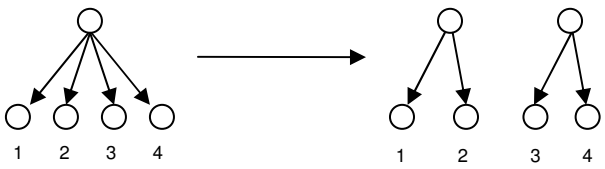
Анализ на Задача 5 СПИСЪЦИ

За разлика от предишните кръгове в пети кръг на конкурса предложихме задача, която има детерминистични решения, които предполагат изобретателност и алгоритмични знания у участниците особено в областта на структурите от данни.
За решението на поставената задача може да се използва структура наречена 2-3 дърво. С нейна помощ се изпълняват коректно зададените операции като операциите merge и split се изпълняват с логаритмична сложност, а останалите отнемат константно време.
2-3 дървото представлява дърво, в което всеки връх освен листата има два или три наследника. Освен това всички листа са на едно ниво. Наследниците на един връх се определят като ляв и десен, а ако са три на брой има и среден. За всеки един списък се използва едно 2-3 дърво. Елементите му се подреждат в листата. Първият елемент се поставя в най-лявото листо, а последния в най-дясното. При създаването на списък с един елемент се създава дърво с един връх, който съдържа елемента.
При обединяването на две дървета трябва листата в новото дърво да са на едно ниво и всеки връх да има два или три наследника. Ако двете дървета имат еднакви височини се създава един нов връх и корените на двете дървета се поставят като негови наследници. По-интересен е случая, когато височините на двете дървета са различни. Тогава по-ниското се "долепя" към по-високото. В новото дърво листата трябва да са подредени така, че елементите в тях да отговарят на подредбата в новия списък. Нека по-ниското дърво има височина h, а по-високото H. Корена на дървото е на дълбочина 1. Ако по-малкия списък се долепя отдясно, се започва от корена и се върви по десните наследници докато се стигне до височина $H - h$. Върха, в който е завършило "спускането" ще бъде родител на корена на по-малкото дърво. Ако по-малкото дърво се долепя отляво се прави същото, но се върви по левите наследници. Когато един връх се добавя като наследник на някой друг може да се получи така, че родителя да има четири наследника. Това вече нарушава условието един връх да има два или три наследника. Ето защо в този случай родителят се разделя на два върха като всеки един има по два наследника. Разбира се те трябва да са подредени в такъв ред в какъвто са били преди това (фиг. 1). Нека върха, който се разделя на два нови е V. Тогава двата нови се прикрепят като наследници на родителя на върха V. По този начин той вече може да има четири наследника. За него трябва да се направят същите операции като за върха V. Това се прави докато е нарушено условието всеки връх да има два или три наследника. Възможно е корена да бъде разделен на два нови върха. Тогава се прави един нов връх, който става корен на дървото, а двата върха, на които се разделя корена стават негови наследници. "Изкачването" по дървото се прави с не повече от $H - h$ стъпки. Височината на дървото е число пропорционално на логаритъм от броя върхове. Ето защо операцията merge се изпълнява със сложност $O(logN)$, където N е броя върхове в дървото. Следва да се отбележи, че сложността е пропорционална на разликата във височините на двете дървета, която е H - h. Това е така, защото слизането и изкачването по по-голямото дърво се прави само по върховете с дълбочина от 1 до H - h. Това ще се окаже полезно при операцията split.

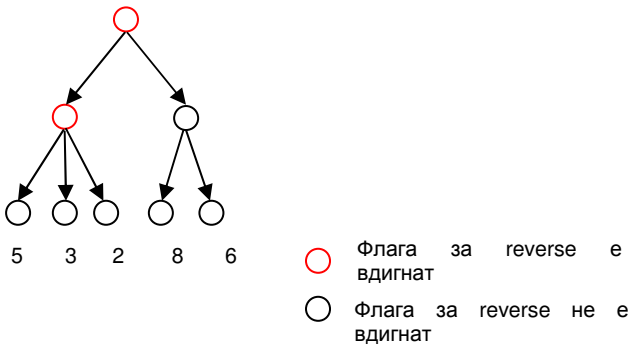
Случай с 4 наследника



Операцията split се свежда до множество операции merge. Нека е нужно да се раздели един списък на два нови $l1$ и $l2$ като първите $p - 1$ елемента трябва да са в първия, а елементите от p до последния са във втория списък. Започва се обхождане от корена като целта е да се достигне елемента, който ще е последен в списъка $l1$. За целта във всеки връх се пази информация колко листа има в поддървото, чийто корен е този връх. Така лесно може да се определи в кой от наследниците на един връх трябва да се продължи, за да се достигне търсеното листо. По този начин се получава път от върховете, през които е преминато. Във всеки връх от пътя без последния се избира наследник, към който да се продължи. Тези наследници, които са наляво заедно с техните поддървета трябва да останат в първия списък. Наследниците надясно ще са във второто дърво. Ето защо когато се прави обхождането от корена към листата, наследниците вляво се поставят на върха на един стек в ред отляво

надясно, а наследниците отдясно се поставят в друг стек, но в ред отдясно наляво. Накрая и листото, до което е достигнато се поставя в първия стек. Когато обхождането е направено елементите от двата стека се обединяват в две нови дървета като се използва операцията merge. От първият стек се изваждат двата елемента от върха и този, който е по-горен се долепя отдясно на този, който е бил под него. След обединението полученото дърво се поставя на върха на стека. За втория стек се прави подобно нещо. Разликата е, че когато се извадят двата елемента на върха, този, който е бил отгоре се долепя отляво на този под него. След това се прави същото както в първия стек. Накрая в двата стека остава по един елемент. Тези два елемента са двете дървета, които се получават след разделянето. На пръв поглед изглежда, че сложността на сливането не е логаритмична. Сложността на операцията merge обаче е пропорционална на разликата във височините на двете дървета, които се обединяват. Ето защо като се обединят всички дървета от двата стека се извършват операции, чийто брой е пропорционален на *H*, където *H* е височината на дървото.
За да се поддържат операциите min и max за всеки връх се пази най-малката и най-голямата стойност сред листата в това поддърво. Тези стойности се обновяват по време на операциите merge и split, за да остане коректна информацията във върховете. По този начин се постига константна сложност за отговаряне на операциите min и max.
Това, което прави задачата по-трудна е наличието на операция reverse. За да може да работи структурата и с тази операция се поддържа един флаг във всеки връх на дървото. Той показва дали листата на поддървото в този връх са в реда, в който са подредени или са на обратно. Когато е подадена команда reverse за едно дърво просто се сменя значението на флага в корена му. Може да се получи така, че в едно дърво флага за обръщане на листата да е вдигнат за някои от върховете различни от корена. За да се пресметне дали реално поддървото чийто корен е даден връх е с обърнати листа е нужно да се направи изключващо "или" на всички флагове във върховете по пътя нагоре от този връх до корена. Не е достатъчно просто да се провери какъв е флага в даден връх, а да се вземат предвид и тези над него. Тази особеност на дървото трябва да се има предвид при изпълнението на операциите merge и split. Например наследниците на даден връх може да трябва да се разгледат отдясно наляво, за да се обходят в ред, който всъщност ги дава в ред отляво надясно, защото дървото е обърнато. Като се използват тези reverse флагове се поддържа операцията reverse заедно с останалите операции. Например списъка, който се изобразява от дървото на фигура 2 е (6, 8, 5, 3, 2).

Примерно дърво



Задачата може да бъде решена и с други подходи. Част от тях са описани от някои участници в кръга. Техните анализи може да намерите на сайта на конкурса.