Задача 1. Церемония

Автор: Тодор Петров Петров

Да започнем с това как да представяме ефикасно една ротация, така че същото представяне да може да се използва, за да се изразят няколко ротации направени една след друга. Логично е, че ако използваме представяне, което казва всяка една позиция от квадрата къде отива, то ще можем да изразим и последоватленост от ротации. В този смисъл представянето: *a1, a2, ..., a16* е точно такова представяне, a1,.....,a16 са числата от 1 до 16 и показват къде отива всяка една клетка след ротацията.

Клетките на квадарата са номерирани по следния начин:

1	2	3	4
5	6	7	8
9	10	11	12
13	14	15	16

Например ако имаме ротацията (1, 2, 2), това може да се представи като: *2, 6, 3, 4, 1, 5, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16* (1 отива в 2, 2 отива в 6, 6 отива в 5, 5 отива в 1 и всички останали отиват в себе си).

Сега да разгледаме какво се случва ако приложим няколко ротациии едновременнно. Например (1, 1, 2), (2, 2, 3) и (1, 1, 4). Това е еквивалентно на:

(2, 6, 3, 4, 1, 5, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16), (1, 2, 3, 4, 5, 7, 8, 12, 9, 6, 11, 16, 13, 10, 14, 15) и (2, 3, 4, 8, 1, 6, 7, 12, 5, 10, 11, 16, 9, 13, 14, 15)

И остава да видим как да обединим тези три ротации направени една след друга като една еквивалентна.

Почваме с клетка 1, при ротация 1 – тя отива в 2, при ротация 2 клетка 2 отива в себе си и при ротация 3 клетка 2 отива в 3 => като приложим трите ротации все едно прилагаме ротация при която клетка 1 отива в 3. Същото правим за всички останали числа и получаваме следната еквивалентна ротация: (*3, 7, 4, 8, 2, 1, 12, 16, 5, 6, 11, 15, 9, 10, 13, 14*).

След като уточнихме представянето сега можем да прибегнем към истинското решение. Тази задача изключително прилича на задачата, в която трябва да се смятат бързо суми от числа като числата постоянно може да се менят и след всяка промяна трябва новият резултат да се смята бързо. Някои от състезателите вече сигурно се сещат, че тази задача се решава с индексно дърво. За тези, които не знаят какво е индексно дърво, в Приложение А съм копирал статия, която е точно за тази структура – какво представлява и как се използва.

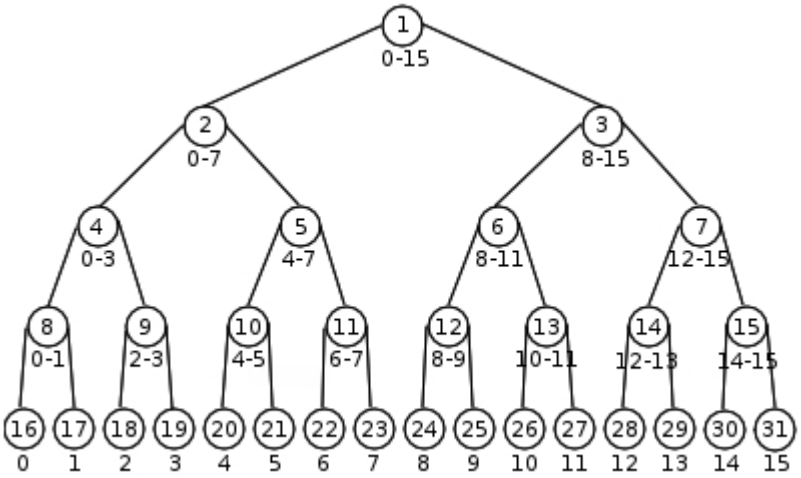
Е в нашата задача, ние също трябва да изпозлваме индексно дърво като вместо числа имаме ротации представени по начина, по който описахме по-горе (поредица от 16 елемента). По този начин ние може да пази еквивалентните ротации на ротациите в даден интервал и при всяка промяна със сложност logN да преструктурираме дървото. Нещо повече след всяка промяна последователността, която трябва да изпечатаме на екрана лесно може да се получи имайки ротацията във върха на дървото.
Единственото нещо, което остана да уточним е как допълваме дървото да има брой листа – степен на двойката. В обикновенното индексно дърво се допълва с нули, а тук можем да допълним с ротации от типа (r, c, 1), които сами по себе си не променят нищо тъй като всеки елемент отива в себе си.
Сложността на този алгоритъм е приблизително O(16*N + 16*Q*logN), което влиза в ограниченията.

Приложение А (Индексни дървета):

Въведение

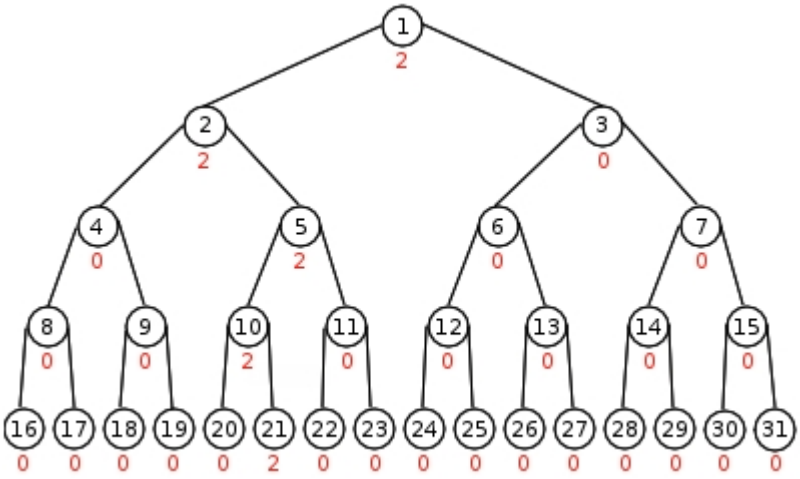
В последните години много често се срещата задачи, които се решават с помощта на така наречените индексни дървета. Индексните дървета са изключително полезни, защото чрез тях лесно се добавя, променя или извлича информация за O(LogN) време, където N е броят на върховете.

Индексно дърво е пълно двоично **дърво**, в което всяко листо отговаря за един елемент с дадент индекс, а всеки друг връх отговаря за интервала образуван от двата му наследника(Фиг. 1). Индексното **дърво**(както и всяко пълно двоично **дърво**) лесно се имплементира с помощта на едномерен масив, като всяка клетка от масива отговаря на връх от дървото. Наследниците на всеки връх (i) са върхове с номера (2*i) и (2*i+1), а неговият предшественик(баща му) е (i/2)(целочислено делене).

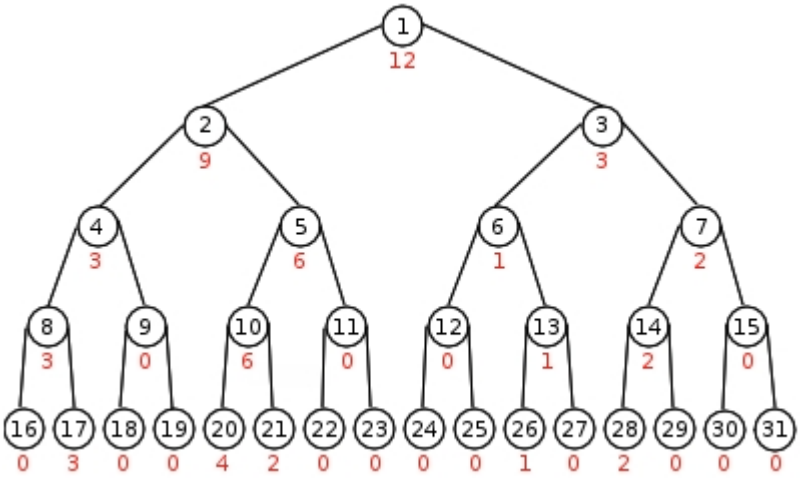


Фиг. 1 и Фиг.2

Първо да покажем как се добавя елемент елемент лесно в индексното **дърво**. Първо добавяме стойността в листото, което отговаря за елемента. След това обаче трябва да я добавим и във всички интервали, в които участва индексът. Т.е трябва да го добавим в бащата на листото. След това в бащата на бащата на листото и така повтаряме докато не стигнем до върха, който отговаря за всички индекси(връх с номер 1). Тъй като минаваме през LogN върха сложността на намирането на f(i) е O(LogN), където е броят на върховете в дървото.



Например нека добавим в индекс 5 стойност 2. Първо намираме листото, което отговаря за индекс 5. Номерът на листото е 16+5 = 21(16 е броят на индексите и трябва винаги да е степен на две). Добавяме стойност 2 във връх номер 21. След това намираме бащата на връх с номер 21. Баща му е 21/2 = 10(целочислено делене). Добавяме 2 във връх 10 и продължаваме нататък. Следвайки същите стъпки, ще добавим 2 във върхове с номера 5, 2 и 1. Така добавихме 2 във всички върхове, който отговарят за интервали съдържащи индекс 5(връх 21 отговаря за интервал [5-5], връх 10 отговаря за интервал [4-5], връх 5 отговаря за интервал [4,7], връх 2 за интервал [0-7] и връх 1 за интервал [0-15].



Ето как изглежда дървото след добавяне на стойност 2 в индекс 5, стойност 3 в индекс 1, стойност 4 в индекс 4, стойност 1 в индекс 10 и стойност 2 в индекс 12.

Сега да видим как можем да намерим сумата на стойностите във всички индекси, които са по-малки(вляво) от даден индекс i, нека я означим с f(i). Проверяваме дали върхът, който отговаря за i е ляв или десен наследник на баща си(ако е десен наследник номера на върха е нечетен). Ако върхът е десен наследник, то левият наследник отговаря за по-малки

индекси от i следователно трябва да добавим стойноста на левия наследник. Повтаряме тази стъпка, докато стигне връх с номер 1. Тъй като минаваме през LogN върха сложноста на намирането на f(i) е O(LogN), където е броят на върховете в дървото.

Например за i=11. Ще минем през върхове 27, 13, 6, 3 и 1 и ще добавим стойностите във върхове 26, 12 и 2(защото минава през техния десен брат) и ще получим, че стойността на всички индекси вляво от индекс 11 е f(11)= 0+1+9 = 10.

За да илюстрираме по-добре широката приложимост на индексните дървета, ще ги приложим в няколко задачи.

Задача Circle

Първата задача е от Circle от Пролетния Турнир 2003г. Накратко се уловието е: Дадена е окръжност с 2N точки на нея, като всяка точка е свързна с точно една друга точка. Да се намери броят на пресечните точки, ако се знае, че никой три прави не се пресичат в една точка. Т.е. задачата се свежда до намиране на броя на пресичащите се отсечки.

В конкретната задача, за да проверим дали една отсечка се пресича с друга, не ни трябват никакви геометрични познания, достатъно ни е да проверим дали едната точка от едната отсечка, лежи между индексите на двете точки от другата отсечка. От тук лесно се вижда как задачата може да се реши за O(N*N) време – като за всеки две прави проверим дали се пресичат. За съжаление(на някои състезателите) и за щастие на почитателите на красотата в информатиката, при ограничение N<=500 000 решение с такава сложност е твърде бавно. За това прилагайки индексни дървета ще предложим друго, което е със сложност O(N*LogN).

Нека по-малкия индекс на всяка отсечка i да означим с Li, а по-големия с Ri(Li<Ri). Първо, за да си улесним живота, сортираме отсечките(интервалите) по L, така че ако i<j, то Li < Lj. След това обхождаме отсечките една по една. За да намерим всички точки, които пресичат права i са преди нея в редицата, трябва да намерим броя на точките Rj (j<i), които се намират между Li и Ri. Забележете, че между Li и Ri няма нито една точка Lj, такава че (j<i) заради сортировката, която направихме предварително.

Първо в индексното **дърво** добавяме отсечка k, като на място(индекс) Lk добавим стойност -1, а на място(индекс) Rk добавим стойност 1. Така лесно ще можем да проверим колко са отсечките j(j<i), които се пресичат с отсечка i. За целта трябва да намерим сумата от стойностите между индекси Li и Ri(като сме добавили вече всички отсечки j). За да намерим направим това е достатъчно да намерим f(Ri)-f(Li).

За да докажем, че горният алгоритъм работи ще разгледаме трите възможни случаи за разположението на Rj и интервала Li и Ri(Lj задължително е по-малко от Li, след като j<i).

- Първи случай, ако Rj < Li, то след като добавим -1 във индекс Lj и 1 във Rj, то това няма да проемни нито f(Li) нито f(Ri)(-1 + 1 = 0) следователно няма да броим отсечка j като пресичаща се с отсечка i.

- Втори случай, ако Rj>Ri добавянето на -1 в индекс Lj ще добави към f(Li) и f(Ri) -1, докато добавянето на 1 в индекс Rj няма да повлияе. В крайна сметка разликата f(Ri) - f(Li) няма да бъде променена от отсечка j следователно няма да бъде броена за пресичаща се с отсечка i.

- Трети случай, ако Li<Lj<Ri. То добавяенто на -1 в индекс Lj и добавянето на 1 в индекс Rj. Няма да засегане f(Ri), но ще прибави -1 към f(Li), следователно f(Ri)-f(Li) ще се увели с 1 при добавянето на тосечка j. Така, че отсечка j се пресича с отсечка i.

Тъй като извършваме предварителна сортировка за O(N*LogN) и за всяка отсечка прилагаме едно добавяне и едно намиране на f(i), то крайната сложност на алгоритъма е O(N*LogN).

Задача Mobiles

Задачата Mobiles е давана на IOI 2001 и показва едно доста красиво приложение на индексни дървета. Накратко условието на задачата е даден е квадратна матрица N*N и от нас се изиска да можем да прилагаме две операции, добавяне на стойност в клека с кординати (x,y). И проверка на сумата от стойности в правоъгълника с долен ляв ъгъл (x1,y1) и горен десен ъгъл(x2,y2).

Тривиален алгоритъм може да се осъществи със сложност O(X*N*N+Y), където Y е броя на добавянията, а X броят на на питанията. За щастие този подход е твърде бавен и затова ще покажем подход със сложност O((X+Y)*LogN*LogN), като ще използваме **индексно дърво**, на което всяка клетка е **индексно дърво**(не е толкова сложно колкото звучи - реализира се с двумерен масив). Всяко малко **индексно дърво**, отговаря за определен ред или интервал от редове, а голямото за цялата таблица.

За да добавим стойност с в клетка с кординати (x,y), започваме от голямото **индексно дърво**(всеки елемнт, на който е обикновенно **индексно дърво**). Започваме от листото, което отговаря за ред x и в него добавяме стойноста с в индекс y(по-горе описания начин). След това се качваме в бащата на листото и добавяме в неговото **индексно дърво** стойност с в индекс y. След това поватряме същата стъпка за бащата на бащата на листото отговарящо за x и така докато стигнем връх 1. Тъй като в LogN дървета обхождаме LogN клетки сложността на добавянето на стойност в клетка е O(LogN*LogN), където N е ширината(височината) на таблицата.

Ще означим с g(x,y) сумата от всички стойности в клетки с кординати i,j, такива че i<=x и j<=y. За да намерим g(x,y), започваме да разглеждаме голямото идексно **дърво**(това чийто клетки са обикновени индексни дървета). Започваме от листото(обикновеното идексно

дърво), което тоговаря за индекс x. Ако това листо е десен наследник на баща си добавяме f(y) от левия наследник. След това повтаряме същото за бащата на листото отговарящо за x и така докато стигнем връх 1. Сложността на намирането на g(x,y) е O(LogN*LogN).

За да намерим сумата от стойностите в правоъгълника (x1,y1,x2,y2) достатъчно е да намерим g(x2,y2) - g(x1,y2) - g(x2,y1) + g(x1,y1).

Задача Traversal

Задачата е от BOI 2003 и също съдържа приложение на индексни дървета, което е много поучително. Условието на кратко е: Дадена е редица от елементи с дадена височина с най-много N(N<=100 000) елемента. Да се намери броя на подредиците на дадената(включваща някои от елементите в същия ред), чийто съседни елементи се различават най-много с H(H<=100 000 000). За улеснение и стестяване на имплементирането на дълги числа резултата трябва да се изкара под модул 9901.

Лесно се вижда решение със сложност O(N*N). Ако d(i) е броя на начините да конструираме редицата с първите i елемнта. То d(i) = sum(d(j)+1), където j<i и разликата между елемент с номер i и елемент с номер j е по-малка от H. Алгоритъма е доста лесен за измисляне и реализация, но е твърде бавен за конкретната задача.

С помоща на **индексно дърво** ще сведем сложността до O(N*LogN), като оптимизираме намирането на стойността d(i). Тъй като височината на кутиите е доста голямо число, а имаме най-много 100 000 кутии, можем да си сортираме кутиите по-височина и да боравим с индексите им в сортирания масив. Нека индекса на кутия с номер i в оригиналната подредба е Pi в сортирания масив.

С помоща на две двоични търсения във всяка кутия или общо линейно за всички кутии можем да сметнем номерата Li и Ri(съответно номерата на най-ниската и най-високата кутия, до която можем да стигнем от кутия с номер i).

След като сме сметнали тази информация можем да обходим всички кутии в оригиналния ред. Смятаме d(i), като f(Ri) - f(Li)(дървото на този етап сме добави елементи с номер j<i). След което добавяме d(i) на позиция Pi.

Всяка кутия се добавя точно един път, и два пъти се вика фукнция f(). Което прави алгоритъма с обща времева сложност O(N*LogN).

Задача Mars

Четвъртата задача е Mars от Baltic Olympiad in Informatics 2001. Накратко дадени са N(N<=30 000) правоъгълници чрез кординатите на най-долната лява точка(x1,y1) и най-горната дясна(x2,y2), търси се площа, която заемат сумарно всички правоъгълници(някои от тях може да се припокриват). Въпреки, че задачата изглежда геометрична, знанията, които са нужни за решаването й не са свързани с аналитичната геомтерия.

Единственото достатъчно бързо решение, което е известно на автора на статията е чрез изполвано на по-специално индескно **дърво** боравещо с интервали. Този тип индексни дървета е известен като Interval tree или Марсиански индексни дървета(това е много разговорно наименование и идва поради известността на задачата).

Разделяме всеки правоъгълник на две вертикални отсечки с кординати (x1,y1)-(x1,y2) (x2,y1)-(x2-y2)(лявата и дясната му страна), като една е разглеждаме като "започваща", а другата като "завършваща" правоъгълника. Сега сортираме всички отсечки по x кордината. Обикаляме всички отсечки, ако отсечката е "започваща" добавяме интервала [y1,y2] в дървото, ако е "завършваща" го изваждаме. Така площа на покритата площ между всеки две отсечки е равна на дължината на покрития интервал(може да е сума от няколко отделни интервала) в дървото умножена по ширината(разликата в x кординатите) между двете отсечки.

Въпроса е как достатъчно бързо да добавяме и изваждаме интервали от индексното **дърво**. За целта нека си означим с left(i) и right(i), двата края на интервала, за който отговаря връх i, с Ci броя на пътите, който е покрит целия интервал [left(i),right(i)], а с Si сумата от дължината на интервалите покрити от децата на i(2*i и 2*i+1). Така, ако целия интервал на върха е покрит поне веднъж(т.е Ci>=1) дължината на покрития интервал за връх i е равна на right(i)-left(i)+1. В противен случай е равен на Si.

Сега нека да видим как да добавим интервал [L,R] във връх с номер i, като знам, че L<=left(i) и R>=right(i). Ако L=left(i) и R=right(i), то значи целия интервал е покрит и трябва да увеличим Ci с единица(това винаги ще е изпълнено ако върхът е листо, където left(i)=right(i)). В противен случай имаме три възможности:

- Първо, интервалът [L,R] да е изцяло в интервал [left(2*i),right(2*i)], тогава добавяме интервал [L,R] във връх (2*i).
- Второ, интервалът [L,R] да е изцяло в интервал [left(2*i+1),right(2*i+1)], тогава добавяме интервал [L,R] във връх (2*i+1).
- Трето, разделяме интервала [L,R] на две части. Добавяме интервал [L,right(2*i)] във връх (2*i) и интервал [left(2*i+1),R] във връх с номер (2*i+1).

В който и от трите случая да попаднем не трябва да забравяме да променим Si, след добавянето на интервали в децата(иначе ще си нарушим правилата в дървото).

Добавянето на интервал в цялото **дърво** е еквивалентно на добавяне на интервал във връх 1. Ще оставя на читателя като лесно упражнение сам да докаже, че сложността на това добавяне също е O(LogN)(Джокер: само двата края на интервала стигат до долу на дървото). Извличането на информация става за O(1), защото сумарно покрития интервал е S1, ако C1=0 и right(1)-left(1)+1 в противен случай.

В крайна сметка получаваме алгоритъм с времева сложност O(N*LogN), което е достатъчно бързо за задачата.

Ресурси

- Тренировки на националния отбор 2003-2005г. с огромната подкрепа на Велин Цанов.
- Условие на Circle
- Условие на Mobiles
- Условие на Traversal
- Условие на Mars Maps