সেগমেন্ট ট্রি ব্যাবহার করে 2D ডাইনামিক প্রোগ্রামিং অপটিমাইজেশন

তাসমীম রেজা

২ জানুয়ারী ২০১৯

ধর তোমাকে একটি রিকারেন্স রিলেশন $f(i,j)$ ক্যালকুলেট করতে হবে যাতে i এর n টি সম্ভাব্য মান আছে এবং j এর m টি সম্ভাব্য মান আছে। তাহলে রিকারেন্সটি ক্যালকুলেট করতে সর্বনিম্ন $O(nm)$ কমপ্লেক্সিটি লাগবেই। তবে কিছু ক্ষেত্রে এই রিকারেন্সটি সেগমেন্ট ট্রি এর সাহায্যে $O(n \log m)$ এও ক্যালকুলেট করা সম্ভব।

যেসব ক্ষেত্রে এই অপটিমাইজেশনটি সম্ভব :

প্রথমে দেখা যাক মেমোরি কমপ্লেক্সিটি কিভাবে কমানো যায়। $f(i,j)$ রিকারেন্সটির মেমোরি কমপ্লেক্সিটি $O(nm)$ । তবে যদি $f(i,j)$ কে ক্যালকুলেট করার জন্য শুধুমাত্র $f(i-1,x)$ এর মানের প্রয়োজন হয় (যেখানে x যেকোনো মান হতে পারে) তাহলে শুধুমাত্র আগের row টি-কে স্টোর করলেই হচ্ছে। অর্থাৎ এই ধরনের মেমরি অপটিমাইজেশন করা সম্ভব হলেই সেগমেন্ট ট্রি দিয়ে টাইম কমপ্লেক্সিটিকেও কিছুক্ষেত্রে অপটিমাইজ করা সম্ভব। সুতরাং যদি আমাদের কাছে একটি অ্যারে F এ $f(i,1),\ f(i,2),\ f(i,3),\dots f(i,m)$ এর ভ্যালুগুলো ক্যালকুলেট করা থাকে $(F[x]=f(i,x))$, তাহলে আমরা $O(m)$ লুপের মাধ্যমে F অ্যারের ভ্যালুগুলোকে $f(i+1,1),\ f(i+1,2),\ f(i+1,3),\dots f(i+1,m)$ এ আপডেট করতে পারি। এইভাবে n বার লুপ চালালে $f(n,1),\ f(n,2),\ f(n,3),\dots f(n,m)$ এর ভ্যালুগুলো পাওয়া যাবে। মেমরির জন্য এখানে আমাদের শুধুমাত্র F অ্যারেটিই যথেষ্ট, অর্থাৎ $O(m)$ মেমরি দিয়েই আমরা রিকারেন্সটি ক্যালকুলেট করতে পারি। টাইম কমপ্লেক্সিটি ইমপ্রুভ করার জন্য আমাদের মূল আইডিয়াটি হল F অ্যারেটিকে আপডেট করার জন্য $O(m)$ লুপ না চালিয়ে সেগমেন্ট ট্রি ব্যবহার করা।

উদাহরন:

সমস্যা ১

তোমার কাছে n $(\leq 10^5)$ টি বক্স এবং দুটি বল আছে। বক্সগুলো 1 থেকে n পর্যন্ত নাম্বার করা আছে। শুরুতে তোমার বল দুটি বক্স নাম্বার A এবং B তে রাখা আছে। এখন তোমার বন্ধু তোমাকে Q $(\leq 10^5)$ টি অপারেশন করতে বলবে। i তম অপারেশনে তোমাকে সে একটি বক্স নাম্বার p_i বলবে এবং তোমাকে ২ টি বলের মধ্যে যেকোনো একটি বল নিয়ে p_i তম বক্সটিতে রাখতে হবে। X তম বক্স থেকে Y তম বক্সে কোন বল রাখতে তোমার $|X - Y|$ সময় লাগে। তুমি আগে থেকেই জান তোমার বন্ধু কোন অপারেশনে কোন বক্সটি সিলেক্ট করবে। তোমার বন্ধুর দেওয়া সবগুলো অপারেশন সম্পন্ন করতে তোমার সর্বনিম্ন কত সময় লাগবে?

সমাধান: এই প্রবলেমটিতে ডাইনামিক প্রোগ্রামিং সলিউশন কী হতে পারে? সবচেয়ে obvious সলিউশন হলো $f(i, a, b)$ = প্রথম i টি অপারেশন সম্পন্ন করতে সর্বনিম্ন সময় যেখানে একটি বল থাকবে a তম বক্সে এবং অপর বলটি থাকবে b তম বক্সে। তবে এইখান থেকে একটি state সহজেই কমানো যায়, কারণ $i \geq 1$ হলে a এবং b এর মধ্যে অন্তত একটিকে p_i এর সমান হতে হবে।

অর্থাৎ আমাদের DP টি দাড়ায় $f(i, a)$ = প্রথম i টি অপারেশন করতে সর্বনিম্ন সময় যেখানে একটি বল থাকবে p_i তম বক্সে এবং অপর বলটি থাকবে a তম বক্সে। অর্থাৎ,

$$f(i+1, a) = \begin{cases} \min_{y=1}^{n}\{f(i, y) + |p_{i+1} - y|\}, & \text{if } a = p_i, \\ f(i, a) + |p_{i+1} - p_i|, & \text{otherwise.} \end{cases}$$

ধরি আমাদের কাছে $f(i, 1),\ f(i, 2),\ f(i, 3), \ldots f(i, n)$ এর ভ্যালুগুলো ক্যাল্কুলেট করা আছে। এখন দেখা যাক সেগমেন্ট ট্রি দিয়ে এই ভ্যালু গুলো কে কিভাবে $f(i+1, 1),\ f(i+1, 2),\ f(i+1, 3), \ldots f(i+1, n)$ এ আপডেট করা যায়।

প্রথম কেসটি হল $a = p_i$ অর্থাৎ $f(i+1, p_i)$ আপডেট করা। প্রথম কেসটির জন্য ডিপি টাকে এইভাবে লেখা যায়

$$f(i+1, p_i) = \min\{p_{i+1} + \min_{y=1}^{p_{i+1}}(f(i, y) - y), -p_{i+1} + \min_{y=p_{i+1}+1}^{n}(f(i, y) + y)\}$$

অর্থাৎ $f(i+1, p_i)$ এর ভ্যালু বেসিক্যালি $f(i, y) - y$ এবং $f(i, y) + y$ এর ভ্যালু গুলোর ওপর যথাক্রমে $[1, p_{i+1}]$ এবং $[p_{i+1} + 1, n]$ রেঞ্জে রেঞ্জ মিনিমাম কুয়েরি করলেই পাওয়া

যাবে। তারপর সেগমেন্ট ট্রি তে p_i তম পজিশনে আপডেট করে দিতে হবে। (কমপ্লেক্সিটি $O(\log n)$)

দ্বিতীয় কেসটি হল $a \neq p_i$, এক্ষেত্রে $|p_{i+1} - p_i| = c$ ধরলে ডিপি টিকে এইভাবে লেখা যায়

$$f(i+1, a) = f(i, a) + c$$

অর্থাৎ সেগমেন্ট ট্রি দিয়ে $f(i, a)$ (যেখানে $a \neq p_i$) এর প্রতিটি ভ্যালুর সাথে c যোগ করে দিলেই তা $f(i+1, a)$ এ কনভার্ট হয়ে যাবে। (কমপ্লেক্সিটি $O(\log n)$)

সুতরাং আমাদের $f(i, a) + a$ এবং $f(i, a) - a$ ভ্যালু গুলোর ওপর সেগমেন্ট ট্রি মেইনটেইন করলেই হচ্ছে। সেগমেন্ট ট্রিতে $f(i, \dots)$ থেকে $f(i+1, \dots)$ এ কনভার্ট করতে আমাদের $O(\log n)$ কমপ্লেক্সিটি লাগছে। অর্থাৎ টোটাল কমপ্লেক্সিটি হবে $O(Q \log n)$

সমস্যা ২

তোমার কাছে n টি রেঞ্জ আছে, i তম রেঞ্জ হল $[l_i, r_i]$। তোমাকে কিছু বিন্দু সিলেক্ট করতে হবে যেন প্রতিটি রেঞ্জে অন্তত একটি বিন্দু থাকে। কাজটি করার জন্য মিনিমাম কয়টি বিন্দু সিলেক্ট করতে হবে এবং কতভাবে মিনিমাম সংখ্যক বিন্দু সিলেক্ট করা সম্ভব। $(n \leq 10^5, 1 \leq l_i \leq r_i \leq 10^6)$

সমাধান: প্রবলেমটি পোলিশ অলিম্পিয়াড ইন ইনফরমেটিক্স এর কোন এক রাউন্ডের প্রশ্ন। যদিও প্রবলেমটিতে কিছু লিমিট আরো বড় ছিল, তবুও বুঝানোর সুবিধার্থে এই কিছু লিমিট ছোট করে দিয়েছি। শুধুমাত্র মিনিমাম কয়টা বিন্দু প্রয়োজন তা বের করার জন্য একটা সিম্পল গ্রিডি অ্যালগরিদমই যথেষ্ট। প্রবলেমটার মেইন ডিফিকালটি হল কতভাবে মিনিমাম বিন্দু সিলেক্ট করা যায় তা বের করা। ধরা যাক মিনিমাম কয়টা বিন্দু প্রয়োজন সেটাও আমরা ডিপি দিয়ে বের করব, তাহলে ডিপি স্টেট কি হতে পারে? এজন্য প্রথমে আমাদের কিছু রেঞ্জ কে বাদ দিতে হবে।

অবজারভেশন ১: যদি কোন রেঞ্জ $[l_i, r_i]$ এর সম্পূর্ণ ভিতরে আরেকটি রেঞ্জ $[l_j, r_j]$ থাকে অর্থাৎ $l_i \leq l_j \leq r_j \leq r_i$ হয় তাহলে $[l_i, r_i]$ কে বাদ দিয়ে দিলেও অ্যানসার চেঞ্জ হয় না।

অবজারভেশন ২: অবজারভেশন ১ ব্যবহার করে ওইরকম সব রেঞ্জ বাদ দিয়ে দেওয়ার পরে যদি দুটি রেঞ্জ $[l_i, r_i]$ এবং $[l_j, r_j]$ পাওয়া যায় তাহলে $l_i < l_j$ হলে $r_i < r_j$ হবে।

অর্থাৎ রেঞ্জ গুলোকে l_i অনুযায়ী সর্ট করলে তারা r_i অনুযায়ীও sorted অবস্থায় থাকবে। অবজারভেশন ১ ব্যবহার করে ওইরকম সব রেঞ্জ বাদ দেওয়ার পরে l_i অনুযায়ী সর্ট করি। এরপর ধরা যাক $f(i, j) =$ প্রথম i টি রেঞ্জ থেকে মিনিমাম কয়টি বিন্দু সিলেক্ট করতে হবে যেন শেষ বিন্দুটি j হয়। j অবশ্যই l_i থেকে r_i এর মধ্যে হবে, অর্থাৎ $l_i \leq j \leq r_i$। $c = \min_{x=l_{i-1}}^{r_{i-1}} f(i-1, x)$

ধরে নিলে,

$$f(i,j) = \begin{cases} f(i-1,j), & \text{if } j \le r_{i-1}, \\ 1+c, & \text{otherwise.} \end{cases}$$

অনুরূপ ভাবে ধরি, $g(i,j) =$ প্রথম i টি রেঞ্জ থেকে কতভাবে মিনিমাম সংখ্যক বিন্দু সিলেক্ট করা যায় যেন শেষ বিন্দুটি j হয়। এক্ষেত্রেও $l_i \le j \le r_i$ হবে। ধরি

$$d = \sum_{\substack{l_{i-1} \le x \le r_{i-1} \\ f(i-1,x)=c}} g(i-1,x)$$

তাহলে,

$$g(i,j) = \begin{cases} g(i-1,j), & \text{if } j \le r_{i-1}, \\ d, & \text{otherwise.} \end{cases}$$

সেগমেন্ট ট্রি দিয়ে সহজেই c এর ভ্যালু বের করা যায়। d এর ভ্যালু বের করার জন্য সেগমেন্ট ট্রি এর প্রতিটি নোডে দুটি ভ্যালু স্টোর করতে হবে। সেগমেন্ট ট্রি এর নোডটির রেঞ্জ যদি $[b,e]$ হয়, তাহলে প্রতিটি নোডে নিচের ভ্যালু দুটি স্টোর করতে হবে,

$$C(b,e) = \min_{x=b}^{e} f(i,x)$$

$$D(b,e) = \sum_{\substack{b \le x \le e \\ f(i,x) = C(b,e)}} g(i,x)$$

এবং চাইল্ড নোড গুলো থেকে $C(b,e)$ ও $D(b,e)$ ক্যালকুলেট করার জন্য,

$$m = \frac{b+e}{2}$$

$$C(b,e) = \min\{C(b,m), C(m+1,e)\}$$

$$D(b,e) = \begin{cases} D(b,m), & \text{if } C(b,m) < C(m+1,e), \\ D(m+1,e), & \text{if } C(b,m) > C(m+1,e) \\ D(b,m) + D(m+1,e), & \text{if } C(b,m) = C(m+1,e). \end{cases}$$

ওভারঅল কমপ্লেক্সিটি হবে $O(n \log r_{\max})$

অনুরূপ সমস্যা:

Batman and Tree : প্রবলেমটি অন্য প্রবলেম গুলোর তুলনায় কিছুটা ভিন্ন, কারণ এখানে ট্রি এর ওপর ডিপি করতে হয়। ডাটা স্ট্রাকচার হিসেবে সেগমেন্ট ট্রিই যথেষ্ট, তবে অয়লার অর্ডারও প্রয়োজন হয়।
Slalom : আইডিয়া আগের গুলোর মতই, তবে ইমপ্লিমেন্ট করা কিছুটা কঠিন। সেগমেন্ট ট্রি এর সাথে সাথে লাইন সুইপ অ্যালগরিদম এরও প্রয়োজন হয়।
Many Moves : সমস্যা ১ এর লিঙ্ক
Conductor : সমস্যা ২ এর লিঙ্ক