দিয়ে আমরা কাজটা $\mathcal{O}(\log n)$ এই করে ফেলতে পারি। কারণ দেখ, n তম ফিবনাচ্চি নাম্বার বের করতে আমাদের A^n কে ফাস্ট ক্যালকুলেট করতে হবে। এজন্য কিন্তু আমরা সংখ্যার ক্ষেত্রে a^b যেভাবে বাইনারি এক্সপনেন্সিয়েশন দিয়ে বের করি সেভাবেই কাজটা করে ফেলতে পারি। অর্থাৎ n জোড় হলে প্রথমে $A^{\frac{n}{2}}$ বের করে তাকে বর্গ করে দিলেই হচ্ছে। আবার n বিজোড় হলে প্রথমে A^{n-1} বের করে তার সাথে A গুন করে দিলেই হচ্ছে। এভাবে আমাদের $\mathcal{O}(\log n)$ বার দুটি 2×2 ম্যাট্রিক্স গুন করতে হচ্ছে। দুটি 2×2 ম্যাট্রিক্স গুন করার কমপ্লেক্সিটি আমরা $\mathcal{O}(1)$ ই ধরতে পারি। তাই সবমিলিয়ে কমপ্লেক্সিটি হবে $\mathcal{O}(\log n)$।

তবে একটা জিনিশ বলে রাখা দরকার। এখানে ম্যাট্রিক্স এর আকার অনেক ছোট বলে আমরা দুটি ম্যাট্রিক্স গুন করার কমপ্লেক্সিটি $\mathcal{O}(1)$ ধরেছি। কিন্তু অনেক ক্ষেত্রে বেশ বড় ম্যাট্রিক্স লাগতে পারে (যেমন ধর 50×50 ম্যাট্রিক্স)। সেক্ষেত্রে কিন্তু ম্যাট্রিক্স গুন করার কমপ্লেক্সিটি $\mathcal{O}(1)$ ধরলে হবে না। খেয়াল করলে দেখবে দুটি $k \times k$ ম্যাট্রিক্স গুন করতে আমাদের $\mathcal{O}(k^3)$ কমপ্লেক্সিটি প্রয়োজন। সেক্ষেত্রে আমাদের ম্যাট্রিক্স এক্সপনেন্সিয়েশনের কমপ্লেক্সিটি হবে $\mathcal{O}(k^3 \log n)$

§ ২.৩ আরো কিছু উদাহরণ

আরেকটা উদাহরণ দেখা যাক। ধর এবার আমাদের রিকারেন্সটি হল

$$\begin{aligned} f_0 &= 0 \\ f_1 &= 2 \\ f_2 &= 1 \\ f_n &= 2f_{n-1} + 3f_{n-2} - 7f_{n-3} \end{aligned}$$

যেহেতু f_n আগের তিনটি পদের ওপর নির্ভরশীল, তাই আমাদের এবার একটি 3×3 ম্যাট্রিক্স খুঁজতে হবে। ফিবোনাচ্চির ম্যাট্রিক্স তা যদি বুঝে থাক তাহলে এটা বের করাও তেমন কঠিন না। নিচের ম্যাট্রিক্সটা দেখ

$$\begin{pmatrix} 2 & 3 & -7 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} f_n \\ f_{n-1} \\ f_{n-2} \end{pmatrix} = \begin{pmatrix} 2f_n + 3f_{n-1} - 7f_{n-2} \\ 1f_n + 0f_{n-1} + 0f_{n-2} \\ 0f_n + 1f_{n-1} + 0f_{n-2} \end{pmatrix} = \begin{pmatrix} f_{n+1} \\ f_n \\ f_{n-1} \end{pmatrix}$$

এবার একটু জটিল উদাহরণ চেষ্টা করা যাক। ধর এবার আমাদের কাছে ২ টি রিকারেন্স আছে।

$$\begin{aligned} f_n &= 2f_{n-1} + g_{n-2} \\ g_n &= g_{n-1} + 3f_{n-2} \end{aligned}$$

ধরে নাও f_0, f_1, g_0, g_1 এর মান জানা আছে। এবার আমাদের ভেক্টরে কিন্তু শুধু f_n, f_{n-1} রাখলে চলবে না, বরং g_n, g_{n-1} এর মানও রাখতে হবে। যদি এটা ধরতে পারো তাহলে আগেরগুলোর মতই এটাও সমাধান করা যায়

$$\begin{pmatrix} 2 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 3 & 1 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} f_n \\ f_{n-1} \\ g_n \\ g_{n-1} \end{pmatrix} = \begin{pmatrix} 2f_n + g_{n-1} \\ f_n \\ 3f_{n-1} + g_n \\ g_n \end{pmatrix} = \begin{pmatrix} f_{n+1} \\ f_n \\ g_{n+1} \\ g_n \end{pmatrix}$$

সমস্যা ২.১: নিচের রিকারেন্সটির জন্য ম্যাট্রিক্স বের কর।

$$\begin{aligned} f_0 &= 0 \\ f_1 &= 1 \\ f_n &= f_{n-1} + f_{n-2} + n \end{aligned}$$

সমাধান: এটা প্রায় ফিবনাচ্চি সমস্যাটির মতোই, কিন্তু ঝামেলা হচ্ছে রিকারেন্সে একটি n যোগ করা হয়েছে। এটা না সরালে ধ্রুবক কোন ম্যাট্রিক্স পাওয়া যাবেনা। এজন্য আমরা আগের সমস্যার মত এমন আরেকটি রিকারেন্স g বের করতে পারি যেন $g_n = n$ হয়। এটা বের করা বেশ সহজ

$$\begin{aligned} g_0 &= 0 \\ g_n &= g_{n-1} + 1 \end{aligned}$$

এরপর n এর বদলে g_n বসিয়ে দিলেই আমরা ঠিক আগের উদাহরণের মত ম্যাট্রিক্সটি বের করতে পারব। রিকারেন্স দুটোকে এক করলে পাব

$$\begin{aligned} g_n &= g_{n-1} + 1 \\ f_n &= f_{n-1} + f_{n-2} + g_n \end{aligned}$$

■

সমস্যা ২.২: নিচের ধারাটির জন্য ম্যাট্রিক্স বের কর

$$\sum_{i=1}^{n} i^k = 1^k + 2^k + 3^k + \cdots + n^k$$

সমাধান: যদিও এটা ঠিক ডাইনামিক প্রোগ্রামিং এর সমস্যা না, এরপরেও ম্যাট্রিক্স এক্সপো এর খুব সুন্দর একটা উদাহরণ। যোগফলের জন্য খুব সহজ একটা রিকারেন্স বের করতে পারি

$$\begin{aligned} f_0 &= 0 \\ f_n &= f_{n-1} + n^k \end{aligned}$$

এখানেও n^k পদটা ঝামেলা করছে। যদি $k = 1$ হত তাহলে কিন্তু আমরা আগের মতই $g_n = n$ এর রিকারেন্সটা বসিয়ে দিতে পারতাম। তাহলে আরেকটু কঠিন

কেস চিন্তা করি। $k = 2$ হলে কী করতাম? তখন আমাদের এমন একটি রিকারেন্স h লাগত যেন $h_n = n^2$ হয়। এটা বের করাও কিন্তু বেশ সহজ।

$$h_0 = 0$$
$$h_n = h_{n-1} + 2g_{n-1} + 1$$

এখানে আমরা $n^2 = (n-1)^2 + 2(n-1) + 1$ অভেদটি ব্যবহার করেছি। n^2 এর বদলে h_n, $(n-1)^2$ এর বদলে h_{n-1} এবং $(n-1)$ এর বদলে g_{n-1} বসিয়ে দিলেই রিকারেন্সটি পেয়ে যাব। একইভাবে আমরা n^3 এর রিকারেন্সটিও বের করতে পারি। p_n যদি n^3 এর রিকারেন্স হয়, তাহলে $n^3 = (n-1)^3 + 3(n-1)^2 + 3(n-1) + 1$ থেকে আমরা পাব

$$p_0 = 0$$
$$p_n = p_{n-1} + 3h_{n-1} + 3g_{n-1} + 1$$

প্যাটার্নটি কি বুঝতে পারছ। n^k কে আমরা $(n-1)$ এর বিভিন্ন পাওয়ার দিয়ে লেখছি। দ্বিপদী উপপাদ্য দিয়ে পরের রিকারেন্সগুলো সহজেই বের করে ফেলতে পারি। নিচের অভেদটি ব্যবহার করে $n^1, n^2, n^3, n^4, \ldots, n^k$ সবকিছুর জন্যই রিকারেন্স বের করতে পারব

$$n^m = \sum_{i=0}^{m} \binom{m}{i} (n-1)^i$$

সবমিলিয়ে আমরা $k+1$ টি রিকারেন্স পাব। সুতরাং আমাদের ম্যাট্রিক্সটি হবে একটি $(k+1) \times (k+1)$ ম্যাট্রিক্স। ম্যাট্রিক্স এক্সপনেন্সিয়েশনের দিয়ে আমরা সমস্যাটি $\mathcal{O}(k^3 \log n)$ এ সমাধান করতে পারি। k যদি বেশ ছোট হয় (যেমন $k \leq 50$) এবং n যদি অনেক বড় হয় (যেমন $n \leq 10^9$) তাহলে এভাবেই আমাদের সমস্যাটি সমাধান করতে হবে। ■

§ ২.৪ গ্রাফ থিওরি এবং ম্যাট্রিক্স

গ্রাফকে প্রকাশ করার জন্য অ্যাডজাসেন্সি ম্যাট্রিক্স প্রায় ব্যবহার করি। এই ম্যাট্রিক্স দিয়েও বেশ কিছু কাজ করা যায়। নিচের সমস্যাটি দেখ

সমস্যা ২.৩: ধর তোমার কাছে n টি নোডের একটি গ্রাফ দেওয়া আছে। গ্রাফ 1 নম্বর নোড থেকে n তম নোডে ঠিক k টি এজ ব্যবহার করে কতভাবে যাওয়া যায়?

সমাধান: প্রথমে আমরা ডাইনামিক প্রোগ্রামিং দিয়ে প্রবলেমটি চিন্তা করব। ধর $D_{k,i,j} =$ গ্রাফের নোড i থেকে নোড j তে ঠিক k টি এজ ব্যবহার করে কতভাবে যাওয়া যায়। এটা আমরা নিচের রিকারেন্স দিয়ে বের করতে পারি

$$D_{k,i,j} = \sum_{x=1}^{n} D_{k-1,i,x} \times A_{x,j}$$

যেখানে A হল আমাদের অ্যাডজাসেন্সি ম্যাট্রিক্স। এর ব্যাখ্যা হল প্রথমে আমরা i থেকে কোন একটি নোড x এ $k-1$ টি এজ ব্যবহার করে গিয়েছি। এ কাজটি করা যাবে $D_{k-1,i,x}$ উপায়ে। এরপর x থেকে আমরা j তে গিয়েছি একটিমাত্র এজ ব্যবহার করে। এ কাজটি করা যাবে $A_{x,j}$ উপায়ে, কেননা $A_{x,i} = 1$ হলে x আর j এর মধ্যে এজ বিদ্যমান, সুতরাং একভাবেই যে এজ ব্যবহার করে x থেকে j তে যাওয়া যাবে; আবার $A_{x,j} = 0$ হলে তাদের মধ্যে কোন এজ নাই, তাই শূন্য উপায়ে x থেকে j তে যাওয়া যাবে। দুটি গুন করলেই আমরা সর্বমোট উপায় পাব। আবার x তো কোন নির্দিস্ট নোড না, তাই $x = 1, 2, 3, \ldots, n$ সবার জন্যই $D_{k-1,i,x} \times A_{x,j}$ যোগ করতে হবে।

এটি দেখে কি ম্যাট্রিক্স গুনের কথা মনে পড়ে না? ম্যাট্রিক্স গুন কিন্তু আমরা প্রায় একইভাবে সংজ্ঞায়িত করেছিলাম। ধর $D_{(k)}$ ম্যাট্রিক্সের (i,j) তম এন্ট্রি $D_{k,i,j}$। তাহলে উপরের রিকারেন্সটিকে ম্যাট্রিক্স গুণফল দিয়েই আমরা প্রকাশ করতে পারি

$$D_{(k)} = D_{(k-1)} \times A$$

আবার D_1 এবং অ্যাডজাসেন্সি ম্যাট্রিক্স A কিন্তু একই ম্যাট্রিক্স। তাই

$$\begin{aligned} D_{(1)} &= A \\ D_{(2)} &= D_{(1)} \times A = A^2 \\ D_{(3)} &= D_{(2)} \times A = A^3 \\ &\cdot \end{aligned}$$

.

.

$$D_{(k)} = D_{(k-1)} \times A = A^k$$

অর্থাৎ গ্রাফের অ্যাডজাসেন্সি ম্যাট্রিক্স এর k তম পাওয়ার বের করলেই আমরা আমাদের উত্তর পেয়ে যাব!! কমপ্লেক্সিটি হবে $\mathcal{O}(n^3 \log k)$ ■

§ ২.৫ অন্যান্য সাব-রিং

একটা জিনিশ খেয়াল করে দেখেছ? আমরা কিন্তু ম্যাট্রিক্সের অ্যাসোসিয়েটিভিটি ছাড়া আর কোন ধর্মই ব্যবহার করিনি। সাধারণভাবে যেভাবে ম্যাট্রিক্স গুন সংজ্ঞায়িত করা হয় তাকে বলে হয় $(+, \times)$ সাব-রিং। কারণ A ও B এর গুনফল C বের করতে $A_{i,x}$ এবং $B_{x,j}$ গুন করে সেগুলো আমরা যোগ করছি। ম্যাট্রিক্স গুণফল অ্যাসোসিয়েটিভ কারণ যোগ এবং গুন দুটি অ্যাসোসিয়েটিভ অপারেটর। আমরা যদি যোগ, গুনের বদলে অন্য অ্যাসোসিয়েটিভ অপারেটর ব্যবহার করে ম্যাট্রিক্স গুণফল সংজ্ঞায়িত করতাম তাহলেও কিন্তু আমাদের ম্যাট্রিক্স গুণফল অ্যাসোসিয়েটিভই থাকত। একইভাবে আমরা ম্যাট্রিক্সের পাওয়ারও বের করতে পারব। এমন একটি বিশেষ সাব-রিং হচ্ছে $(\max, +)$ সাব-রিং। এই রিং-এ যদি $C = AB$ হয় তাহলে

$$C_{i,j} = \max_{x=1}^{m} \{A_{i,x} + B_{x,j}\}$$

হবে। এটিও আগের মতই অ্যাসোসিয়েটিভ হবে।

সমস্যা ২.৪: ধর তোমার কাছে n টি নোডের একটি ওয়েটেড গ্রাফ (weighted graph) দেওয়া আছে। গ্রাফ 1 নম্বর নোড থেকে n তম নোডে ঠিক k টি এজ ব্যবহার করে এমন শর্টেস্ট পাথের (shortest path) মান কত?

সমাধান: এটা কিন্তু প্রায় আগের সমস্যাটির মতই। যদি আমরা অ্যাডজাসেন্সি ম্যাট্রিক্স A এর $A_{i,j} = i$ এবং j এর মধ্যে এজের ওয়েট ধরি (যদি এজ না থাকে তাহলে এর মান ∞ হবে) এবং $D_{k,i,j} =$ গ্রাফের নোড i থেকে নোড j তে ঠিক k টি এজ ব্যবহার করে শর্টেস্ট পাথ ধরি তাহলে আমাদের

রিকারেন্সটি হবে

$$D_{k,i,j} = \max_{i=1}\{D_{k-1,i,x} + A_{x,j}\}$$

এর ব্যাখ্যাও ঠিক আগের সমস্যার মতই। শুধু পার্থক্য হচ্ছে $\sum$ এর বদলে $\max$ এবং $\times$ এর বদলে $+$ বসেছে এখানে। তাই এটিকে আমরা $(\max, +)$ সাব-রিং এর ম্যাট্রিক্স গুণফল হিসেবে চিন্তা করতে পারি। এই সাব-রিং এ A^k এর মান বের করলেই আমরা আমাদের উত্তর পেয়ে যাব! ■

§ ২.৬ শেষ কথা

ম্যাট্রিক্স কোড করার জন্য আমি সাধারণত একটা ক্লাস লেখে ফেলি। ক্লাসে তুমি যোগ, গুন এসব অপারেটর ওভারলোড করতে পারবে। আরেকটা ট্রিক হল যদি তোমাকে একই ম্যাট্রিক্স A এর পাওয়ার বারবার বের করতে হয় তাহলে $A^1, A^2, A^4, A^8, \ldots, A^{2^k}$ ম্যাট্রিক্স গুলো আগের বের করতে রাখতে পারো। এরপর পাওয়ারকে বাইনারিতে প্রকাশ করে তুমি বের করা ম্যাট্রিক্সগুলো দিয়েই যেকোনো পাওয়ার বের করতে পারবে। আবার তুমি এই ম্যাট্রিক্সগুলোকে সরাসরি ভেক্টরের সাথে গুন করতে পারো (অ্যাসোসিয়েটিভিটি!!)। দুটো $n \times n$ ম্যাট্রিক্স গুন করতে $\mathcal{O}(n^3)$ কমপ্লেক্সিটি লাগে, কিন্তু একটি $n \times n$ ম্যাট্রিক্সের সাথে একটি $n \times 1$ ভেক্টর গুন করতে $\mathcal{O}(n^2)$ কমপ্লেক্সিটি লাগছে। তাই অনেক সমস্যায় $A^1, A^2, A^4, A^8, \ldots, A^{2^k}$ বের করার পরে $\mathcal{O}(n^2 \log k)$ কমপ্লেক্সিটিতেই তুমি উত্তর বের করতে পারবে।

পড়া থামাও, নিজে চেষ্টা করো

তোমার কাছে একটি $1 \times n$ গ্রিড আছে এবং যথেষ্ট সংখ্যক 1×1 এবং 1×2 ডোমিনো আছে। কত ভাবে তুমি গ্রিডটিতে ডোমিনো গুলো বসাতে পারবে যেন একই ঘরে একাধিক ডোমিনো না থাকে। $(1 \leq n \leq 10^9)$

অধ্যায় ৩

ন্যাপস্যাক

§ ৩.১ $0/1$ ন্যাপস্যাক

ধর তোমার কাছে n টি বস্তু আছে, i তম বস্তুর ওজন w_i এবং দাম v_i। তোমার কাছে একটা ব্যাগ (ন্যাপস্যাক) আছে যা সর্বোচ্চ W ওজনের বস্তু ধারণ করতে পারে। এই ব্যাগে তুমি সর্বোচ্চ কত দামের বস্তু রাখতে পারবে?

একে $0/1$ ন্যাপস্যাক বলা হয়, কারণ এখানে প্রতিটি বস্তু সর্বোচ্চ একবারই নেওয়া যাবে। এটির জন্য আমাদের ডাইনামিক প্রোগ্রামিং এর সাহায্য নিতে হবে। ধরি $f_{i,j} =$ প্রথম i টি বস্তুর মধ্যে সর্বোচ্চ কত দামের বস্তু নেওয়া যায় যাতে বস্তুগুলোর ওজনের যোগফল $\leq j$ হয়। তাহলে আমাদের রিকারেন্সটি

$$f_{i,j} = \max\{f_{i-1,j},\ f_{i-1,j-w_i} + v_i\}$$

অর্থাৎ $f_{n,W}$ এর মানই হবে আমাদের অ্যান্সার। এখানে টাইম ও মেমরি কমপ্লেক্সিটি উভয়ই $\mathcal{O}(nW)$। তবে যেহেতু $f_{i,j}$ এর মান কেবলমাত্র $f_{i-1,0}\,,\ f_{i-1,1}\,,\ f_{i-1,2}\,,\ldots,\ f_{i-1,W}$ এর ওপর নির্ভর করে তাই $\mathcal{O}(W)$ মেমরি দিয়েও কাজটি করা সম্ভব। (মেমোরি অপটিমাইজেশনের চ্যাপ্টারটা দেখ)

§ ৩.২ 0-K ন্যাপস্যাক

ধর তোমার কাছে n টাইপের বস্তু আছে, i তম টাইপের বস্তু আছে k_i টি এবং এদের প্রত্যেকটির ওজন w_i এবং দাম v_i। তোমার কাছে একটা ব্যাগ

(ন্যাপস্যাক) আছে যা সর্বোচ্চ W ওজনের বস্তু ধারণ করতে পারে। এই ব্যাগে তুমি সর্বোচ্চ কত দামের বস্তু রাখতে পারবে?

আগেরটার সাথে এটার পার্থক্য হচ্ছে এখানে i তম বস্তু সর্বোচ্চ k_i সংখ্যক বার নেওয়া যাবে। এখানেও আগের মতই ডাইনামিক প্রোগ্রামিং ব্যবহার করা যায়, ধরি $f_{i,j} =$ প্রথম i টি বস্তুর মধ্যে সর্বোচ্চ কত দামের বস্তু নেওয়া যায় যাতে বস্তুগুলোর ওজনের যোগফল $\leq j$ হয়। তাহলে,

$$f_{i,j} = \max_{m=0}^{k_i}\{f_{i-1,j-w_i m} + v_i m\}$$

অর্থাৎ i তম বস্তু কতবার নিচ্ছি সেটার সবগুলো অপশন কনসিডার করতে হবে। আগেরটার কোড বুঝে থাকলে এটার কোড নিজেরই পারার কথা। এখানে টাইম কমপ্লেক্সিটি হবে $\mathcal{O}(W \times \sum k_i)$

কিন্তু এইখানে সমস্যা হচ্ছে $\sum k_i$ এর মান অনেক বড় হতে পারে। আশার কথা হল এই প্রবলেমের এইটাই সবচেয়ে অপটিমাল সলিউশন না। $\mathcal{O}(W \times \sum \log k_i)$ কমপ্লেক্সিটিতেও এই প্রবলেমটি সল্ভ করা সম্ভব।

আইডিয়াটি হচ্ছে প্রত্যেক k_i এর বাইনারি রিপ্রেজেন্টেশনকে ব্যবহার করা। একটি উদাহরণ দেখা যাক, ধর কোন এক টাইপের বস্তুর $(k_i, w_i, v_i) = (27, 13, 5)$। অর্থাৎ ঐ টাইপের বস্তু আছে 27 টি এবং তার ওজন 13 ও দাম 5। এখন 27 কে এইভাবে লেখা যায়:

$$27 = 11011_2 = 1111_2 + 1100_2 = (2^4 + 2^3 + 2^2 + 2^1 + 2^0) + 12$$

অর্থাৎ আমরা যদি $(27, 13, 5)$ বস্তুটির বদলে $(1, 13 \times 2^4, 5 \times 2^4)$, $(1, 13 \times 2^3, 5 \times 2^3)$, $(1, 13 \times 2^2, 5 \times 2^2)$, $(1, 13 \times 2^1, 5 \times 2^1)$, $(1, 13 \times 2^0, 5 \times 2^0)$ এবং $(1, 13 \times 12, 5 \times 12)$ বস্তুগুলোর ওপর ন্যাপস্যাক ডিপি চালাই তাহলে উত্তর চেঞ্জ হবে না, এর কারন হচ্ছে 2^4, 2^3, 2^2, 2^1, 2^0 এবং 12 দিয়ে 0 থেকে 27 পর্যন্ত সব সংখ্যা কে লেখা যায়, তবে 27 এর বড় কোন সংখ্যাকে লেখা যায় না (কিছু কিছু সংখ্যাকে একাধিক উপায়ে লেখা যেতে পারে, কিন্তু সেটা আমাদের জন্য সমস্যা না)। এইভাবে প্রতিটি বস্তুকে তার বাইনারি

রিপ্রেজেন্টেশন অনুযায়ী ভেঙ্গে দিতে হবে। ভেঙ্গে দেওয়ার পর কিন্তু আমাদের আর 0-K ন্যাপস্যাক থাকছে না, 0-1 ন্যাপস্যাক হয়ে যাচ্ছে। কারণ ভেঙ্গে দেওয়ার পর প্রত্যেক বস্তুকে সর্বোচ্চ একবারই নেওয়া সম্ভব $(k_i = 1)$। অর্থাৎ ভেঙ্গে দেওয়ার পর আমাদের মোট বস্তু হবে $\mathcal{O}(\sum \log k_i)$ টি। তাই 0-1 ন্যাপস্যাক এর কমপ্লেক্সিটি হবে $\mathcal{O}(W \times \sum \log k_i)$।

মজার ব্যাপার হল এই প্রবলেমের $\mathcal{O}(W \times \sum \log k_i)$ এর চেয়েও ভাল সলিউশন আছে। $\mathcal{O}(nW)$ কমপ্লেক্সিটিতেও 0-K ন্যাপস্যাক সল্ভ করা সম্ভব। রিকারেন্সটি আবার লক্ষ্য করি:

$$f_{i,j} = \max_{m=0}^{k_i}\{f_{i-1,j-w_i m} + v_i m\} \quad (1)$$

কোনো ফিক্সড i এর জন্য $f_{i,0}$, $f_{i,1}$, ... , $f_{i,W}$ এর মান যদি আমরা $\mathcal{O}(W)$ তে বের করতে পারি, তাহলেই $\mathcal{O}(nW)$ কমপ্লেক্সিটি হয়ে যাবে। এখন লক্ষ্য করি, $f_{i,j}$ এর মান $f_{i-1,j}$, $f_{i-1,j-w_i}$, $f_{i-1,j-2w_i}$, $f_{i-1,3w_i}$, ... মানগুলোর ওপর নির্ভর করে। অন্যভাবে বলা যায় $f_{i,j}$ এর মান এমন সব $f_{i-1,p}$ এর মানের ওপর নির্ভর করে যাতে $p \equiv j \mod w_i$ হয়। এটাকে কাজে লাগিয়েই $\mathcal{O}(W)$ তে কাজটি করা সম্ভব। আমরা $f_{i,j}$ এর মান $0 \le j \le W$ এর জন্য একসাথে বের না করে w_i এর প্রত্যেক মডুলো ক্লাসের জন্য আলাদা ভাবে বের করতে পারি। বুঝানোর সুবিধার্তে ধরি,

$$g_m(i,j) = f_{i,m+jw_i}$$

যেখানে $0 \le m < w_i$। এখন আমরা একটা ফিক্সড m এর জন্য $g_m(i,j)$ এর সকল মান বের করব, যেখানে $0 \le m + jw_i \le W$। (1) নং রিকারেন্সের সাহায্যে $g_m(i,j)$ কে এইভাবে লেখা যায়:

$$\begin{aligned} g_m(i,j) &= \max_{h=j-k_i}^{j}\{g_m(i-1,h) + (j-h)v_i\} \\ &= \max_{h=j-k_i}^{j}\{g_m(i-1,h) - hv_i\} + jv_i \end{aligned}$$

এখান থেকেই বুঝা যাচ্ছে $g_m(i-1,0), g_m(i-1,1)-v_i, g_m(i-1,2)-2v_i, \ldots$ এর প্রতিটি k_i+1 দৈর্ঘ্যের সাবঅ্যারের মিনিমাম ভ্যালু বের করতে পারলেই $g_m(i,j)$ এর সকল মান আমরা সহজেই বের করতে পারব। কোনো n দৈর্ঘ্যের অ্যারের প্রতিটি m দৈর্ঘ্যের সাবঅ্যারের মিনিমাম (বা ম্যাক্সিমাম) ভ্যালু $\mathcal{O}(n)$ এই বের করা যায় (স্লাইডিং উইন্ডোর সাহায্যে)। অর্থাৎ প্রত্যেক মডুলো ক্লাসের জন্য আমরা লিনিয়ার টাইমেই g_m এর মান বের করতে পারব। যেহেতু প্রত্যেকটি সংখ্যাই কেবলমাত্র একটি মডুলো ক্লাসের অন্তর্ভুক্ত তাই ওভারঅল কমপ্লক্সিটি হবে $\mathcal{O}(W)$। তাই প্রত্যেকটি i এর জন্য $f_{i,j}$ এর মান বের করতে $\mathcal{O}(nW)$ কমপ্লেক্সিটি প্রয়োজন।

§ ৩.৩ সাবসেট সাম:

এই সেকশনের সব জায়গায় সেট বলতে মাল্টিসেট বুঝান হবে। অর্থাৎ সেটে একই উপাদান একাধিক বার থাকতে পারে।

ন্যাপস্যাকের সবচেয়ে গুরুত্বপূর্ণ ভ্যারিয়েশন এটি। ধর তোমার কাছে n দৈর্ঘ্যের একটা অ্যারে a এবং একটি নাম্বার m দেওয়া আছে। তোমাকে বলতে হবে a এর নাম্বার গুলো ব্যবহার করে যোগফল m বানানো যায় কিনা।

অর্থাৎ $S=\{1,2,3,\ldots,n\}$ হলে এমন কোন সাবসেট T পাওয়া সম্ভব কিনা যাতে $T \subseteq S$ এবং $\sum_{i\in T} a_i = m$ হয়।

ধরি,

$$f_{i,j} = \begin{cases} 1, & \text{যদি প্রথম } i \text{ টি সংখ্যা হতে যোগফল } j \text{ বানানো সম্ভব হয়,} \\ 0, & \text{সম্ভব না হয়.} \end{cases}$$

তাহলে,

$$f_{i,j} = f_{i-1,j} \vee f_{i-1,j-a_i}$$

$\vee$ এখানে or অপারেটরটাকে বুঝাচ্ছে। তাহলে এই ডিপিটা ক্যালকুলেট করতে আমাদের $\mathcal{O}(nm)$ টাইম ও $\mathcal{O}(m)$ মেমরি লাগছে। তবে এই সলিউশন কে অপটিমাইজ করার জন্য আরেকটা সস্তা অপটিমাইজেশন আছে। তা হল `bitset` ব্যবহার করা। `bitset` ব্যবহার করলে টাইম কমপ্লেক্সিটি দাড়ায় $\mathcal{O}(\frac{nm}{64})$ এবং মেমোরি কমপ্লেক্সিটি দাড়ায় $\mathcal{O}(\frac{m}{64})$।

§ ৩.৪ ডাইনামিক সাবসেট সাম:

ধর সাবসেট সাম প্রবলেমটায় তোমাকে কিছু আপডেট আর কুয়েরিও দেওয়া হল। অর্থাৎ প্রত্যেক আপডেটে তোমাকে একটি সংখ্যা p দেওয়া হবে এবং তোমাকে সংখ্যাটাকে সেটে অ্যাড করতে হবে অথবা সেট থেকে রিমুভ করতে হবে। প্রত্যেক কুয়েরিতে তোমাকে একটি সংখ্যা r দেওয়া হবে এবং তোমাকে বলতে হবে r সংখ্যাটিকে সেটের সংখ্যাগুলোর যোগফল হিসেবে লেখা যায় কিনা।

ধরা যাক মোট আপডেট ও কুয়েরি Q টি। তাহলে যদি আমরা Q বারই সাবসেট সাম-এর ডিপি টা নতুন করে আপডেট করি তাহলে কমপ্লেক্সিটি $\mathcal{O}(\frac{Qnr_{\max}}{64})$ হয়ে যাচ্ছে। তবে এই প্রবলেমটি $\mathcal{O}(Qr_{\max})$ টাইমেও করা সম্ভব, যেখানে $r_{\max}$ হল r এর ম্যাক্সিমাম ভ্যালু।

এর জন্য আমাদের ডিপি টাকে একটু চেঞ্জ করতে হবে। ধরি, $f_j =$ সেটে যেসব উপাদান আছে তাদের কোনো সাবসেট নিয়ে কতভাবে j সংখ্যাটি বানানো যায়। তাহলে প্রত্যেক কুয়েরিতে $f_r > 0$ কিনা তা চেক করলেই হচ্ছে আমাদের। আর যদি নতুন কোন নাম্বার অ্যাড বা রিমুভ করতে হয় তাহলে নরমাল সাবসেট সাম ডিপির মতই f_j এর মান আপডেট করা যায়। এখন সমস্যা হচ্ছে f_j মান অনেক বড় হয়ে যেতে পারে, এমনকি `long long` এও আটবে না। তাই f_r কে আমরা $\mod P$ ক্যালকুলেট করব যেখানে P র‍্যানডম কোন প্রাইম নাম্বার। এখন যদি $f_r = 0$ হয়, এবং তারপরেও r কে যোগফল হিসেবে লেখা যাবে সেটির সম্ভাবনা নেয় বললেই চলে। (কেউ চাইলে ২-৩ টি `mod` ও ব্যবহার করতে পারে)।

§ ৩.৫ $\mathcal{O}\left(s\sqrt{s}\right)$ সাবসেট সাম:

এখানে s সেটের সবগুলো সংখ্যার যোগফল বুঝাচ্ছে। যদি কোন সংখ্যা t এর থেকে বড় হয়, তাহলে আমরা নরমালি `bitset` দিয়ে ডিপি টা আপডেট করব, এটি করতে $\mathcal{O}\left(\frac{s}{64} \times \frac{s}{t}\right)$ কমপ্লেক্সিটি লাগে (কারন t এর থেকে বড় সংখ্যা সর্বোচ্চ $\frac{s}{t}$ বার পাওয়া যাবে)। আর যদি t এর থেকে ছোট হয় তাহলে আমরা 0-k ন্যাপস্যাক এর মত ডিপি টাকে আপডেট করব। অর্থাৎ t এর থেকে ছোট কোন সংখ্যা কতবার আছে সেটা বের করে তার ওপর 0-k ন্যাপস্যাক প্রয়োগ করব। এ কাজটি করতে সর্বোচ্চ $\mathcal{O}(st)$ কমপ্লেক্সিটি লাগে। $t = \sqrt{\frac{s}{64}}$ হলে টোটাল কমপ্লেক্সিটি দাড়ায়:

$$\mathcal{O}\left(\frac{s}{64} \times \frac{s}{t} + s \times t\right) = \mathcal{O}\left(s\sqrt{\frac{s}{64}}\right)$$