# ძიუდოს შეჯიბრის მონაწილეთა შეწყვილების ალგორითმი

ძიუდოს შეჯიბრის აპლიკაციას ესაჭიროება შეჯიბრის მონაწილეების შეწყვილება მათი ძალების მიხედვით. შეწყვილების ისეთი სქემაა საჭირო რომლითაც შეჯიბრებაში მონაწილეთაგან უძლიერესები ტურნირში ერთმანეთს შეხვდებიან რაც შესაძლებელია გვიან, ტურნირის დასასრულში.

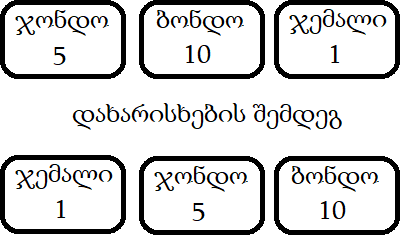
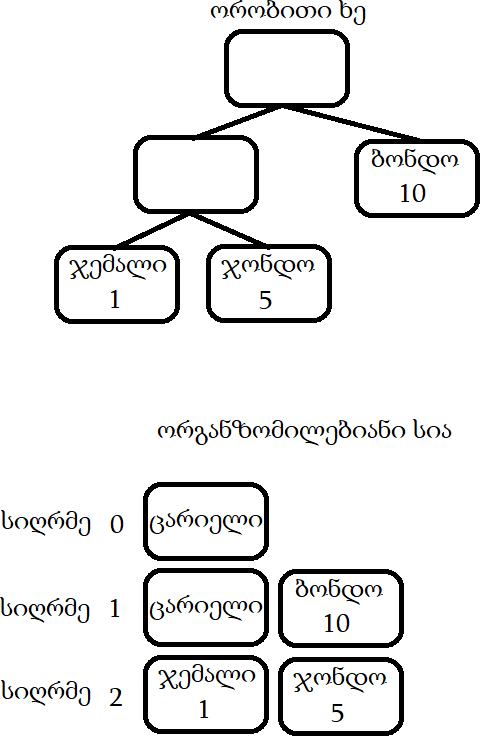
ამოცანის დასმა - შემოდის მონაცემები ტურნირის მონაწილეთა შესახებ. საჭიროა მათი სიძლიერის (შედარებადი ველი) მიხედვით დახარისხება/შეწყვილება და ისეთი სახით დაბრუნება რომ გარკვევადი იყოს თუ ვინ ვის ეჯიბრება და რა ეტაპზეა ტურნირის დაწყებამდე:

* ნორმალური შემთხვევა - როდესაც მონაწილე კარგად ზის ტურნირის შეხვედრათა ცხრილში.
* დამატებითი შემთხვევა - შეჯიბრთა ორობითობისა და მონაწილეთა რაოდენობიდან გამომდინარე შესაძლოა ზოგიერთ მონაწილეს ე.წ. დამატებითი შეხვედრების ჩატარება.

გამოყენებული ტექნიკა - ალგორითმი მონაწილეთა შეწყვილებისათვის და  
მონაცემთა ადვილი აღმისათვის იყენებს ორობითი ხის სისტემას. იგი დაწერილია C# და იყენებს .Net-ის სტანდარტულ ბიბლიოთეკებს მონაცემთა სიაში შენახვისა და დასორტირებისთვის.

ორობითი ხე - ორობითი ხე არის მონაცემთა სტრუქტურა რომლის მონაცემები მოთავსებულია კვანძებში. ყოველი კვანძი შეიცავს ინფორმაციას ხეში საკუთარი სიღრმის, მშობელი კვანძის, მარცხენა და მარჯვენა შვილი კვანძების შესახებ. ყოველ ხეს გააჩნია ე.წ. ფესვი - ეს არის კვანძი რომელსაც არ ჰყავს მშობელი კვანძი, მისი პირობითი სიღრმეა 0 და ყოველი სხვა კვანძი ამ ხისა არის მისი შვილი და შვილთა შვილი სხვადასხვა სიღრმეში. ფოთოლი - არის კვანძი რომელსაც არ ჰყავს არც მარჯვენა და არც მარცხენა შვილი კვანძი.

მუშაობის ეტაპები - ალგორითმის მუშაობა პირობით სამ ნაწილად შეიძლება:

* მონაცემთა მიღება, სიაში ჩაწერა და დახარისხება. დახარისხება ხდება მონაწილეთა სიძლიერის მიხედვით. დახარისხებისთვის გამოიყენება .Net ის სტანდარტულ ბიბლიოთეკაში quicksort-ის იმპლემენტატორი რომელიც ახარისხებს List ტიპის კოლექციას და მუშაობისთვის ითხოვს რომ კოლექციაში მყოფი ობიექტები იყვნენ System.IComparable ინტერფეისის მაინმპლიმენტერები და გადატვირთავდნენ ნახსენები ინტერფეისის CompareTo(object o) ფუნქციას.  
    
  
* დახარისხებული მონაწილეთა კოლექციის გამოყენებით მონაწილეთა შეხვედრების ამსახველი ორობითი ხის გენერირება. უკეთ ვისაუბრებთ მოგვიანებით.
* შეწყვილების შედეგად მიღებული ხის უკეთ აღსაქმელი ფორმით გამოტანა. ორობითი ხე არ არის ინფორმაციის მარტივად გასაგები ფორმა, საჭიროა მთლიანი ხის გარკვეული ხით ჩამოვლა და მონაცემთა გადატანა.  
  დაწერილია ფუნქცია რომელიც იღებს ხის ფესვს და აბრუნებს ორგანზომილებიან შეჯიბრის სიას: პირველი განზომილება აღნიშნავს შეჯიბრის სიღრმეს ( ფინალიდან შერჩევითებამდე ზრდადობით ), მეორე განზომილებაში მოქცეულია მონაცემები მოცემულ სიღრმეში მონაწილეთა წყვილების შესახებ ( დაწყებული ნულიდან ზრდადობით, მიმდევრობით ).  
  ტურნირი იწყება ყველზე ღრმა სიიდან მონაწილეთა წყვილების არჩევით და შემდეგი სიმაღლის ცარიელ უჯრებში გამარჯვებულის გადაყვანით.  
  

**მონაწილეთა შეწყვილება**

თავდაპირველად საჭიროა მონაწილეთა რაოდენობის მიხედვით გამოთვლილი რიცხვის შესაბამისი მქონე ცარიელი ორობითი ხის ინიციალიზება.  
მთავარი ხის სიღრმე გამოითვლება მონაწილეთა რაოდენობიდან ორობითი ფუძის ლოგარითმის ამოღებით და მთელობამდე ნაკლებობით დამრგვალებით:

Depth = floor( log( n ));

შემდგომ თვით შეწყვილებას სანამ დავიწყებთ საჭიროა კიდევ ერთი მონაცემის, დამატებითი შეხვედრების რაოდენობის განსაზღვრა. იძებნება უახლოესი, მცირე, 2 ის ხარისხი რიცხვი და იგი აკლდება მონაწილეთა რაოდენობას. ორობით ხეში ყოველი დამატებითი შეხვედრა არა ერთი, არამედ ორი დამატებითი კვანძის გაჩენას იწვევს, შედეგად საჭირო დამატებით კვანძთა რაოდენობა ორმაგდება.

Additional = 2 \* (n – 2floor( log(n ));

ახლა ჩვენ უკვე შეგვიძლია დავიწყოთ მონაწილეთა სიის დამუშავება მათი შეწყვილების მიზნით. ზევით ინიციალიზებული ცარიელი ხე, მონაწილეთა სია და დამატებით კვანძთა რაოდენობა გადაეცემა ფუნციას რომელიც გარკვეული სახით ორად ხლეჩს სიას და რეკურსიულად იძახებს თავის თავს მარცხენა და მარჯვენა შვილებისთვის ყოველი ახლადშექმნილი სიით. რეკურსიული გამოძახება მოხდება მანამ, სანამ გადმოცემული სია არ ჩამოვა ერთ ელემენტამდე, ანუ მივაღწევთ ხის ფოთლამდე, ამ შემთხვევაში ხდება მონაწილის მონაცემის შენახვა კვანძში და ფუნქციის დასრულება. განსაკუთრებული ყურადღება სჭირდება დამატებითი კვანძების შექმნას, ყოველთვის როდესაც ვცდილობთ გადავცეთ მონაწილეთა სია შვილებს, მაგრამ აღმოჩნდება რომ ასეთი შვილი კვანძი არ იყო განსაზღვრული საწყისად ინიციალიზებულ ხეში, ეს ნიშნავს რომ ასეთი კვანძი არის დამატებითი შეჯიბრისათვის, შესაძლებლობისდაგვარად ემატება ახალი კვანძი და რიცხვითი ცვლადი მცირდება ერთით. სიის გახლეჩა - ყოველ ბიჯზე შემომავალი სია უნდა იყოს დახარისხებული (ზრდადობით ან კლებადობით); იქმნება ორი დამატებითი, ცარიელი სია მარცხენა და მარჯვენა შვილებისთვის გადასაცემად. სიაზე ხორციელდება ინდექსირებული ჩამოვა ( for ციკლი ), ინდექსატორი მონაცვლეობს 0-დან (საწყისი) სიაში ჩანაწერთა რაოდენობა**მდე** (არჩათვლით) ერთის ბიჯით; შემდეგი მანიპულაცია მარტივია, ყოველი სიის ელემენტის ინდექსი მოწმდება ლუწობა კენტობაზე და შედეგის მიხედვით ემატება სიას მარცხენა ან მარჯვენა შვილისათვის.

სიის დამუშავების ალგორითმის ფსევდოკოდი:

დაამუშავე\_სია( კვანძი, სია, დამატებითი )  
{  
 თუ სია შეიცავს ერთ ელემენტს   
 მაშინ   
 მიანიჭე კვანძს სიის ერთადერთი ელემენტი  
 დაასრულე ფუნქცია  
 შექმენი ცარიელი სია მარცხენასთვის  
 შექმენი ცარიელი სია მარჯვენასთვის  
 ინდექსი ჩამოივლის მთელ რიცხვებს 0 დან სიის ელემენტთა რაოდენობამდე  
 ყოველი ინდექსის მნიშვნელობისთვის  
 თუ ინდექსი ლუწია  
 მაშინ   
 სიის ინდექსის შესაბამისი ელემენტი დაამატე მარცხენასთვის სიას  
 წინააღმდეგ  
 სიის ინდექსის შესაბამისი ელემენტი დაამატე მარჯვენასთვსი სიას  
 თუ საჭიროა კვანძის მარცხენა შვილის დამატება და დამატებითი დადებითია  
 მაშინ  
 დაამატე კვანძის მარცხენა შვილი ცარიელი მონაცემით  
 დამატებითი შეამცირე ერთით.  
 თუ საჭიროა კვანძის მაჯვენა შვილის დამატება და დამატებითი დადებითია  
 მაშინ  
 დაამატე კვანძის მარჯვენა შვილი ცარიელი მონაცემით  
 დამატებითი შეამცირე ერთით.  
 დაამუშავე\_სია(კვანძის მარცხენა შვილი, მარცხენასთვის, დამატებითი)  
 დაამუშავე\_სია(კვანძის მარჯვენა შვილი, მარჯვენასთვის, დამატებითი)  
}

აღწერილი ალგორითმი თავისი სიმარტივის მიუხედავად მშვენივრად ასრულებს საკუთარ დანიშნულებას და მონაწილეები გადანაწილდებიან ხეზე იმგვარად, რომ ყველზე ძლიერთა შეხვედრა მოხდება რაც შესაძლებელია გვიან; და ეს არ ეხება მხოლოდ 2 უძლიერეს მონაწილეს, ყოველი წყვილისთვის უძლიერესებისა, მათი შეხვედრა იქნება შეძლებისდაგვარად უგვიანესი!  
