

Създаване на Web-базирана система за подготовка на ученици за състезания по информатика

Валентин Михов

Август 2010

Абстракт

В настоящата дипломна работа ще опиша опита ми в създаването на Web система за обучение на ученици по информатика. Ще опиша как за кратко време успях да изградя ядрото на системата, за имплементацията на което обикновено отнема много време, като така успях да се концентрирам върху важните части на системата, които се ползват от учениците и ръководителите. Освен това ще споделя опита, който натрупахме заедно с още няколко колеги, докато използвахме тази система. Какво направихме добре и какво видяхме, че не работи при подготовката на състезатели по информатика.

Съдържание

1	Въведение	5
1.1	Състезателните системи.	5
1.2	Идеята за Маусамр	6
1.2.1	Защо още една система	7
2	Изграждане на системата	8
2.1	Изграждане на ядрото за пускане на код	9
2.1.1	Разработване на модула за контролирано пускане на код	10
2.2	Архитектура на системата	12
2.3	Изграждане потребителският интерфейс	14
2.4	Формат на състезанията и задачите	15
2.5	Описание на системата за рейтинги	16
2.5.1	Математически модел	16
2.5.2	Примери	21
2.5.3	Прибавяне на външни състезания	22
2.5.4	Национална ранглиста на състезателите	22
3	Анализ на постигнатото до момента	23
3.1	Статистика на съдържанието на системата	23
3.2	Статистика за посещаемостта на системата	25
3.3	Статистика за популярността на състезанията	27
3.4	Скалируемост	27
4	Примери и описание на работата със системата	27
4.1	Част за ученици	27
4.1.1	Начална страница	27
4.1.2	Регистрация	29
4.1.3	Вход	30
4.1.4	Участие на състезание	32
4.1.5	Резултати от състезание	34
4.1.6	Практика	35
4.2	Част за учители	35
4.2.1	Прибавяне на акаунт на ръководител	36
4.2.2	Създаване на състезание	37
4.2.3	Прибавяне на задачи и тестове	38
4.2.4	Прибавяне на резултати от външно състезание	39
4.2.5	Тестване на решения	40
4.2.6	Доклади	42

4.2.7	Пращане на съобщения	43
5	Идеи за бъдещето	44

Списък на фигурите

1	Архитектура на системата	12
2	Нормално разпределение	17
3	Функция на грешката	19
4	Статистика за посещаемостта от Ноември 2009 до Май 2011	25
5	Анализ на посещаемостта от 1 Декември 2009 до 31 Март 2010	25
6	Съпоставка на посещаемостта - 1 Януари до 30 Април 2010 и 2011	26
7	Брой на потребителите пратили поне 1 решение разбито по месеци	26
8	Маусамп Arena - Начална страница	27
9	Маусамп Arena - Регистрация	29
10	Маусамп Arena - Вход	30
11	Маусамп Arena - Начална страница на влязъл потребител	31
12	Маусамп Arena - Начална страница със текущо провеждащо се състезание	32
13	Маусамп Arena - Страница на текущо провеждащо се състезание	33
14	Маусамп Arena - Резултати от състезание	34
15	Маусамп Arena - Практикуване на състезание	35
16	Маусамп Arena - Администриране на потребител	36
17	Маусамп Arena - Създаване на състезание	37
18	Маусамп Arena - Създаване на задача	38
19	Маусамп Arena - Класиране от външно състезание	39
20	Маусамп Arena - Разглеждане на пратено решение	41
21	Маусамп Arena - Доклади	42
22	Маусамп Arena - Пращане на съобщения	43

1 Въведение

Това не е първата или последната разработка на автоматизирана система за оценка на задачи по информатика. Има много разработки по темата и не малко сайтове, които предлагат изключително добре направени системи, със изключително голямо количество задачи в тях. Така например Sphere Online Judge съдържа над 5000 задачи и 40 различни езика за програмиране [7]. Въпросът е защо решихме да направим още една такава и кому това е нужно? В тази глава ще разкажа, какво всъщност е състезателна система и как възникна идеята за Маусамр.

1.1 Състезателните системи.

Подробно описание на състезателните системи (СС) - предназначение, архитектура, основни функции и т.н., може да бъде намерено в [1], затова тук ще се спрем съвсем накратко на техните характеристики. Всяка СС (на английски Grading system или прсто Grader), по своята същност представлява софтуерна система, която предоставя на участниците в състезания по програмиране възможност да изпращат решенията на състезателните задачи (изходен код на програма, написан на един от езиците за програмиране поддържани от системата). Системата архивира изпратеното решение, компилира го до изпълним код и проверява дали получената програма решава поставената задача, като изпълнява програмата върху зададено множество от тестове, в рамките на поставени ограничения за времето и памет, и сравнява получените резултати с очакваните.

Това разбира се е доста опростено описание на същността на СС, тъй като в повечето случаи една СС трябва да предоставя възможност за организиране на състезания по програмиране, за съхранение и визуализиране на условията на задачите, за генериране на класиране, поддръжка на различни видове задачи и оценявания и т.н. СС са изключително полезни при организирането на състезания по програмиране, тъй като автоматизират напълно тестването на решенията. Без използването на СС, тестването на програмите на състезателите се извършваше ръчно и отнемаше много време. Освен това СС може да автоматизира и други дейности, като генериране на класирания, статистики и архиви на състезанията. Една от първите състезателни системи е системата PC^2 , днес официална състезателна система на ACM ICPC – Международната олимпиада по програмиране за студенти [2]. От 2000 година Международните олимпиади по информатика за ученици също използват състезателни системи, създавани обикновено от страната домакин. В органи-

зираните в България състезания по програмиране – както национални, така и международни – се използват системите SMOС [3] и spoj0 [4].

Практиката отдавна е показала, че СС са незаменим помощник за организаторите на състезания по програмиране. Интересното е, че не е толкова очевидно как една такава система може да помогне в обучението на ученици и студенти по програмиране. Има примери на СС, които се използват в процеса на обучение по програмиране. В [5] авторите описват опита си в прилагане на СС в обучението по програмиране в Португалия. Системата spoj0 се използва активно във Факултета по математика и информатика на Софийския университет при преподаване на курсове по алгоритми. Елементи на обучение предлага, например, системата USACO за подготовка на американските ученици, които се готвят за участие в Международната олимпиада по програмиране [6].

За съжаление, споменатите системи не са много популярни сред учениците в България, особено сред по-малките. Това се дължи до голяма степен на факта, че тези системи са “затворени” в рамките на съответните институции или за използването им е необходимо добро владение на чужд език (най-често английски), което прави използването им от по-малки ученици трудно или невъзможно.

Разглежданият в тази статия проект има за цел да подпомогне подготовката на начинаещи програмисти с използване на състезателна система. В раздел 2 е представена една ранна фаза на проекта и някои негови недостатъци. В раздел 3 са формулирани проектните идеи, които трябваше да ни приближат по близко да целта на проекта. В раздел 4 са представени някои резултати от едногодишната работа на сайта, а в раздел 5 - насоки за бъдещи усъвършенствания.

1.2 Идеята за Маусамр

Нека първо да разясним кой сме ние и как започна всичко. Всичко започна в един обед в София, когато бяхме заедно със моя колега Слави Маринов в един ресторант. Разговаряхме за това как образованието в България е доста струпено и има нужда от нещо, което да го оправи. Слави имаше идеята да направим промяна в подготовката по информатика: нещо в което имаме познания и опит. Останалото ще дойде с времето. Това беше началото. Една проста идея: да подобрим подготовката по информатика в България.

След известно време вече бяхме събрали доста съмишленици, най-вече сред бившите състезатели по информатика, които са останали в България. Събрахме се не малко хора: Антон Димитров, Тодор Петров, Орлин Тенчев, Слави Маринов, Даниел Славов, Кремена Роячка, Слави

Маринов, Пламена Александрова и аз.

Първоначалната идея беше да се направи сайт, на който да се публикуват видео лекции. Лекциите са организирани на нива и за да можеш да преминеш на следващото ниво трябва да решиш задачите от текущата лекция, като така показваш, че си разбрал материала. Освен това всеки участник има ментор, който му помага когато има трудности.

След едногодишна работа се оказва, че този формат не е толкова ефективен, колкото на нас ни се иска и е труден за поддръжка от хора, които се занимават само през свободното си време с това. Всички хора в проекта работихме на пълен работен ден и трябваше да вменяваме работата по проекта в графика си. Така на годишната сбирка на екипа, след като разгледахме резултатите от анкетата, която проведохме сред учасниците, решихме да сменим формата на подготовката. Почти всички участници бяха посочили, че ако правим тренировъчни състезания това ще подобри новото им много повече. Така решихме да направим система, на която да даваме състезания.

Системата не беше единственото нещо, което решихме да създадем. Освен редовните състезания и възможността за практика на стари задачи, решихме на сайта да има и анализи на решенията, както и статии на различни теми, като например стратегии за решаване на задачи на състезание.

Беше решено, че ще направим 12 състезания през годината, като всяко състезание ще има 3 дивизии: бронзова, сребърна и златна, като разликата ще е в нивото на трудност. Идея, която взимашме от USACO [8]. След всяко състезание ще се публикуват анализи на задачите и всяка задача ще може да се практикува, като ще има класация по практика. Освен това задачите ще се оценяват на принципа на ученическите състезания, като на една задача можеш да имаш между 0 и 100 точки според това за какъв процент от тестовите програмата ти работи.

1.2.1 Защо още една система

Един от най-важните въпроси и е защо решихме да правим изцяло нова система, след като има толкова много готови. Разбира се ние мислихме по този въпрос и като крайно прагматични хора разгледахме алтернативите. Ето и причините, които ни накараха да поемем по този път:

1. Искане всички материали да са на български език - след като направихме проучване сред състезателите, се оказа че по-малките състезатели имат проблем с подготовката от сайтове на английски език. Тъй като според нас малките състезатели са много важни

(дори може би по-важни от големите), решихме че това е много важен аргумент, който трябва да имем предвид.

2. Искаме задачите да се оценяват на принципа на ученическите състезания. Оказва се че повечето системи за подготовка се основават на АСМ правилата, които гласят че една задача е или решена за 100 точки или не е решена, т.е. 0 точки. Това не беше допустимо за нас, тъй като на едно ученическо състезание е от много голямо значение всеки състезател да може да извлече всяка една точка. Това, че не може да измисли решението за 100 точки не значи, че трябва да се откаже от дадена задача. Това според нас е умение, което трябва да се тренира в участниците.
3. Искаме да даваме лесни задачи в бронзовата дивизия. Според нас това е жизнено важно за привличането на нови състезатели в малките възрастови групи. Повечето системи за подготовка наблягат на сериозните състезания, което кара малките и нови състезатели да се отчайват от трудността на задачите. Нашата цел е точно обратна - да накраме тези по-малки и неопитни деца да усетят тръпката на състезанието и победата над съучениците им.
4. Възможността ние да контролираме всеки един аспект от системата. Свободата, която ще имаме ако системата е наша е много важна за да можем да вървим напред. Разбира се това идва на цена: повече време отделено за разработка и нуждата да поддържаме сами всичко.

Разбира се ние сме наясно, че няма вечни неща на този свят и е възможно в близко бъдеще да се появи нова система правена от други хора, която да засенчи нашата. Затова и целта беше да изградим това начинание с възможно най-малко усилия, без да се задълбаваме в технически подробности. В следващите глави ще опиша какви решения направихме по време на имплементацията и как всъщност с доста малко усилия изградихме всичко.

2 Изграждане на системата

В следващата секция ще проследя етапите през, които премина изграждането на Арената. Ще се опитам да опиша какво беше направено добре и какво не.

2.1 Изграждане на ядрото за пускане на код

Във всяка една система за провеждане на системни тестове по програмиране има ядро, което се занимава със пускане на решенията на участниците в контролирана среда и оценка на решенията срещу дадено множество от тестове.

Обикновено ядрото на системата включва в себе си модул, който умее да пуска код в контролирана среда - така нареченият *sandbox*. Идеята на този модул, е че кода който състезателите пращат обикновено не е коректен и пускането му може да доведе до нестабилност във системата. Освен това е нужен за да се предотврати опити за хакване на системата или за измами, като се пращат решения, които по заобиколен и нечестен начин успяват да работят за 100 точки.

Има много публикации за това как може да бъде направен един подобен модул. Това е доста сложна тема и един такъв модул може да е изключително сложен. Може би това е най-трудната част в имплементирането на една подобна система.

Единият от начините е да се направи модул към ядрото на Linux, с който да се следи какво прави изпълняваният код. За повече информация можете да видите [10], където е анализирано подобно решение. Оказва се обаче, че подобна имплементация има доста проблеми, като например често сменящи се интерфейси за модулите за сигурност на Linux, не добра документация и като цяло не добра поддръжка на този вид модули от екипа разработващ ядрото на Linux. Освен това е добре да се отбележи, че подобен модул няма да е никак тривиален за имплементация, тъй като изисква дълбоки познания за ядрото в Linux и работа с недокументирани интерфейси.

Другият начин за справяне с този проблем е с прихващане на системните извиквания. Подобно решение е ползвано и в системата Мое и повече за него може да бъде прочетено тук [11]. Накратко идеята е да се ползва така нареченият *ptrace* интерфейс в Linux, с който кода на състезателя за пуска в контролирана среда и всяко едно системно извикване от кода се прихваща и проверява от контролиращият модул. Ако контролираният код направи системно извикване, което не е позволено, то той се прекратява и се смята, че участника се опитва да направи нещо против правилата.

Това е може би най-широко използваният в момента модел, като системите на IOI използват точно него. За нещастие и той има някои слаби страни: може да забави доста тестването на решенията [12], а и не е никак тривиален за имплементация.

Трябва да се отбележи, че подобни модули вече съществуват и дори

могат да се ползват на готово, както например беше направено на международната олимпиада в България 2009 [13]. Модулът, който се ползва на тази олимпиада е от системата Мое и вече е ползван на няколко големи олимпиади и показва стабилни резултати. За съжаление, когато аз започнах да разработват системата за Маусамр, не знаех за съществуването на този модул, а нямах никакво време за да разработя сложни решения. Така стигнах до решение, което е доста просто като логика, много кратко и след 11000 тествани решения и над 10 проведени състезания е показало незначителни проблеми, за които ще напиша по-нататък.

Решението, което предлагам предоставя защиты срещу няколко основни типове атаки и мисля, че до голяма степен елиминира 95% от възможните хакове, които биха могли да бъдат пробвани.

2.1.1 Разработване на модула за контролирано пускане на код

Когато започнах изграждането на модула, първото нещо което направих беше да идентифицирам основните ситуации, които искам модулт да засича. Това са:

1. Решения, които зациклят в безкраен цикъл
2. Решения, които заспиват в системно извикване (например `getc()`)
3. Решения, които се опитват да заделят повече памет от позволеното
4. Безкрайна рекурсия
5. Fork-бомба
6. Опити за установяване на мрежова връзка

Въз основа на тези основни проблеми, направих проста система за тестване, която симулираше всяка от тези ситуации и проверяваше дали изграденият модул се справяше с тях. Самият модул представлява скрипт, който приема като аргументи параметрите, в които решението трябва да се изпълни и като прикючи, в статуса му има код, който описва дали това се е случило. Ако по време на изпълнението е възникнала извънредна ситуация, статусът описва какво точно е станало. Ето и псевдо-код на скрипта:

```
fork the current process
in the child:
    set limit of the number of child processes
    set the priority of the current process to the lowest
```

```

        execute the program to be tested

in the parent:
    while the child process is running:
        check the running time of the child process
        check the used memory of the child process
        check the running state of the child process

        if any of these are out of expected, kill the process

process the exit status of the child process

```

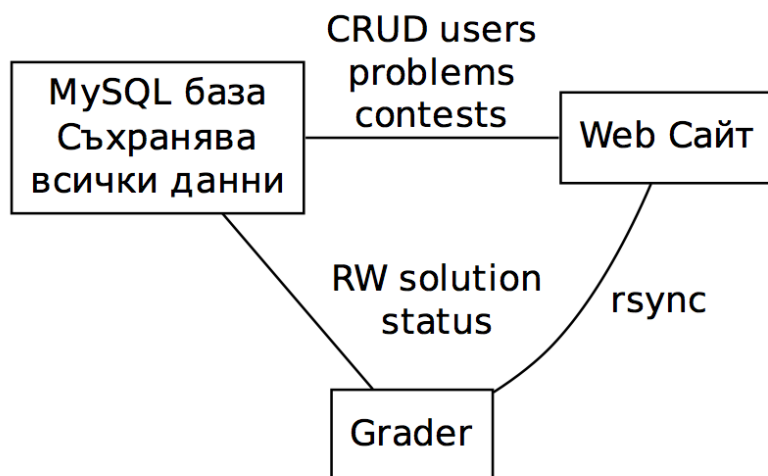
Както виждате първото нещо, което прави той е да клонира текущият процес и да заложил на който от параметрите на новият процес. Задаването на брой позволени процеси става посредством `setrlimit`. След проведени тестове се оказва, че това работи стабилно под Linux и предотвратява атаките със `fork`-бомби, които са може би едни от най-лесните и най-опасните за подобна система.

След това се сваля приоритета на новият процес до минимума, като така се гарантира, че нашият скрипт, който ще следи решението ще се изпълнява достатъчно често за да може своевременно да засече всякакви нередности.

Накрая новият процес пуска кода на програмата за тестване, която напълно заменя новият процес.

През това време в процеса-баща вървят постоянни проверки за състоянието на процеса-дете. На много малки интервали от време се проверява във `proc` файловата система, колко точно време и памет е използвана, както и състоянието на самият процес. Ако по време на една такава проверка се види, че процесът е заспал, то значи той прави нещо нередно. Нито едно от системните извиквания, които са позволени на състезателите не могат да накарат програмата да е в състояние `sleep`. Това означава, че е направено системно извикване, което чака някакви данни да се получат или за някакво друго събитие. Задачите, които се тестват на тази система, винаги четат от стандартният вход и пишат на него и това са единствените системни операции, които им са позволени. Освен това винаги когато едно такова решение бива извикано, входните данни са подадени на стандартният вход, т.е. не се очаква решението да заспи в чакане на входни данни.

Така чрез тези няколко изключително прости механизми се гарантира прихващането на изброените по-горе сценарии.



Фигура 1: Архитектура на системата

2.2 Архитектура на системата

При изграждането на Арената, реших че е крайно неприемливо цялата система да е на една физическа машина. Тъй като пускането на чужд код е една крайно несигурна операция, дори и в контролирана среда, възможността при евентуална атака цялата система да бъде компрометирана ми се струваше крайно неприемлива. Затова и решението беше решенията да се оценяват на машина, която е изцяло независима от системата, която поддържа Web интерфейса към системата. Така ако има атака, чрез прашане на опасен код, сайта ще продължи да работи, а машината, която тества ще бъде афектирана - Фигура 1.

Това е доста прост начин за изолация и не е никак нов при изграждането на подобни системи. Тъй като при нас все още данните са сравнително малко решихме да обединим MySQL машината с Web машината, като така намалим разходите си за инфраструктура. Така цялата система се състои от 2 машини. Едната се хоства в dreamhost.com, сериозен доставчик на подобни услуги, а другата се намира в България, на сървър, които ние закупихме за разнообразни нужди. Така израдената архитектура има своите определени преимущества, но има и някои недостатъци, с които трябваше да се преборим.

Първият проблем, е скоростта на комуникация между отделните машини. В нашият случай Web частта се намира на чуждестранни сървъри, където сме сигурни, че машината ще работи 99% от времето и ще

е свързана постоянно към Интернет. От друга страна машината, която оценява решенията се намира в България. За всяко едно оценяване на решение трябва да се осъществи комуникация между двете машини, като скоростта между тях е многократно по-малка от скоростта ако те бяха в една локална мрежа. Това прави нужно да се минимизира комуникацията между тях. Затова решихме двете машина да обенят единствено сорс кодовете на решенията, резултатите от тестването и логовете от тестването на решението.

Това решение се оказа добро, но малко след пускането на системата се оказа, че има проблем в него. Проблема, беше не толкова в архитектурата, колкото в конкретната имплементация. Оказа се, че ако някое решение генерира много голям вход, лога на тестването става много голям, тъй като той съдържа разликата между правилният отговор и отговора на състезателя. Така ако разликата е много голяма комуникацията между Web частта и Grader-а се задръства. Това наложи да сложим ограничения на размера на показваната разлика в лога, което реши проблема.

Вторият проблем е как да се пазят тестовите на задачите. Те могат да са доста големи и както видяхме в предната точка не е приемливо много данни да се прехвърлят при всяко тестване. Така решихме да пазим две копия на тестовите. Едно на Web сървъра и едно на тестваният сървър. Въпросът е как синхронизираме тези две копия?

Първо за да можем да сме сигурни, че във всеки един момент тестваме с най-новите тестове, преди всяко тестване се проверява поле в базата, което пази последната дата на промяна на тестовите. Ако тази дата е по-нова от датата на тестовите, които има на Grader-а в момента, тестовите се обновяват. За да работи тази схема коректно е нужно двата сървъра да имат синхронизирано време. За целата за използва NTP услугата, която синхронизира часовниците на сървърите с точното време взето от световни сървъри предназначени за това.

Първоначално бях направил синхронизацията да става с изтегляне на zip файл със тестовите, но това доста бързо се оказа неподходящо. В момента тестовите на всички задачи заемат 945Mb некомпресирани, но дори и компресирани това е доста голямо количество данни за пренос между двата сървъра. За целата направих решение използващо rsync, което копира единствено променените файлове между двата сървъра. Това се оказа прекрасно решение! Авторите на задачи могат в рамките на секунди да качат нова задача и да я тестват. Времето за синхронизация на тестовите е на практика незабележима.

2.3 Изграждане потребителският интерфейс

Една от основните цели, които си поставихме при създаването на системата, беше потребителският интерфейс да е лесен за ползване и да имаме възможност да о променяме по лесен начин. Тъй като аз бях човека, който основно разработваше кода, реших да ползваме за изграждането на системата Ruby on Rails [16]. Това мое решение беше продиктувано от опита ми с тази библиотека и увереността, че тя ще ни предостави лесен начин за разработване на потребителският интерфейс, както и лесен начин за неговата промяна с времето. Всъщност една от силните страни на Ruby on Rails е неговата гъвкавост и липса на “поддържащ код”, т.е. код, който не е свързан пряко със основната функционалност на системата.

По време на разработката на потребителският интерфейс се придържахме към итеративен процес. Аз разработвах парче нова функционалност и след това няколко човека започвахме да го ползваме. В процеса на ползване се откриваха проблеми и недостатъци, които се оправяха. Създавах се и интеграционни тестове, които спомагаха да се откриват регресии своевременно. За разработване на тестовете се ползваше cucumber [17]. Това е библиотека за тестване на високо ниво, като в общи линии тестовете симулират човешки действия на сайта, като например: отваряне на дадена страница, следване на линк от текущата страница, попълване на форми и прашенето им и други. Тестването е много съществен елемент в едно подобно гъвкаво разработване на система, тъй като нещата се променят много бързо и е много лесно части от системата да спрат да работят. Подържането на добро множество от тестове, които освен това се пускат автоматично след всяка качена промяна на кода е ключово за това процеса на работа да върви безпрепятствено.

Последният елемент, който допълнително направи разработката много гъвкава беше разработването на система за обновяване на сайта с едно кликване. Това означава, че след като някаква промяна е направена и качена в централното хранилище на кода, за няколко минути се вижда дали няма регресии (функционалност, която е спряла да работи) и след това само с една команда и в рамките на секунди новият код може да се качи на сайта и да бъде достъпен за всички потребители. За разработването на това автоматизирано обновяване ползвахме capistrano [18].

Този процес ни осигури много голяма гъвкавост и изключително кратко време на реакция при откриване на проблеми или нуждата за разработване на нова функционалност. Тези основни елементи са нещо, което мога да препоръчам при разработването на всеки един проект. Възможността да във всеки един момент да се види каква е текущата версия и да може да се качи обновена версия в рамките на секунди са

незаменими възможности при разработката на един софтуер.

2.4 Формат на състезанията и задачите

Формата на състезанията и задачите е много подобен на този на системата `spoj0`. Това е така, тъй като в началото системата `Arena` започна като разклонение на `spoj0`. В последствие кода толкова много се промени, че в момента двете системи почти нямат нищо общо, но формата остана същият, което е добре, тъй като позволява много лесно състезания да се качват от `spoj0` на `Arena`.

Най-общо едно състезание има две части: част която се пази в база данни и част, която се пази на файловата система. На практика всичко освен условието на задачата, тестовете и чекъра се пазят в базата. Това са името на задачите, ограниченията за време и памет, продължителността на състезанието и решенията на участниците. Както може да се забележи това е най-вече мета информация за състезанието, която е нужно да бъде лесно и бързо достъпна за да работи сайта добре. Информацията, която се пази на файловата система се ползва най-вече от `“grader”`-а, който оценява решенията. Това е защото се предполага, че `grader`-а може да е отдалечена машина и неговият достъп до базата данни е по-бавна. Това прави невъзможно да се пазят големи данни в базата, тъй като биха забавили много оценяването.

Както споменахме по-рано тестовете на задачите са доста голям обем от информация, достигат почти 1ГБ и се увеличават стремително. Интересен е проблема как тези данни се дистрибутират на машините, които оценяват решения. Отговора е много прост: `rsync` [19]. Това е инструмент с отворен код, който се ползва широко за синхронизация на масиви от данни между няколко машини. Хубавото на този инструмент, е че пренася само променените файлове, т.е. ако се качат тестове на нова задача, единствено тези нови тестове се синхронизират между системите, които оценяват. Така след като се качи ново състезание в рамките на секунди `grader`-ите получават новите тестове на състезанието и са готови да тестват решения. Аналогично ако тестовете се обновят или дадено състезание се изтрие, отново в рамките на секунди промените се пренасят на всички машини.

Формата на входните и изходните файлове е много прост. Единствените изисквания е файловете да са в една директория и имената им да съответстват на определен формат. Входните файлове трябва да са именувани във формата `име.інномер`, където `име` е обикновено името на задачата, а `номер` е номера на теста. Аналогично изходните файлове са във формата `име.ansномер` или `име.solномер`. Номерата в имената де-

финиран на кой входен файл, кой изходен файл съответства. Този начин да дефиниране е много познат на авторите на задачи и много често на националните състезания формата е точно такъв. Освен това този формат позволява на авторите на задачи да съставят тестовите данни, да ги архивират и качат като архив на Арената. Самата система разархивира файла и слага тестовете на правилните места, след което е вече възможно да се тестват решения. За момента Арената няма функционалност, която да следи различните версии на тестовете, които са качени. Това би била интересна добавка.

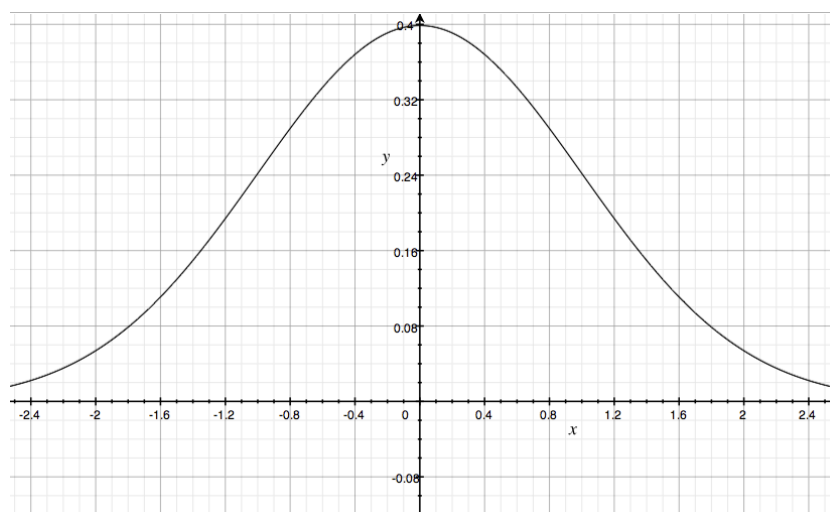
Арената също поддържа задачи, които се проверяват от проверяващи програми. Това са така наречените “чекъри” (checkers). Необходимостта от подобни програми често се налага, тъй като на много задачи е възможно да има повече от едно правилно решение и това прави простото сравнение на изходните файлове навъзможно. Проверяващите програми могат да бъдат написани както на C/C++ така и на скриптов език като например питон или руби. Изискването, което проверяващите програми трябва да изпълняват е да приемат 3 аргумента при изпълнението си, които са “входен файл”, “изходен файл” и “изходен файл на участника”. Този формат позволява на автора на задачата да сложи данни, които биха помогнали на проверяващата програма в изходния файл на всеки тест и този файл ще се приеме като втори аргумент. В повечето случаи проверяващите програми четат единствено входния файл и изходния файл на участника и оценяват дали решението на участника решава поставения проблем. Изискването е програмата да завърши с изходен код 0, ако решението е правилно и с изходен код различен от 0 ако то е грешно. Освен това всичко, което проверяващата програма напише на козолата ще се появи в лога на системата и ще може да се види от администраторите на системата.

Също е добре да се отбележи, че от скоро Арената има възможността сама компилира проверяващи програми написани на C/C++. Това много улеснява качването на такива програми, тъй като те трябва да са компилирани за системата, на която се проверяват задачите.

2.5 Описание на системата за рейтинги

2.5.1 Математически модел

В тази секция ще разясня как точно работи модела на рейтингите на арената. Както вече казах, решихме да използваме системата за рейтинги на TopCoder [9]. Въпреки, че математическият модел е описан като формули на сайта на TopCoder, каква е идеята зад самият модел не е



Фигура 2: Нормално разпределение

толкова очевидно.

В тази секция ще се опитам да разясня как работи този модел, а в следващата секция ще видим защо всъщност е добър за нашите цели.

Модела е статистически и приема, че при едно състезание разпределението на точките е нормално. Това значи, че хората с най-малко точки са малко, после тези със около среден брой точки най-много и тези които са с най-много точки също са малко. Виж Фигура 2 за нагледен пример.

Системата за оценяване пази 2 стойности за всеки състезател: текущ рейтинг и променливост. Рейтинга е оценка на състезателя спрямо останалите състезатели, а променливостта оценява до колко представянията на този участник са постоянни, т.е. ако състезателя винаги е на първо място неговата променливост ще е малка, но ако понякога е на първо и понякога на последно място, тогава променливостта му ще е голяма. Можем да си мислим и за рейтингите като случайни величини, а променливостта е тяхната дисперсия.

След всяко състезание, алгоритъма преизчислява рейтингите и променливостта. Алгоритъма, който прави това взема като вход класиране на състезатели, техните рейтинги и тяхната променливост. Трябва да се отбележи, че това колко точки има всеки състезател е без значение. Важно е единствено мястото в класирането. Ако двмата състезателя имат равен брой точки, то се смята че те са на едно и също място, което е средното аритметично от местата покрити от тях, т.е. ако има трима човека на 1 място с равен брой точки, т.е. те са все едно на $(1 + 2 + 3)/3 = 2$

място.

Отначало се изчислява средният рейтинг на участниците в състезанието:

$$AveRating = \frac{\sum_{i=1}^{NumCoders} Rating_i}{NumCoders} \quad (1)$$

Важно е да се отбележи, че винаги се съпоставят състезатели, които са участвали на едно и също състезание с едни и същи задачи.

В тази формула $Rating_i$ е рейтинга на i -тият състезател, а $NumCoders$ е общият брой състезатели.

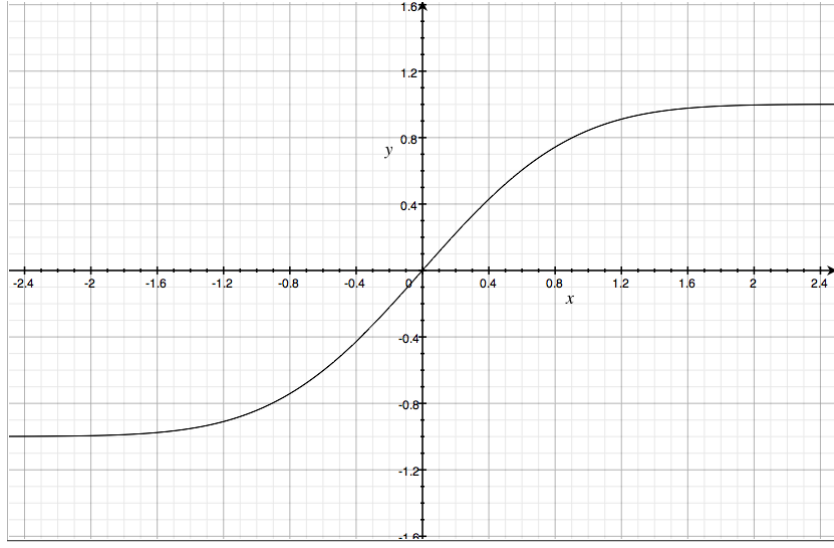
Следващата стъпка е да се изчисли, конкурентността на състезанието. Това е число, което зависи от две компоненти. Първо от средното квадратично от променливостта на всички състезатели, и второ от това колко са различни рейтингите на всички състезатели. Колкото повече състезатели има със различни рейтинги, толкова това число е по-голямо. Ето и формулата която се ползва:

$$CF = \sqrt{\frac{\sum_{i=1}^{NumCoders} Volatility_i^2}{NumCoders} + \frac{\sum_{i=1}^{NumCoders} (Rating_i - AveRating)^2}{NumCoders - 1}} \quad (2)$$

Идеята е, че колкото по-конкурентно е едно състезание, толкова повече точки носи то.

Следващата формула е може би една от най-интересните. Това което се опитва да изчисли тя е каква е вероятността един състезател да има повече точки от друг състезател, т.е. да е по-напред в класирането от него. Тъй като предполагаме, че разпределението е нормално, можем да използваме функцията за грешка на нормалното разпределение и да използваме разликата на рейтингите на двамата състезатели. Функцията за грешка има стойност 0 за 0, което ще рече, че ако имаме двама състезателя с еднакъв рейтинг то шанса всеки един от тях да е по-напред от другия е 0.5, т.е. те имат равни шансове. Както виждате на Фигура 3 тази функция има много подходяща графика, която е много стръмна около нулата и постепенно става по-полегата, т.е. колкото е по-голяма разликата в рейтингите на двамата състезателя, толкова преимуществото на по-силният ще е по-малко осезаемо.

Тъй като рейтингите, са на практика случайни величини и когато ги изваждаме ние създаваме нова случайна величина. Затова трябва да разделим разликата на тяхната комбинирана дисперсия. Освен това имаме



Фигура 3: Функция на грешката

условието при равни рейтинги вероятността да е 0.5, което ни води към следната формула:

$$WP = 0.5 \left(\operatorname{erf} \left(\frac{\operatorname{Rating1} - \operatorname{Rating2}}{\sqrt{2(\operatorname{Vol1}^2 + \operatorname{Vol2}^2)}} \right) + 1 \right) \quad (3)$$

След като имаме тази формула, можем да изчислим вероятността всеки състезател да победи всеки друг и така можем да изчислим предполагаемото място на всеки състезател в класирането:

$$ERank = 0.5 + \sum_{i=1}^{NumCoders} WP_i \quad (4)$$

Където WP_i е вероятността текущият състезател да победи състезателя с номер i .

Следващата стъпка е да изчислим каква е вероятността състезателя да е на оцененото място. Това е лесно тъй като предполагаме, че състезателите са разпределени нормално в класирането, така че ще ползваме обратната на кумулативната функция на нормалното разпределение Φ . Тук изваждаме 0.5 и делим на брой състезатели, за да можем да работим с нормалното разпределение, което работи с интервала (0..1).

$$EPerf = -\Phi \left(\frac{ERank - 0.5}{NumCoders} \right) \quad (5)$$

По аналогичен начин изчисляваме реалното представяне на състезателя като вземаме неговото реално място $ARank$ в състезанието и изчисляваме каква е вероятността той да е на него:

$$APerf = -\Phi \left(\frac{ARank - 0.5}{NumCoders} \right) \quad (6)$$

Така можем да изчислим представянето на състезателя като рейтинг. Това е разликата между това което очакваме и това което реално той е показал умножено по конкурентността на състезанието плюс старият рейтинг на състезателя:

$$PerfAs = OldRating + CF * (APerf - EPerf) \quad (7)$$

Тук е много важно да отбележим какво става, когато едни състезател участва за първи път. В този случай той няма реално рейтинг в системата и за целта му слагаме фиктивен такъв. Всеки нов състезател започва със рейтинг 1200 и променливост 515. Освен това в началото се позволява на състезателите да си променят по-бързо рейтинга и за целта се изчислява функция на тежестта, която зависи от броя участия на състезателя:

$$Weight = \frac{1}{1 - \left(\frac{0.42}{timesPlayed+1} + 0.18 \right)} - 1 \quad (8)$$

Освен това се изчислява и максимална промяна на рейтинга:

$$Cap = 150 + \frac{1500}{TimesPlayed + 2} \quad (9)$$

Накрая се изчислява новият рейтинг по този начин:

$$NewRating = \frac{Rating + Weight * PerfAs}{1 + Weight} \quad (10)$$

Като, ако $|NewRating - Rating| > Cap$, то $NewRating$ се намества така, че той да не се променя повече от Cap .

Освен това се изчислява и новата променливост, която е приближение на дисперсията на случайна величина, като се има предвид и тежестта на състезанието:

$$NewVolatility = \sqrt{\frac{(NewRating - OldRating)^2}{Weight} + \frac{OldVolatility^2}{Weight + 1}} \quad (11)$$

2.5.2 Примери

След като разяснихме как работи модела, нека да разгледаме малко примери за да видим защо всъщност този модел е успешен и как той ни помага. Нека например да разгледаме резултатите от състезанието Арена 8 от 2010 година:

Място	Очаквано място	Участник	Точки	Стар рейтинг	Нов рейтинг	Разлика
1	1	Антон Анастасов	195	2104.28	2165.4	61.12
2	6	Momchil Peychev	175	1548.29	1660.53	112.24
3	4	Йордан Чапъров	165	1654.95	1707.54	52.59
3	9	Тодор Марков	165	1358.14	1469.44	111.3
5	3	Николай Хубанов	135	1751.74	1736.54	-15.2
5	7	Dimitar Hristov Hristov	135	1499.91	1538.06	38.15
7	10	Михаил Ковачев	130	1304.56	1344.68	40.12
8	11	Красимир Георгиев	105	1281.06	1294.7	13.64
9	5	Никола Таушанов	100	1611.43	1550.93	-60.5
10	12	Емануел	55	1196.11	1189.25	-6.86
11	13	Александър Коджабашев	35	1057.37	1048.67	-8.7
12	8	Михаил Карев	25	1385.14	1308.65	-76.49
13	2	Румен Христов	0	1996.16	1738.96	-257.2
13	14	Владимир Владимиров	0	1011.67	952.8	-58.87
13	15	zelenkroki	0	750.03	742.64	-7.39

Участниците са наредени по това колко точки имат на състезанието, а в колоната “Очаквано място” е мястото, което се предполага те да заемат спрямо тяхният рейтинг. Това всъщност е критерият, който се ползва от системата за рейтинги.

Както можете да видите Антон Анастасов е на 1 място, и това не неговото очаквано място. Затова и той си увеличава рейтинга, но не с много. От друга страна на второ място е Момчил Пейчев, който е успял да изпревари 4 състезателя с по-голям рейтинг от неговият, т.е. той се е представил много по-добре от колкото рейтинга му предполага и затова получава доста по-голямо увеличение на рейтинга. От друга страна ако разгледаме Румен Христов, той е със втори най-голям рейтинг, но се е класирал с 0 точки и респективно получава най-съществено намаляване на рейтинга.

zelenkroki е с най-нисък рейтинг преди състезанието, класира се на последно място, т.е. това е напълно очаквано и както виждаме рейтинга на този състезател почти не се променя.

Така както виждаме алгоритъма се опитва да “окуражи” състезателите да се представят по-добре от състезатели с по-голям рейтинг от техният, както и да наказва състезатели, които са с висок рейтинг, но постигат ниски резултати. Това е много подходяща схема за оценяване, тъй като взема предвид как се е представял един състезател преди състезанието и ако един състезател с много нисък рейтинг се представи добре на състезание от по-високо ниво, той ще напредне много бързо, докато ако състезател с много високо ниво участва на състезание, на което участват само състезатели с по-нисък рейтинг от неговият той няма да спечели много точки ако е на 1 място, докато ако не се представи добре ще загуби много точки. Тази система ни накарва да мислим, че рейтинги

ще накарат състезателите да участват в дивизии от техният ранг, както и, че ще имаме възможност въз основа на рейтингите да правим разграничаване на състезателите и да им позволяваме да участват в състезания от по-ниско ниво, ако преди това са показали че са много добри.

2.5.3 Прибавяне на външни състезания

След като интегрирахме системата за рейтинги се подори идеята да я използваме върху националните състезания по програмиране. Ако можем да добавим тези състезания към алгоритъма ще можем да получим нещо като национална ранклиста на състезателите по програмиране и ще можем да следим как се движи нивото им (рейтинга) във времето. Това е много добра идея, но как ще свържем състезателите от националните състезания със потребители на арената?

Подхода, който избрахме е да ползваме имената и града, от който са състезателите. Тъй като в арената имаме информация, от кой град е всеки потребител, то можем да филтрираме потребителите от даден град и да се опитаме да съвпадне имената. Критерият, който използвахме е да гледаме за поне 2 съвпадения на имена, т.е. разделяме името на състезателя на отделни думи и гледаме за поне 2 съвпадащи думи. Освен това пазим история за това кое име с кой потребител сме свързали и ако пак срещнем това име от този град използваме същият потребител. Това ни дава възможност ръчно да свържем някой състезател с потребител от системата и да може след това автоматично да свързваме този състезател с правилният потребител.

Интересно е и как осъществихме лесно въвеждане на резултатите от състезанията в системата. Тъй като много често резултатите се качват като Excel таблици в интернет, много лесно от таблиците може да се сорю/paste-нат резултатите, които се появяват като таблица използваща табулации за разделители. Това позволява лесно да се обработят тези таблици и да се въведат в системата, след което алгоритъма за свързване на имена към потребители се пуска върху данните. За повече информация вижте екраните представени в секцията с описание на системата.

2.5.4 Национална ранклиста на състезателите

Така логично се стигна до идеята да се направи национална ранклиста на състезателите по рейтинги. Класирането по рейтинги е много по-мощно средство от това просто да се сумират точките от състезанията, тъй като мястото е това което е опреелящо за един състезател и чрез неговият рейтинг много лесно могат да се изследват тенденции в представянето на

състезателите. Много лесно може да се види, кой състезател се намира в добра форма и напредва бързо, както и състезатели, които се представят под нивото си и изостават.

За съжаление подържането на такава ранглиста изисква редовно качване на най-новите класирания от национални състезания, както и постоянно подобряване на алгоритъма за свързване на имена със потребители от системата. За момента нямаме толкова много ресурси за да провеждаме това и едната такава ранглиста остава все още идея в процес на разработка.

3 Анализ на постигнатото до момента

В тази секция ще напиша постоженията, които успяхме да постигнем със системата Арена. Също и какво научихме в процес на работа. Тук говоря в множествено число, тъй като успехите се дължат не само на разработката на системата, с която се занимавах аз, но на общите усилия на много други хора, които организираха състезания, превеждаха задачи и отговаряха на въпроси на състезателите.

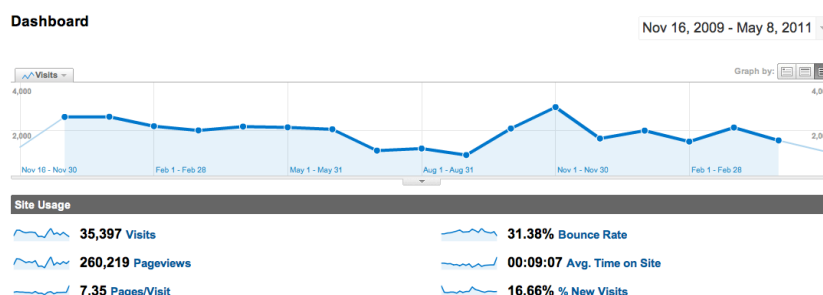
3.1 Статистика на съдържанието на системата

Нека като за начало да направим анализ на съдържанието, което успяхме да съберем в Арена за около година и половина:

- Регистрирани потребители - 763
- Състезания в системата - 59
- Задачи в системата - 321
- Пратени решения - 20841
- Общ обем на тестовите данни и условията на задачите - 1.4ГБ
- Брой качени анализи на задачи - 55
- Брой потребители, които а пратили поне 1 решение за последният 1 месец - 92

Както се вижда Арена вече има една доста солидна база данни от задачи и анализи, които са достъпни по всяко време на желаещите да се занимават. Освен всичко друго всички задачи са на български език,

което ги прави достъпни за по-малките състезатели. Както казах в началото идеята на системата е да е полезна най-вече на малките състезатели и това я отличава от всички други системи в Интернет.



Фигура 4: Статистика за посещаемостта от Ноември 2009 до Май 2011

3.2 Статистика за посещаемостта на системата

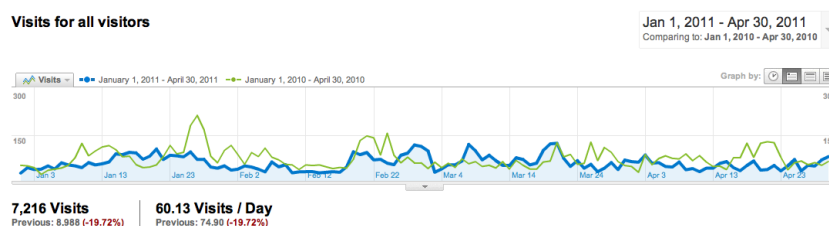
Както се вижда от Фигура 4, посещаемостта на Арената е около 2000 уникални посещения на месец. Трябва да се отбележи, че през първата година имаше доста организирани състезания на арената, което създаде доста трафик към Арената. Може да се види как във дните, когато се организира състезание има покачване от посещаемостта, след което тя спада. Например ако разгледаме статистиката на посещаемостта от 1 Декември 2009 до 31 Март 2010, може да се види как на всеки един пик на посещаемостта съответства състезание организирано на Арената. Виж Фигура 5 за пример.



Фигура 5: Анализ на посещаемостта от 1 Декември 2009 до 31 Март 2010

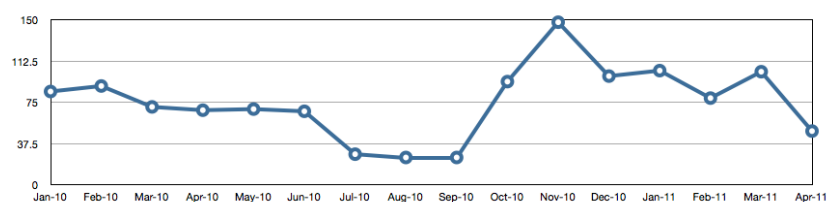
Интересното е че през 2011 година на Арената не се организират толкова много състезания както през 2010. Докато през 2011 се организираха 12 състезания с по 3 дивизии, което прави над 30 състезания, през 2011 има организирани само 6 състезания, които освен това са предназначени за най-малките състезатели. Противно на очакванията посещаемостта не е спаднала съществено въпреки липсата на състезания. Това, което може да значи това е, че хората използват Арената за практикуване, което е една от целите на системата. Статистиката може да се види на Фигура 6

Друга интересна статистика показваща популярност е броя на потребителите, които са пратили поне 1 решение. Това е добър индикатор



Фигура 6: Съпоставка на посещаемостта - 1 Януари до 30 Април 2010 и 2011

за активните уникални потребилите на системата, които я ползват за трениране и подготовка. Като се направи разбивка на това число по месеци (Фигура 7) се вижда в началото на учебната 2010-2011 година, че популярността се е покачила осезаемо, въпреки по-малкото количество състезания. Вижда се също и спадане на това число последните месеци, което може да се дължи на липсата на състезания в последно време. Интересна забележка, е че тази статистика много добре показва



Фигура 7: Брой на потребителите пратили поне 1 решение разбито по месеци

Изводът е, че въпреки, че инерцията, която бяхме набрали в организирането на състезанието е намаляла, това което направихме минлата година е помогнало да се популяризира системата и да се превърне в един от инструментите за подготовка на състезателите по информатика в България.

3.3 Статистика за популярността на състезанията

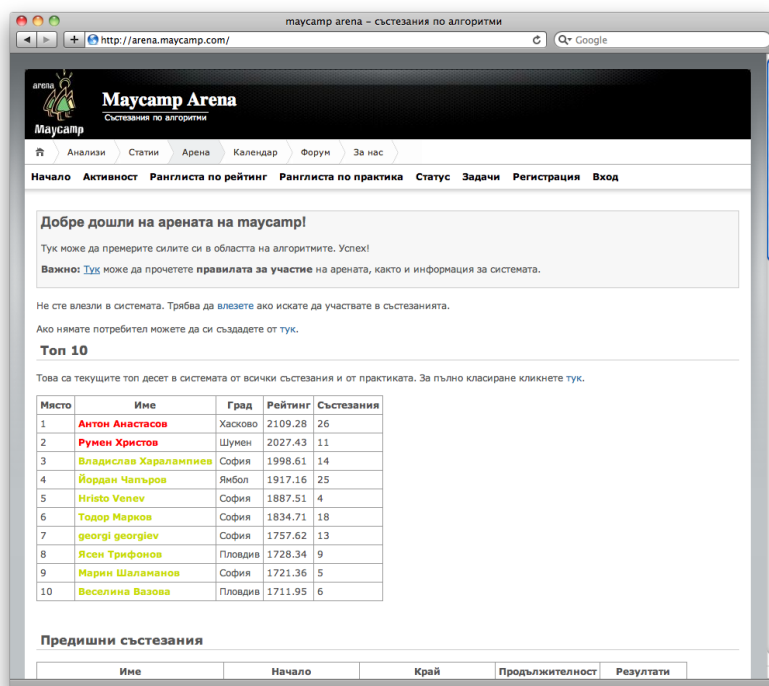
3.4 Скалируемост

4 Примери и описание на работата със системата

В тази секция има описание на основните части на системата. Има описание на това какво виждат учениците, как правят състезания, как си виждат резултатите и как пратикуват. Има описание и на учителската част, на това как се създава учителски акаунт, как се създава състезание, как се слагат задачи и тестове и как се генерират резултатите и рейтингите.

4.1 Част за ученици

4.1.1 Начална страница



Фигура 8: Maucamp Arena - Начална страница

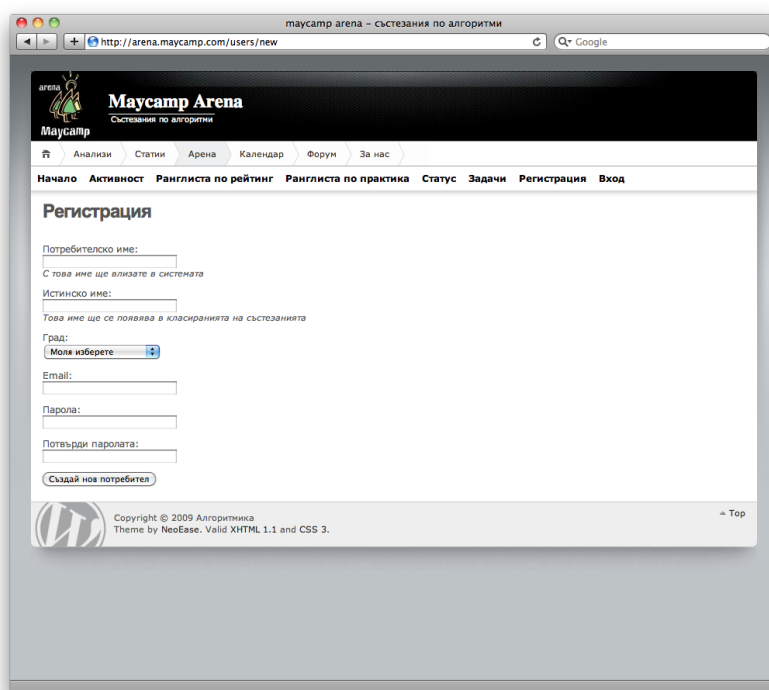
Когато учениците влязат в сайта, първото нещо което виждат са първите 10 участника по рейтинг в системата. Това е направено с цел учениците да са мотивирани да стигнат до първите 10 и така да се появяват на началната страница на сайта. След класирането следва списък с минали състезания и резултати от състезанията. Идеята на този списък е да заинтригува потенциалният потребител на арената и да му покаже, че има редовно организирани състезания, както и хора, които участват на тях. Най-отгоре има меню със навигация, което препраща към няколко страници видими за потребители без регистрация. Това са:

- **Активност** - показва хората, които са практикували през последната седмица. Тази страница има мотивационен характер и служи да покаже на състезателите, че има хора, които тренират. Основното послание, което искаме да изкажем в арената е: “с труд и тренировки се постигат резултати и ако не тренираш другите ще те изпреварят”
- **Ранглиста по рейтинг** - това е глобалната ранглиста по рейтинг на състезателите. Тук са абсолютно всички състезатели в арената. Личният ни опит показва, че е важно за един състезател да може да следи ясно прогреса си. В случая на тази страница това е мястото по рейтинг, на което е в момента. Това отново е мотивационна страница.
- **Ранглиста по практика** - същото като предишната точка, но този път класирането е по практика. Идеята на това класиране е до колко състезателите са решили задачите в системата по време на практика. Колкото повече работещи решения има един състезател, толкова повече точки има в това класиране. Това което се забелязва като интересна тенденция, е че състезателят на първо място по рейтинг е и на първо място по практика.
- **Статус** - това е статус на пратените решения в реално време. Тук може да се види всяко едно решение пратено към системата. На тази страница състезателите могат да следят резултатите от тестването на задача, която са пратили решение, както и да гледат как по същото време някой друг състезател праща задачи. Това освен функционален има и мотивационен характер, тъй като решаването на задачи се превръща в известна степен на групово действие.
- **Задачи** - това е списък със всички задачи в системата. Задачите са подредени по хронологичен ред и за всяка задача има оценка на

трудността, което е средният брой точки от всички пратени решения за задачата. Идеята е че лесните задачи имат повече решения за повече точки от колкото по-трудните. Това е странително обективен начин за оценяване на трудността, тъй като често се случва на състезания от високо ниво да попадне много лесна задача или на състезание за по-малки състезатели да има прекалено трудна задача.

- **Регистрация** - това е формата за регистрация на нови потребители
- **Вход** - от тук регистрираните потребители влизат в системата

4.1.2 Регистрация



The screenshot shows a web browser window with the URL `http://arena.maycamp.com/users/new`. The page title is "maucamp arena - състезания по алгоритми". The website header includes the "Maucamp Arena" logo and navigation links: "Анализи", "Статии", "Арена", "Календар", "Форум", and "За нас". A secondary navigation bar contains links: "Начало", "Активност", "Ранглиста по рейтинг", "Ранглиста по практика", "Статус", "Задачи", "Регистрация", and "Вход". The main content area is titled "Регистрация" and contains the following form fields and instructions:

- Потребителско име:** [text input]
- С това име ще влизате в системата
- Истинско име:** [text input]
- Това име ще се появява в класиранията на състезанията
- Град:** [dropdown menu with "Моля изберете" as the selected option]
- Email:** [text input]
- Парола:** [text input]
- Потвърди паролата:** [text input]
- Създай нов потребител** [button]

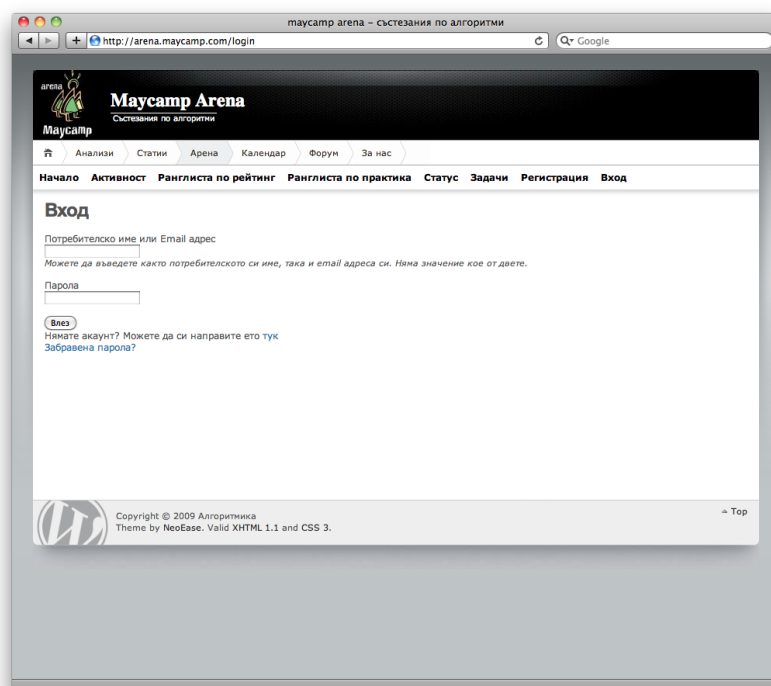
The footer includes a logo, copyright information "Copyright © 2009 Алгоритмика", and the text "Theme by NeoEase. Valid XHTML 1.1 and CSS 3." along with a "Top" link.

Фигура 9: Maucamp Arena - Регистрация

При проектирането на формата за регистрация се постарахме да съкратим данните, от които се нуждаем до минимум. Причината да има

поле “Истинско име” е класиранията от състезанията да изглеждат подобни на тези от националните състезания. Това е и ролята на полето “Трад”. При националните състезания има неофициално съревнование между школите на градовете и дори понякога се прави “неофициално” класиране по градове. Като бъдеща функционалност сме обмисляли дори да добавим класиране по градове, за да има още по-голяма мотивация за състезателите да се представят добре. Полето “Email” не нужно да бъде осъществявана връзка със състезателите. Системата поддържа пращане на масирани писма до всички потребители, като обикновено се ползва за много важни обяви, като например състезания, които ще се проведат скоро или нова функционалност от голямо значение, която е добавена скоро.

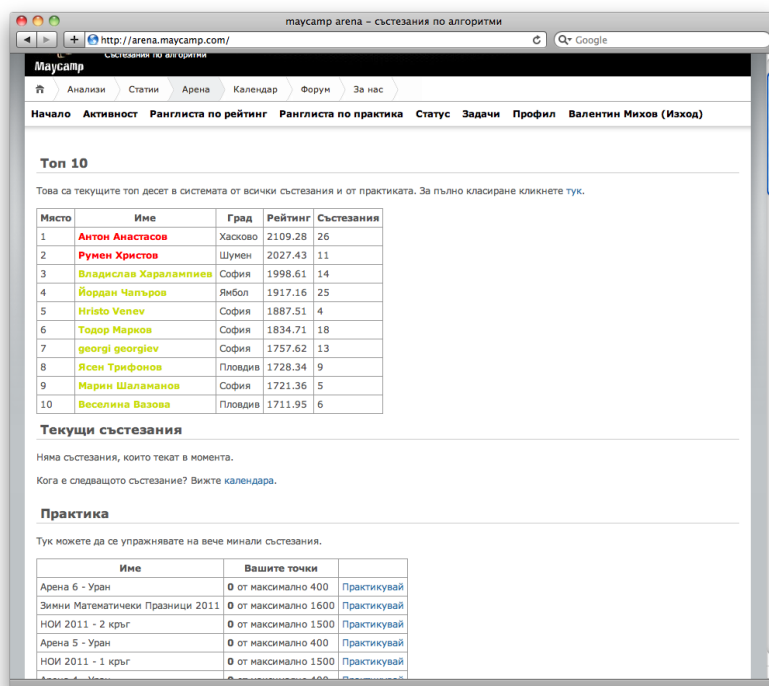
4.1.3 Вход



Фигура 10: Maycamp Arena - Вход

Страницата за вход е доста проста, като тук за улеснение на потребителите сме направили възможен входа както с потребителско име

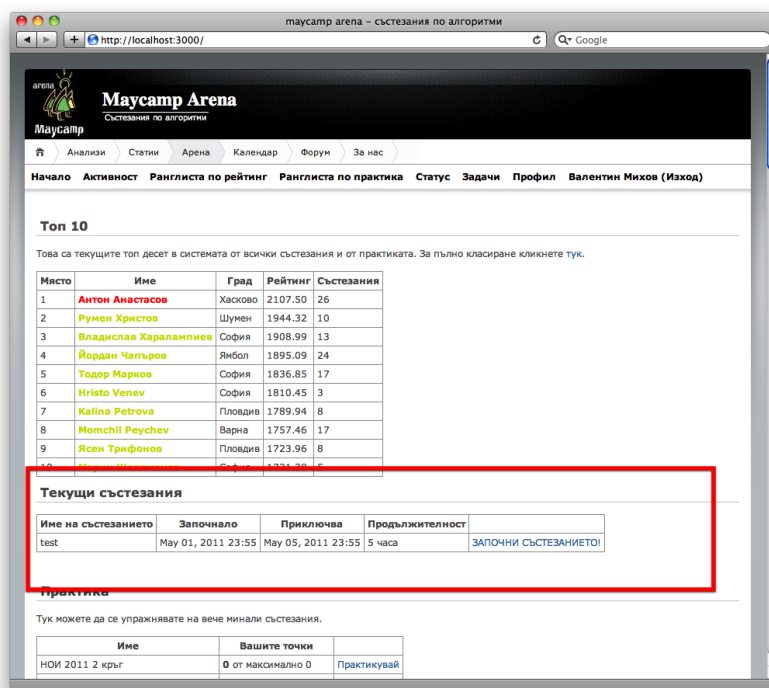
така и със email. Има и страндартна препратка “Забравена парола?”, която позволява да рестартирате паролата на акаунта си и той да ви бъде пратен по email.



Фигура 11: Mauscamp Arena - Начална страница на влязъл потребител

След като потребителят влезе в системата началната страница се променя както можете да видите на Фигура 11. Помяната е, че след списъкът от първите 10 състезателя има смисък от състезания, които се провеждат в момента. След списъка от текущи състезания има списък от минали състезания които можете да практикувате, като до всяко състезание пише колко точки имате събрани. Идеята е да може да се види, на кои състезания все още има задачи, които не сте решили, за да можете да се концентрирате върху тях.

4.1.4 Участие на състезание



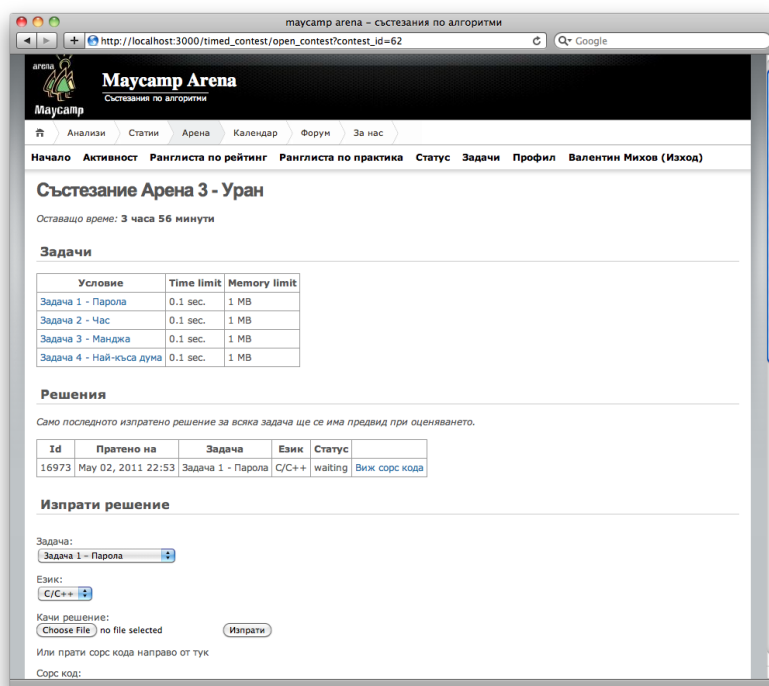
Фигура 12: Маусамп Арена - Начална страница със текущо провеждащо се състезание

Когато има текущо състезание на арената, след като потребителя влезе в системата се вижда това състезание в списъка на текущи състезания. До него има и линк “ЗАПОЧНИ СЪСТЕЗАНИЕТО!” (виж Фигура 12). Както се вижда от фигурата всяко състезание си има начална дата, крайна дата и продължителност, като идеята на това е че състезателя може да прави състезанието по всяко време между началната и крайната дата и след като кликне на линка за започване на състезанието ще има на разположение определено време за да реши задачите. На фигурата се вижда че тази продължителност е 5 часа, т.е. състезателя ще има на разположение 5 часа да реши задачите след което няма да има право да праща повече решения.

След влизане в състезанието участникът вижда задачите и може да прочете условията им. Освен това са показани и лимитите за време и памет за всяка задача. Следва списък от пратените решения от състезателя, като за всяко решение може да се види какъв е статуса върнат от

системата по оценяването. Тук според това в какъв режим е състезанието всяко решение се оценява с първият тест или със всички тестове. Това е за да може да се провеждат състезания както и с моментално класиране така и със оценяване, което се провежда след края на състезанието.

Най-долу на страницата има форма за пратане на решение, като може както да се качи файл, така и да се прати сорса от страницата.



Фигура 13: Mauscamp Arena - Страница на текущо провеждащо се състезание

4.1.5 Резултати от състезание

маусkamp arena - резултати по алгоритми

Q

Google

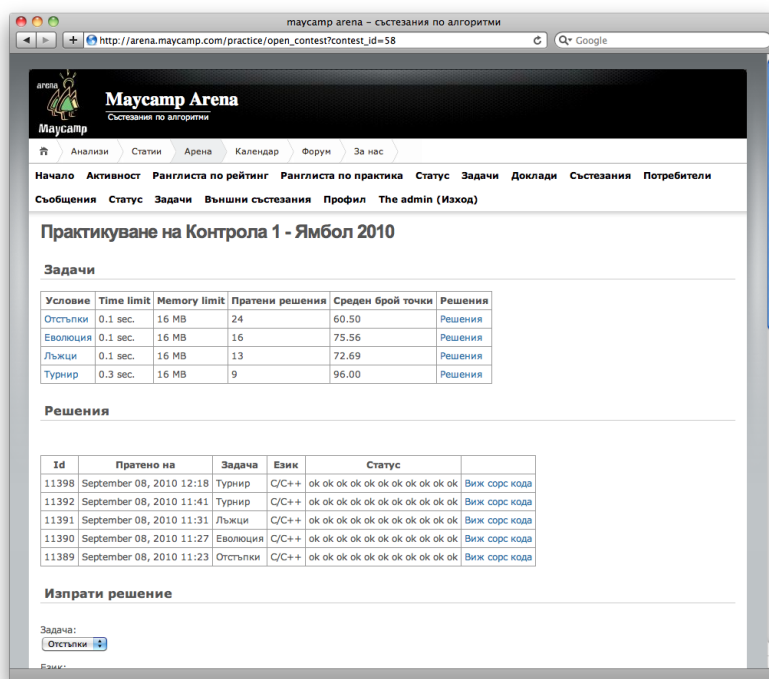
Резултати от Арена 3 - Уран

Место	Участник	Град	Задача 1 - Парола										Задача 2 - Час										Задача 3 - Манджа										Задача 4 - Най-малка дупка										Общ точки	Стар рейтинг	Нов рейтинг	Разлика
			1	2	3	4	5	6	7	8	9	10	Общо	1	2	3	4	5	6	7	8	9	10	Общо	1	2	3	4	5	Общо																
1	zelenkloki	Тринева	10	10	10	10	10	10	10	10	10	80	10	10	10	10	10	10	10	10	10	10	100	10	10	10	10	10	10	10	20	20	20	20	100	380	990.87	1153.48	162.61							
1	San Gaku	Благоевград	10	10	10	10	10	10	10	10	80	10	10	10	10	10	10	10	10	10	10	10	100	10	10	10	10	10	10	10	20	20	20	20	100	380	1305.53	1469.83	164.3							
1	Мариан Шатеванов	Сидон	10	10	10	10	10	10	10	10	80	10	10	10	10	10	10	10	10	10	10	10	100	10	10	10	10	10	10	10	20	20	20	20	100	380	1490.86	1619.96	129.1							
1	Галин Георгиев	Шумен	10	10	10	10	10	10	10	10	80	10	10	10	10	10	10	10	10	10	10	10	100	10	10	10	10	10	10	10	20	20	20	20	100	380	1180.81	1303.59	142.78							
1	Анна Бойкова	Тринева	10	10	10	10	10	10	10	10	80	10	10	10	10	10	10	10	10	10	10	10	100	10	10	10	10	10	10	10	20	20	20	20	100	380	1191.52	1337.29	145.77							
1	Георги Митовски	Варна	10	10	10	10	10	10	10	10	80	10	10	10	10	10	10	10	10	10	10	10	100	10	10	10	10	10	10	10	20	20	20	20	100	380	1355.22	1535.39	180.17							
7	Иван Митовски	Пловдив	10	10	10	10	10	10	10	10	80	10	10	10	10	10	10	10	10	10	10	10	90	10	10	10	10	10	10	10	20	20	20	20	100	370	1581.4	1617.69	36.29							
7	Иван Митовски	Варна	10	10	10	10	10	10	10	10	70	10	10	10	10	10	10	10	10	10	10	10	100	10	10	10	10	10	10	10	20	20	20	20	100	370	1581.4	1617.69	36.29							
7	Павел Невес	Велеш	10	10	10	10	10	10	10	10	80	10	10	10	10	10	10	10	10	10	10	10	90	10	10	10	10	10	10	10	20	20	20	20	100	370	696.26	879.77	183.51							
10	Никола Митов	Русе	10	10	10	10	10	10	10	10	70	10	10	10	10	10	10	10	10	10	10	10	90	10	10	10	10	10	10	10	20	20	20	20	100	360	1243.25	1306.29	63.04							
10	Димитър Попов	Русе	10	10	10	10	10	10	10	10	70	10	10	10	10	10	10	10	10	10	10	10	90	10	10	10	10	10	10	10	20	20	20	20	100	360	1049.31	1167.88	118.57							
12	Иван Митовски	Пловдив	10	10	10	10	10	10	10	10	70	10	10	10	10	10	10	10	10	10	10	10	70	10	10	10	10	10	10	10	20	20	20	20	100	340	1187.2	1252.31	65.11							
12	Мариан Шатеванов	Хасково	10	10	10	10	10	10	10	10	80	10	10	10	10	10	10	10	10	10	10	10	60	10	10	10	10	10	10	10	20	20	20	20	100	340	1200.0	1286.24	86.24							
14	Борислав	Пловдив	10	10	10	10	10	10	10	10	60	10	10	10	10	10	10	10	10	10	10	10	70	10	10	10	10	10	10	10	20	20	20	20	100	330	1508.52	1487.81	-20.71							
14	Радослав	Габрово	10	10	10	10	10	10	10	10	70	10	10	10	10	10	10	10	10	10	10	10	60	10	10	10	10	10	10	10	20	20	20	20	100	330	1253.62	1262.05	28.43							
16	Иван Попов	Русе	10	10	10	10	10	10	10	10	30	10	10	10	10	10	10	10	10	10	10	10	80	10	10	10	10	10	10	10	20	20	20	20	100	330	1248.10	1260.63	12.44							
17	Галин Георгиев	Плевен	10	10	10	10	10	10	10	10	60	10	10	10	10	10	10	10	10	10	10	10	50	10	10	10	10	10	10	10	20	20	20	20	80	290	1200.0	1224.16	24.16							
18	mister_son	Тринева	10	10	10	10	10	10	10	70	0	0	0	0	0	0	0	0	0	0	0	0	0	10	10	10	10	10	10	10	10	20	20	20	20	100	270	1105.54	1137.32	31.78						
19	Валентин Младенков	Шумен	10	10	10	10	10	10	10	60	10	10	10	10	10	10	10	10	10	10	10	10	90	10	10	10	10	10	10	10	0	0	0	0	0	250	1075.46	1104.18	28.72							
20	ivan	Сидон	10	10	10	10	10	10	10	90	10	10	10	10	10	10	10	10	10	10	10	10	50	10	10	10	10	10	10	10	0	0	0	0	0	240	1158.76	1159.07	0.31							
21	Димитър Кара	Хасково	10	10	10	10	10	10	10	70	10	10	10	10	10	10	10	10	10	10	10	10	50	0	0	0	0	0	0	0	0	0	0	0	0	20	20	20	20	100	220	1231.22	1206.29	-24.93		
22	Никола Гьонков	Сидон	10	10	10	10	10	10	10	80	0	0	0	0	0	0	0	0	0	0	0	0	0	10	10	10	10	10	10	10	10	0	0	0	0	0	180	1189.79	1167.41	-22.38						
22	Катерина Кънев	Ябдол	10	10	10	10	10	10	10	80	0	0	0	0	0	0	0	0	0	0	0	0	0	10	10	10	10	10	10	10	10	0	0	0	0	0	180	871.7	935.34	63.64						
24	Радослав Велешки	Хасково	10	10	10	10	10	10	10	80	0	0	0	0	0	0	0	0	0	0	0	0	0	10	10	10	10	10	10	10	10	0	0	0	0	0	110	1294.42	1229.31	-65.11						
25	denislav	Русе	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	10	10	10	10	10	10	10	10	0	0	0	0	0	100	1149.28	1091.47	-57.81							

Фигура 14: Маусамп Арена - Резултати от състезание

След края на състезанието, на началната страница излиза класирането. Там за всеки състезател пише от кой град е, точките на всяка задача, общ брой точки и как се е променил рейтинга на състезателя. Името на всеки състезател е оцветено според рейтинга му. Освен като интересна статистическа информация той служи и за това състезателите да се гордеят със своя рейтинг и да се стремят да достигнат по-високо място в ранглистата.

4.1.6 Практика



Фигура 15: Маусамп Arena - Практикуване на състезание

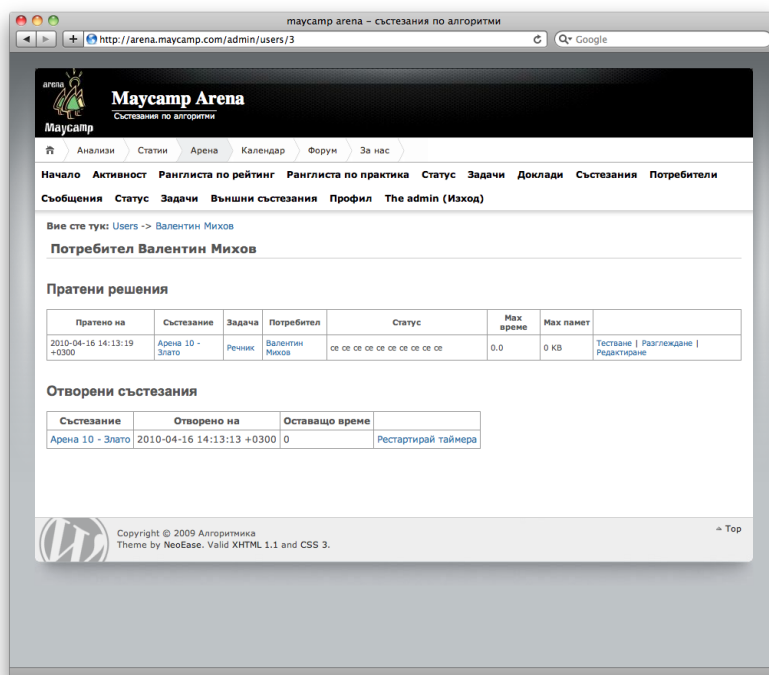
След като е приключило състезанието, има флаг с който може да то се обозначи като такова, което може да се практикува. Когато състезателя иска да практикува състезание, той влиза в страницата на това състезание и вижда страница, много подобна тази по време на състезанието, но този път няма времево ограничение, в което той трябва да прати решението. Освен това се показва и статистика за всяка задача, колко пъти е пратена тя и какъв е средният брой точки на решенията от всички потребители. Може да се види и списък от решенията на задачата, като сорс кода на решенията не се показват никъде в системата. Това помага да се види кой е успял да реши задачата и кои са пробвали да я решат.

4.2 Част за учители

Часта за учители е достъпна само за хора с администраторски права. Тя се използва за да се създават състезания, претестват задачи и администриране на потребители. Също така могат да се гледат графики на

статистики на състезанията и за броя пратени решения към системата.

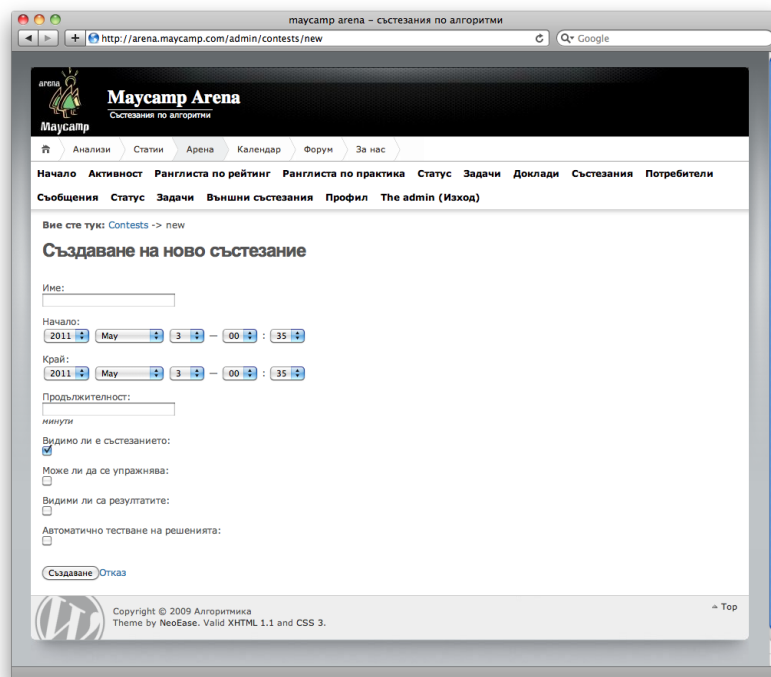
4.2.1 Прибавяне на акаунт на ръководител



Фигура 16: Маускамп Арена - Администриране на потребител

За да бъде направен един акаунт администраторски трябва да се вдигне флаг за потребителя, че той е администратор. Това става на администраторската страница за всеки потребител. Виж Фигура 16.

4.2.2 Създаване на състезание

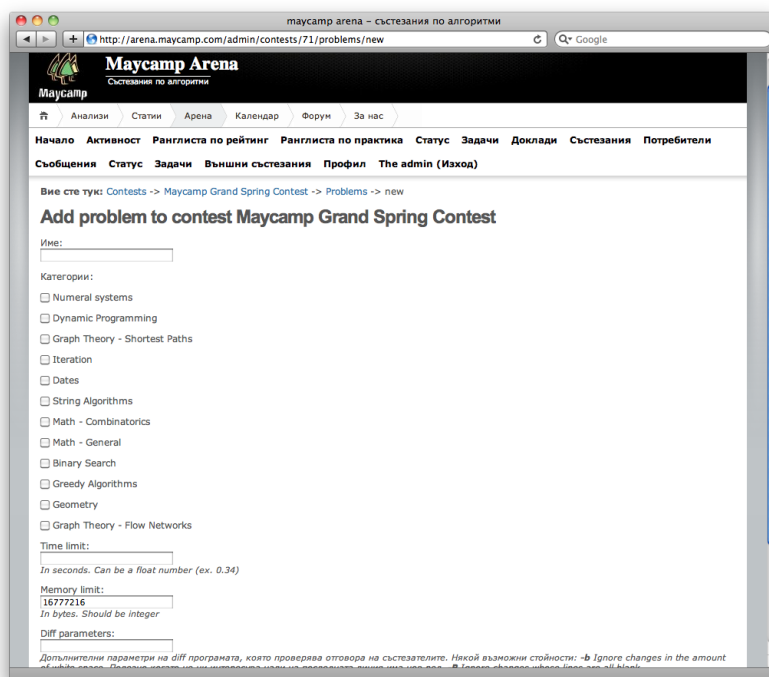


Фигура 17: Маускамп Арена - Създаване на състезание

При създаването на състезание трябва да се окажат началната и крайната дата на състезанието, продължителност и някои флагове, които дефинират поведението на състезанието. Флаговете са следните:

- Видимо ли е състезанието - дали потребителите със състезателни права могат да виждат състезанието на сайта.
- Може ли да се упражнява - дали потребителите ще виждат състезанието в практиката след края му
- Видими ли са резултатите - дали резултатите са видими на началната страница
- Автоматично тестване на решенията - дали всяко пратено решение да бъде тествано веднага срещу всички тестове

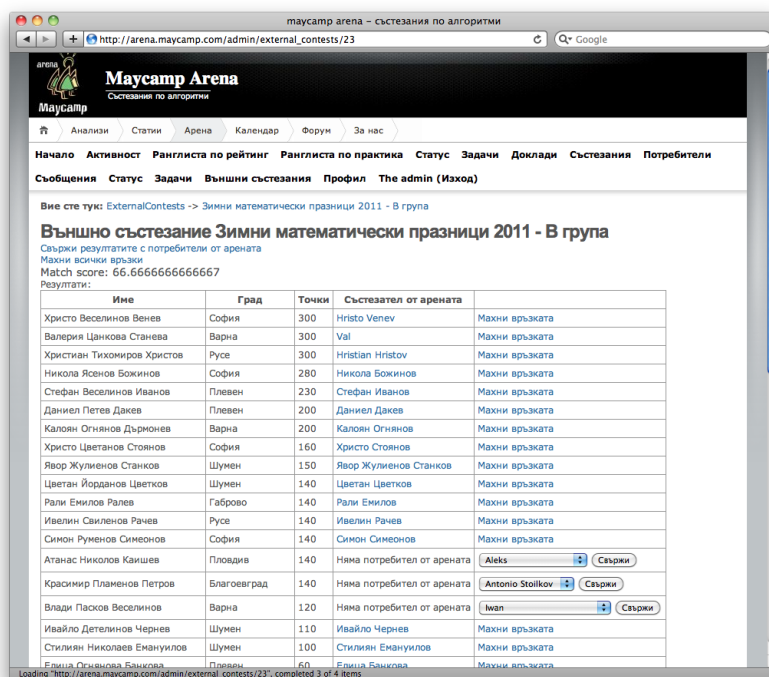
4.2.3 Прибавяне на задачи и тестове



Фигура 18: Маусамп Арена - Създаване на задача

След създаването на състезанието, то излиза в списъка от състезания и там има препратка към “Задачи”, където се създават задачите. За всяка задача се описват лимитите за време и за памет. После за всяка задача могат да се качват файлове, като това обикновено са входните данни, съответните изходни файлове, както и понякога програма за проверка на отговорите (чекер).

4.2.4 Прибавяне на резултати от външно състезание



maucamp arena - състезания по алгоритми

http://arena.maucamp.com/admin/external_contests/23

Мaucamp Arena
Състезания по алгоритми

Анализ Статии Арена Календар Форум За нас

Начало Активност Ранглиста по рейтинг Ранглиста по практика Статус Задачи Доклади Състезания Потребители

Съобщения Статус Задачи Външни състезания Профил The admin (Изход)

Вие сте тук: ExternalContests -> Зимни математически празници 2011 - В група

Външно състезание Зимни математически празници 2011 - В група

Свържи резултатите с потребители от арената

Матрица всички връзки

Match score: 66.6666666666667

Резултати:

Име	Град	Точки	Състезател от арената	
Христо Веселинов Венева	София	300	Hristo Venev	Матрица връзката
Валерия Цанкова Станева	Варна	300	Val	Матрица връзката
Христиан Тихомиров Христов	Русе	300	Hristian Hristov	Матрица връзката
Никола Ясенов Божинов	София	280	Никола Божинов	Матрица връзката
Стефан Веселинов Иванов	Плевен	230	Стефан Иванов	Матрица връзката
Даниел Петев Дакев	Плевен	200	Даниел Дакев	Матрица връзката
Калоян Огнянов Дърмиев	Варна	200	Калоян Огнянов	Матрица връзката
Христо Цветанов Стоянов	София	160	Христо Стоянов	Матрица връзката
Явор Жулиенов Станков	Шумен	150	Явор Жулиенов Станков	Матрица връзката
Цветан Йорданов Цветков	Шумен	140	Цветан Цветков	Матрица връзката
Рали Емилов Ралев	Габрово	140	Рали Емилов	Матрица връзката
Ивелин Савилов Рачев	Русе	140	Ивелин Рачев	Матрица връзката
Симон Руменов Симеонов	София	140	Симон Симеонов	Матрица връзката
Атанас Николов Каишев	Пловдив	140	Няма потребител от арената	Aleks Свържи
Красимир Пламенов Петров	Благоевград	140	Няма потребител от арената	Antonio Stoilkov Свържи
Владислав Пасков Веселинов	Варна	120	Няма потребител от арената	Ivan Свържи
Ивайло Детевинов Чернев	Шумен	110	Ивайло Чернев	Матрица връзката
Стилиан Николов Емануилов	Шумен	100	Стилиан Емануилов	Матрица връзката
Елиса Огнянова Банкова	Плевен	60	Елиса Банкова	Матрица връзката

Loading "http://arena.maucamp.com/admin/external_contests/23", completed 3 of 4 items

Фигура 19: Маусамп Арена - Класиране от външно състезание

Идеята за качването на резултати от външни състезания е те да се включат в общата статистика на рейтингите на състезателите. Така може да се получи нещо, като “национална ранглиста на състезателите”, което има от една страна очевидно мотивационна стойност и от друга страна интересна статистика за състезателите в България. Идеята е това да е ранглиста подобна на световните ранглисти на тенисистите и шахматистите, където да си номер 1 е много престижно постижение.

Зад да се качи външно състезание е нужно да бъде генерирано класирането в подходящият формат. Формата, които избрах се поддържа от повечето програми за обработка на таблици като (Excel и Numbers). Всеки състезател е на отделен ред и на реда, разделени със табулации, са изброени името на състезателя, града от който е и точки от състезанието. Системата прочита тези данни и ги вкарва в базата си данни. След това следва свързване на всеки един ред от класирането със състезател регистриран в системата. Алгоритъмът за свързване е следният:

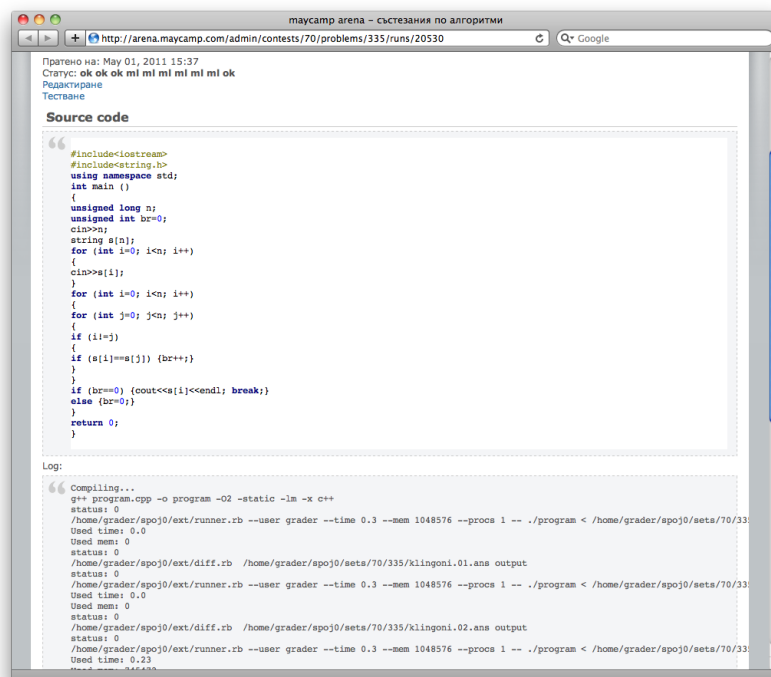
1. Ако вече сме срещали състезател със същото име и град в друго външно класиране и той е асоцииран с потребител на арената, тогава го свързваме с този потребител
2. За всеки потребител на арената от същият град като състезателя от класирането оценяваме до колко имена на потребителя и състезателя съвпадат. Две имена се смята че съвпадат ако първите и последните имена съвпадат.
3. Ако нито един потребител от същият град не съвпада, разглеждаме потребителите без посочен град.

За да е по-точен алгоритъма обръщаме всички имена от кирилица на латиница, тъй като има потребители, които изписват имената си с латиница.

В системата има потребителски интерфейс, който показва как имената от класирането са свързани със потребители от арената и до колко сме успели да свържем класирането. За съжаление никога не може да се постигне 100% покритие на класиранията, тъй като не всеки участник на националните състезания е регистриран в арената, а и не всички регистрирани са посочили точните си имена. Можете да видите на Фигура 19 класирането от В група на Зимните Математически Празници 2011. Както се вижда от фигурата системата е успяла да свърже около 2/3 от състезателите със потребители от арената. Потребителският интерфейс позволява да се махат връзки и да се създават ръчно такива, като идеята е че колкото повече класирания е обработила системата, толкова по-точна ще е тя. Това е заради стъпка 1 от алгоритъма и факта, че имената на състезателите и техните градове се запазват едни и същи между националните състезания.

4.2.5 Тестване на решения

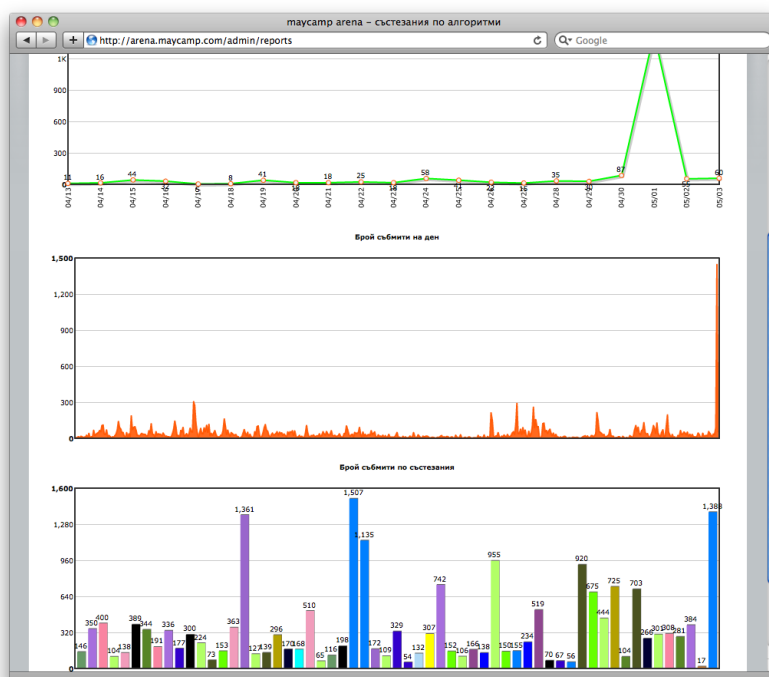
Учителите имат възможността да разглеждат решенията на потребителите на системата, както и да пускат тези решения за повторно тестване. Те имат достъп и до подробен лог от тестването на решението където се вижда какви точно команди са били изпълнение на оценяващата система по време на оценяването, за колко време е работило и колко памет е използвало решението на всеки тест, както и какви различия има изхода на решението с правилното решение на всеки тест, ако има такива. Тъй като това е информация, която може да подсказва на външни хора как да излъжат системата тя не се показва на нормалните потребители.



Фигура 20: Маусамп Арена - Разглеждане на пратено решение

Системата поддържа и глобални операции за повторно тестване на решения, като например да се тестват отново всички решения на дадена задача или на цяло състезание, тъй като е често срещано да се открият грешки в тестовите на някоя задача и да се окаже че резултатите и класирането не са правилни.

Друга важна подробност относно логовете от всяко едно тестване, е че се пази само част от разликите между изхода на решението и правилното решение, тъй като това може да увеличи неимоверно обема на тези логове и да задръсти канала между системата, която оценява и базата данни. Това беше нещо, което беше поправено в последствие, тъй като не беше помислено за него отначало. Тъй като оценяващата машина и базата данни са доста отдалечени една от друга проблема беше много очевиден тъй като скоростта между тях беше малка и системата губеше много време в това да пренася логовете от едната система да другата.



Фигура 21: Маусамп Арена - Доклади

4.2.6 Доклади

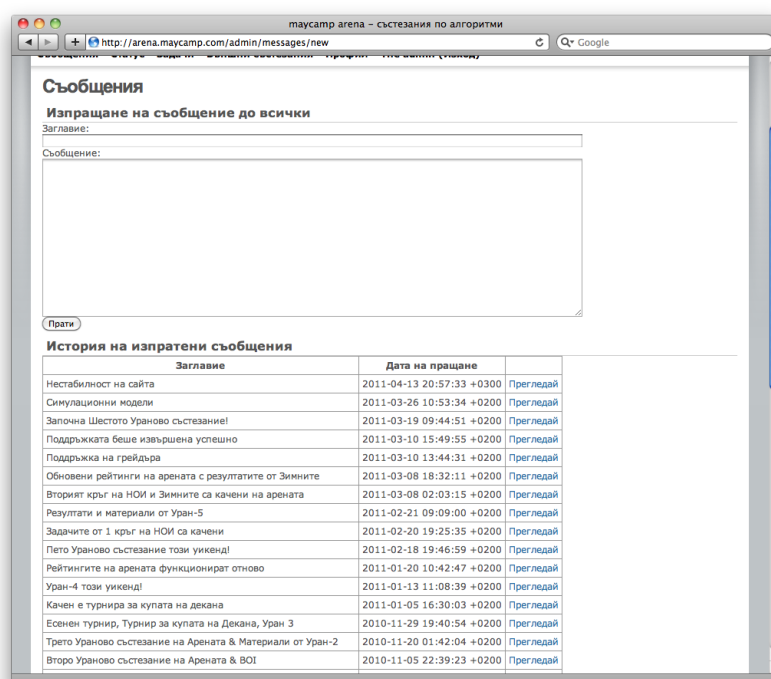
Учителите имат достъп и до статистически данни в арената. За момента има изградени няколко графики, които показват няколко статистики около използването на системата. Тези графики могат да се видят като се отиде в секцията “Доклади”. За момента статистиките, които са изградени са следните:

- Брой пратени решения на ден за последните 3 седмици - това показва графика за броя пратени решения на ден за последните 21 дни
- Брой пратени решения за всички дни - това е глобална статистика на пратените решения от както системата събира данни. Могат да се видят отчетливи покачвания около дните на националните състезания и когато задачите от национално състезание биват качени на арената.
- Брой решения пратени за всяко състезание - графика показваща

за всяко състезание по колко пратени решения има. Забелязват се тенденции като например, че националните състезания имат по най-много пратени решения.

Тези графики служат за да се анализира по какъв начин състезателите използват арената за тренировки и кои състезания са най-търсените. Целата е да се подобри това което арената предлага като задачи, за да могат състезателите да тренират повече.

4.2.7 Пращане на съобщения



Фигура 22: Маусамп Arena - Пращане на съобщения

Арената и секция за пращане на съобщения, която прави пращането на съобщения до всички регистрирани потребители много лесно, както и пази история за всички пратени съобщения до момента. За момента тази функционалност е много проста и не поддържа изключване на хора от масовите съобщения. Тъй като до момента не сме получили нито едно оплакване за това, че хората не искат да получават на нас съобщения

не съм разработил подобна функционалност. Пращането на самите съобщения е доста просто. Използва се ВСС полето на мейл протокола, което позволява да се праща писмо до няколко адреса без получателите да виждат адресите на другите получатели. Тъй като съобщенията се пращат до стотици хора се налага да се групират всички получатели в няколко групи и да се прати по един мейл за всяка група, тъй като мейл сървърите имат ограничение за броя адреси, които могат да се изброят в ВСС полето.

5 Идеи за бъдещето

Въпреки че в момента системата има сравнително добра популярност има много идеи за подобряването и. Тук ще изброя някои от идеите, които в момента имам за бъдещето.

- Поддръжка на няколко оценяващи системи - в момента арената работи само с една система, която оценява пратените решения. Прибавянето на още сървъри, които да оценяват решения ще направи организирането на състезания с оценяване и класиране в реално време по-лесно. Тестовите ми показват, че при едно подобно състезание бързото оценяване може да е проблем, тъй като се натрупват много решения за тестване, е някои моменти (към края на състезанието) и понякога се налага претестване на цели задачи, което още повече забавя оценяването. Като цяло това е функционалност, която се поддържа от архитектурата на системата, но не е тествана и има предизвикателства от сорта на това как времевите лимити се отнасят от система на система.
- Добавяне на метод за комуникация в реално време - това е идея да се добави нещо като “чат” в арената, който да позволява на потребителите, които в момента са на сайта да комуникират по между си. Тъй като процеса по практика обикновено включва продължителен престой на сайта, една подобна функционалност може да се използва за комуникация между практикуващите състезатели. Комуникацията е много важна при състезателното програмиране тъй като се обменят идеи. Разбира се подобна функционалност не трябва да е достъпна по време на състезание.
- Добавяне на други езици за програмиране - в момента системата поддържа единствено C++, а на ученическите състезания се допуска използване на Pascal. Също на много студентски състезания се

позволява използване на Java. Това ще донесе проблеми със следенето на паметта и времето за изпълнение на програмите, тъй като тези езици (особенно Java) по различен начин изпълняват приложенията.

- Интегриране на по-добър начин за изолиране на решенията - идеята е че може да се използват готови решения като например sandbox на Martin Mares, което се използва на международните състезания по програмиране. Интегрирането на подобно решение е сложно тъй като трябва да се прегледат текущите лимити за време и памет и да се променят така че да отразяват новият начин за измерване

Литература

- [1] Manev, Kr., Sredkov, M., Bogdanov, Ts.: Grading Systems for Competitions in Programming, *Mathematics and Education in Mathematics*, Proc. of 38-th Spring Conference of UBM, Borovetz (2009).
- [2] PC2 Home page. <http://www.ecs.csus.edu/pc2/>
- [3] SMOC grading system. <http://openfmi.net/projects/pcms/>
- [4] SPOJ0 training and grading system. <http://judge.openfmi.net/spoj0>
- [5] Ribeiro, P. and P. Guereiro. Early Introduction of Competitive Programming. *Olympiads in Informatics*, 2, 2008, 149-162.
- [6] USACO Training Gateway, <http://train.usaco.org/usacogate/>
- [7] Online judge, Wikipedia, http://en.wikipedia.org/wiki/Online_judge
- [8] USA Computing Olympiad, <http://www.uwp.edu/sws/usaco/>
- [9] Algorithm Competition Rating System, TopCoder, <http://www.topcoder.com/wiki/display/tc/Algorithm+Competition+Rating+System>
- [10] Using a Linux Security Module for Contest Security, Bruce Merry, 2009
- [11] Perspectives on Grading Systems, Martin Mares, 2007
- [12] Performance Analysis of Sandboxes for Reactive Tasks, Bruce Merry, 2010
- [13] Validating the Security and Stability of the Grader for a Programming Contest System, Toncho Tochev, Tsvetan Bogdanov, 2010
- [14] Bulgaria – The Homeland of International Competitions in Informatics for School Students, Petar S. Kenderov, 2009
- [15] Introduction to the Olympiads on Informatics, Dr. Pavel Azalov, 1989 <http://www.ioi2009.org/GetResource?id=238>
- [16] Ruby on Rails - web development framework <http://rubyonrails.org/>

- [17] Cucumber - behavior driven development with elegance and joy <http://cukes.info/>
- [18] Capistrano - Remote multi-server automation tool <https://github.com/capistrano/capistrano>
- [19] rsync - utility that provides fast incremental file transfer <http://samba.anu.edu.au/rsync/>