Universidade Federal de Pernambuco

GRADUAÇÃO EM ENGENHARIA DA COMPUTAÇÃO

CENTRO DE INFORMÁTICA

FERRAMENTA PARA SELEÇÃO DE PROBLEMAS PARA PROGRAMAÇÃO COMPETITIVA

Trabalho de Graduação

Aluno : Sérgio Eduardo Pessoa do Nascimento Filho {sepnf@cin.ufpe.br}

Orientadora: Liliane Rose Benning Salgado {liliane@cin.ufpe.br}

Recife, 09 de Agosto de 2014.

Universidade Federal de Pernambuco

GRADUAÇÃO EM ENGENHARIA DA COMPUTAÇÃO

CENTRO DE INFORMÁTICA

FERRAMENTA PARA SELEÇÃO DE PROBLEMAS PARA PROGRAMAÇÃO COMPETITIVA

Trabalho de Graduação

Aluno : Sérgio Eduardo Pessoa do Nascimento Filho {sepnf@cin.ufpe.br}

Orientadora: Liliane Rose Benning Salgado {liliane@cin.ufpe.br}

Trabalho de graduação apresentado no Centro de Informática da Universidade Federal de Pernambuco por Sérgio Eduardo Pessoa do Nascimento Filho, orientado por Liliane Rose Benning Salgado como requisito para obtenção do Grau de Bacharel em Engenharia da Computação.

Recife, 09 de Agosto de 2014.

Resumo

A maratona de programação é uma atividade extracurricular importante para os alunos que a constituem. O principal problema do processo de treino na maratona de programação é a escolha das questões que serão treinadas pelos envolvidos no projeto. No centro de informática da UFPE, é utilizada atualmente a seleção manual de questões, feitas pela coordenadora do projeto ou por alunos mais experientes. A proposta desse trabalho é o desenvolvimento de um sistema que selecione questões automaticamente de acordo com o nível dos usuários. O ambiente utilizará algoritmos conhecidos e distribuições estatísticas para implementação efetiva do sistema.

Agradecimentos

Agradeço primeiramente a Deus, por ter me dado forças e inspiração para realizar minha graduação. Sem Ele, certamente eu seria uma pessoa diferente em relação a minha pessoa de hoje, e sem sua permissão não teria realizado tudo o que consegui nesse trajeto.

Agradeço também aos meus pais, que me apoiaram durante toda a jornada. Sem seus conselhos quando notavam que eu estava prestes a tomar decisões erradas e sem o apoio emocional que eles me proporcionaram nas fases mais difíceis do meu curso, dificilmente o meu caminhar na universidade teria sido tão tranquilo quanto ele foi, de fato. Estendo esse agradecimento aos meus avôs, que durante grande parte da minha vida me trataram como filho e garantiram que nada faltasse a mim.

Também gostaria de agradecer a todos os meus colegas, que formaram uma das turmas de engenharia da computação mais unidas que já passaram pelo centro de informática. Sem o balanço perfeito entre amizade a ajuda mútua que tive com os mesmos, a jornada no curso teria sido muito mais difícil, já que em muitos estudos coletivos e em muitas dicas de última hora antes da prova consegui ter melhor entendimento dos assuntos estudados.

Aos colegas da maratona de programação, que tiveram papel fundamental no meu desenvolvimento pessoal e profissional, agradeço por me ensinarem que para melhorar no projeto, eu deveria persistir sempre, sem fraquejar. Agradeço principalmente àqueles que confiaram em mim enquanto fazíamos parte de um time, tenhamos ou não competido oficialmente comigo, e àqueles que sempre estiveram dispostos a me ensinar novos tópicos quando eu precisava, ou a me ajudar a depurar problemas quando necessário. A experiência de competir pela maratona de programação foi uma das melhores da minha vida, e sem eles essa experiência não seria possível.

Finalmente, agradeço a todos os professores que levam a arte de ensinar e educar a sério. São esses que fazem o curso de engenharia da computação ser bem graduado no país, e que conduzem os alunos aos ramos que esses mais se identificam.

Sumário

1. Introdução

Com o nascimento da computação, foram intensificadas as pesquisas nas áreas de recorrências matemáticas, algoritmos e estruturas de dados. Por essa demanda, nasceu um projeto que agrega estudantes universitários em uma competição nos temas supracitados. Esse projeto chama-se maratona de programação.

Nas competições da maratona de programação, no formato proposto pela ACM-ICPC, são apresentados ao competidor de 10 a 12 problemas a serem resolvidos, e o aluno deve resolver o maior número de problemas possível em 5 horas. No treinamento para a competição na UFPE, é simulada uma competição por semana, tipicamente aos sábados. Durante a semana, os alunos treinam as questões que não conseguiram resolver durante a simulação, com o objetivo de melhorar suas habilidades.

As questões resolvidas semanalmente estão em sites chamados online judges. Online judges são sites que contém grandes repertórios de provas passadas e questões originais enviadas por problem setters, que são as pessoas envolvidas na criação de problemas.

No caso da UFPE, o treinador é responsável por escolher as questões que farão parte do treino semanal. O treinador normalmente é um dos alunos mais experientes do projeto ou a coordenadora do projeto. Ele sempre deve escolher questões que sejam inéditas aos alunos do projeto e que sejam compatíveis com o nível dos competidores, pois questões fáceis demais não evoluem os alunos, enquanto questões muito difíceis desestimulam os envolvidos no projeto.

Porém, o treinador conhece apenas um número limitado de questões. Isso limita a qualidade dos contests a ser escolhidos, pois é esperado que o mesmo não conheça um bom número de questões potencialmente importantes para o desenvolvimento do grupo.

1.1 Objetivo

A proposta é a criação de uma ferramenta que realize, de forma automática, a escolha das questões a ser treinadas. A ferramenta será capaz de utilizar a informação do nível dos usuários, e, a partir disso, escolher as questões que gerem um contest balanceado para todos os participantes. Além disso, ela precisará também ter uma forma de classificar as questões por dificuldade de acordo com critérios disponibilizados pelos online judges.

Para melhor graduação das questões, consideramos um conhecido algoritmo da área de classificação de documentos, os HITS (Hyperlink-Induced Topic Search) e um método estatístico bayesiano. Para classificação dos usuários, foi utilizado um método estatístico baseado no seu desenvolvimento no projeto.

Como objetivos secundários, estão a mineração de dados dos online judges, atualização periódica dos níveis das questões e usuários, criação de uma interface online simples de fácil uso e integração com o sistema atual de treinamento da maratona na UFPE, o CCS (Cin Contest System).

1.2 Estrutura do Trabalho

Este trabalho está dividido em capítulos, sendo este o primeiro. O segundo capítulo trata das ferramentas que atualmente implementam propostas semelhantes à deste trabalho. Essas ferramentas são estudadas e após análise é feita uma comparação das mesmas, e um estudo de em que tópicos elas sucedem e falham.

O terceiro capítulo é uma curta análise do Cin Contest System, sistema onde a ferramenta apresentada está inclusa. Durante esse estudo, verificamos as características atuais do Cin Contest System.

O quarto capítulo se trata da apresentação do sistema desenvolvido. Nesse capítulo, são feitas análises detalhadas sobre o servidor onde o sistema foi

desenvolvido e sobre a arquitetura de software do sistema implementado. Na análise da arquitetura de software, é estudado o modelo de desenvolvimento do sistema e também feito um breve estudo sobre cada parte da implementação, com a justificativa da escolha das decisões mais importantes para cada módulo.

O quinto capítulo conclui esse trabalho, com uma reflexão sobre a ferramenta proposta e a sugestão de trabalhos futuros.

2. Plataformas Similares

Embora não haja serviço que realize as mesmas funções ao sistema apresentado, existem sistemas que realizam serviços similares, predecessores ao sistema apresentado. Tal área é valorizada pelas pessoas da área, pois o problema de falta de questões conhecidas para escolha é geral entre maratonistas.

Nesse capítulo, serão estudados esses sistemas, com o objetivo de contextualizar a importância da ferramenta apresentada, e mostrar outras formas como o problema de escolher questões pode ser resolvido.

2.1 VOC (Virtual Online Contests)

A2 Online Judge (ou Virtual Online Contests) é um online judge com centenas de problemas e ajuda o usuário a criar, executar e participar em contests virtuais usando problemas dos seguintes online judges: A2 Online Judge, Live Archive, Codeforces, Timus, Codechef, Topcoder, SPOJ, TJU, SGU, PKU, ZOJ e URI. Ele também lhe ajuda a gerenciar e acompanhar seus treinos de maratona e o dos seus amigos. Ele ajuda para treinos da ACM-ICPC e IOI. [1]

A2 Online Judge é um online judge de uso geral, com suporte a escolha de contests antigos completos, criar competições personalizadas e escolher treinos baseados em questões aleatórias.

O sistema contém um módulo onde é possível aos usuários classificar as questões resolvidas por ele em um dos online judges suportados pelo sistema. A classificação ocorre por nível de dificuldade e por assunto.

Por nível de dificuldade, o usuário dá uma nota, entre um e dez, para a questão de acordo com o nível que ele acredita que a questão tenha. O site recebe o input de vários usuários e utiliza-se disso para dar uma nota à

questão. Por assunto, o usuário define uma lista de tags para a questão, que são pré-definidas com assuntos comuns da programação competitiva.

O sistema mostra ao usuário quais questões ele já resolveu, porém não recomenda questões de acordo com o seu nível em determinado assunto.

Atualmente, o sistema funciona apenas para as questões que já foram classificadas voluntariamente por usuários, então apenas um número limitado de questões está adicionado.

2.2 Uva Toolkit

Uva Toolkit é um site com objetivo de ajudar usuários a resolver problemas do Uva Online Judge.[2]

O site consiste de uma lista de problemas. Para cada problema, a plataforma contém informações básicas, como ID, nome, link para o problema no Uva Online Judge, e principalmente, uma lista de tags que identificam o problema.

O sistema é mantido por uma pessoa apenas (Mark Greve), então o repositório de questões do site é tão grande quanto o repertório de questões que ele já resolveu. Isso limita a experiência do usuário com o site, já que variedade de questões é importante para o desenvolvimento das habilidades de resolução de problemas.

Nesse site, é possível ainda enviar um input criado pelo usuário e o site responde com o output correto da questão, de acordo com uma solução aceita escrita por Mark Greve. Isso é útil no desenvolvimento em programação competitiva, pois economiza tempo de depuração de código dos usuários, já que é mais rápido depurar um código quando se conhece um caso de teste onde o código não se comporta como esperado.

2.3 uHunt

uHunt é uma ferramenta de complemento para o Uva Online Judge que mantêm estatísticas, provê seleções de problemas a ser resolvidos, e expõe uma API web para outros desenvolvedores web desenvolverem sobre ele. [3]

Esse site foi desenvolvido com base num livro: “Competitive programming: Increasing the lower bounds of programming competitions”, do mesmo autor (Felix Halim). Esse livro contém uma lista de questões recomendadas para o Uva Online Judge. O livro está em sua terceira versão, e a cada versão lançada, é feita em conjunto uma atualização do site, com os novos problemas classificados.

O site foi desenvolvido inicialmente para manter estatísticas sobre quantos por cento das questões do livro os usuários já tinham resolvido, já que assim os competidores podem acompanhar seus desenvolvimentos pessoais na competição e treinar nos tópicos que julguem que precisam melhorar ou aprender.

Porém, o sistema aumentou com o tempo, e atualmente é capaz também de acompanhar em tempo real as submissões no Uva online judge, além de manter estatísticas sobre todos os usuários do site, como quantas questões cada usuário já resolveu, quais são esses problemas e quais as questões que o usuário já tentou, mas não resolveu.

O sistema classifica as questões do livro com nível entre um e dez, e as separa por assunto. Para problemas que não estão no livro, o sistema apenas as ordena por quantidade de submissões aceitas no mesmo. A plataforma se utiliza da filosofia de que, a facilidade de uma questão é diretamente proporcional ao número de pessoas que a acertaram.

2.4 Comparação

Os sistemas apresentam diferentes perspectivas para resolver o problema do desconhecimento de questões em online judges. O segundo é especialmente diferente, pois não classifica as questões por nível, apresentando assim uma perspectiva diferente sobre a ajuda a contestants.

O A2 Online Judge leva vantagem na classificação, pois os problemas não são classificados por apenas uma pessoa, que aumenta a confiabilidade à classificação, devido ao teorema central do limite, de estatística.

Por ser um projeto com fins lucrativos, o uHunt leva vantagem no número de questões indexadas, e, como as questões são separadas por assunto antes de serem separadas por dificuldade, os usuários que buscam uma maior cobertura de assuntos pode procurar este para conseguir essa cobertura. Para utilização em contests, por outro lado, essa ferramenta não se mostra útil, já que as questões são em sua maioria problemas tutoriais para assuntos, e não requerem uma linha de pensamento complexa do contestant.

3. Cin Contest System

O Cin Contest System, conhecido por CCS pelos integrantes do projeto, é o simulador de competições utilizado pela UFPE nos treinos semanais. Ele é um sistema capaz de carregar questões de diferentes de online judges e mostrar o placar atualizado da competição, à medida que os alunos resolvem as questões propostas.

O sistema tem uma interface de simples uso, de forma que qualquer usuário sem conhecimento avançado do sistema possa utilizá-lo sem maiores problemas.

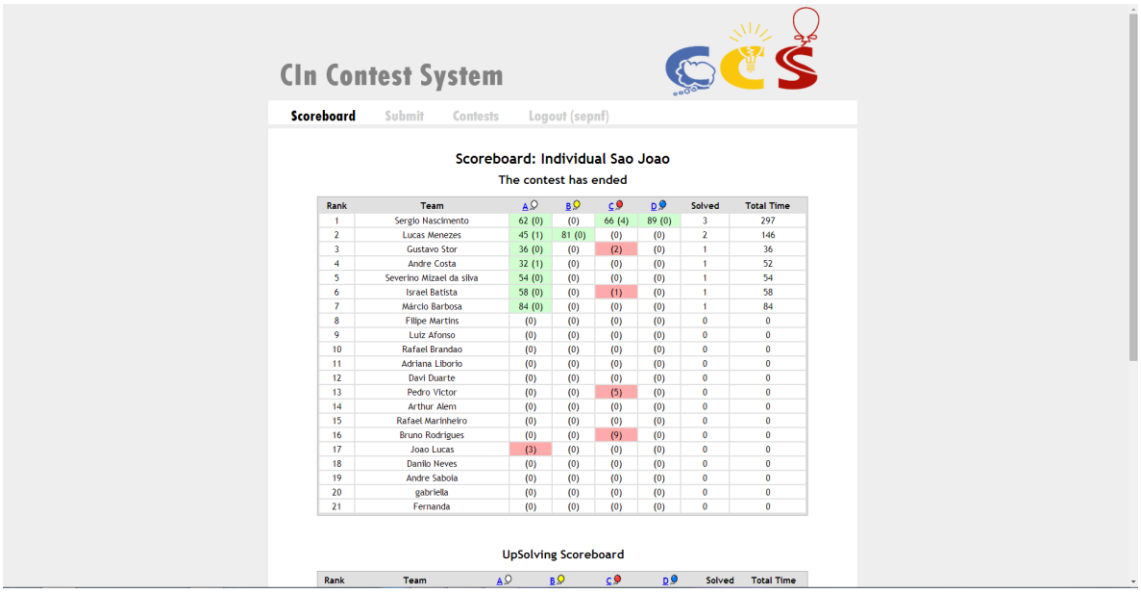

Figura 1: Interface do Cin Contest System.

O CCS contém ainda várias funções avançadas de simuladores conhecidos mundialmente, como o A2 Online Judge, citado anteriormente:

- Na escolha de questões, é possível setar um contest inteiro facilmente, escolher várias questões por vez ou escolher questões individualmente.
- Na atualização do placar, o CCS contém submitors, ferramentas que recebem os códigos dos alunos e submetem nos online judges, para várias linguagens.

- Para acompanhamento do interesse dos alunos, após o contest é aberto o upsolving, que permite aos alunos submeterem as questões não resolvidas no contest da semana.
- Para acompanhamento do crescimento de rendimento dos alunos, o CCS conta também com um sistema de classificação de usuários, que utiliza o consagrado algoritmo ELO de geração de ratings para atualizar o valor atribuído a cada competidor, depois de cada contest.

O sistema possui ainda com três modos de permissão diferentes: Usuário, setter e administrador. Como usuário, o aluno pode apenas participar de competições existentes onde ele foi selecionado e submeter questões, durante competições ou em upsolving. Como setter, o usuário do sistema pode apenas escolher contests e colocar usuários em contests. Como administrador, o usuário pode fazer tudo o que um setter faz, mas tem também acesso aos dados de rankings, a criação de rankings e ao gerenciamento de usuários (inserção, deleção e modificação de permissões).

A ferramenta proposta nesse projeto está desenvolvida de forma integrada com o Cin Contest System, de forma que ela é capaz de identificar os usuários de um dado contest, tal como seus níveis e, após seleção de questões, as coloca automaticamente no contest.

4. Ferramenta

Nesse capítulo, apresentamos a ferramenta proposta no projeto. A ferramenta tem como objetivo ser de simples uso e prover uma seleção avançada de questões, de acordo com o nível dos usuários.

São apresentados detalhes sobre a arquitetura do sistema, a arquitetura do servidor e explicações sobre cada módulo do sistema.

4.1 Arquitetura do Servidor

Devido ao projeto ser criado pensando na utilização do time de maratona de programação da UFPE, o servidor em que o mesmo está hospedado é http://maratona@maratona.cin.ufpe.br. Esse é o mesmo servidor que hospeda o Cin Contest System.

Esse servidor, providenciado pelo centro de informática da UFPE, e é disponibilizado com os softwares constituintes de LAMP. LAMP é uma combinação de softwares livres e de código aberto, cujo acrônimo significa Linux, Apache, MySQL (ou MariaDB) e PHP (ou Perl, ou Python). LAMP é uma solução completa para desenvolvimento de aplicações web e sites dinâmicos. [4]

O Linux atua na plataforma LAMP como sistema operacional, que suporta todos os outros serviços e gerencia os módulos de rede. No nosso servidor, em específico, a distribuição do Linux utilizada é Ubuntu Server 11.10, com versão do kernel 3.0.0-32-server.

O Apache é o servidor web, que é responsável por gerenciar e responder às requisições HTTP feitas ao servidor, sempre que a página é aberta. A ferramenta é responsável também por entender qual página o usuário deseja abrir e por executar os scripts PHP invocados na página, para que ela possa

ser interpretada e então mostrada pelo browser. No nosso servidor, a versão do apache utilizada é a 2.2.20.

O MySQL é a ferramenta responsável pelo gerenciamento de banco de dados, provendo acesso de forma eficiente às informações necessárias para funcionamento do sistema. No servidor do time UFPE, a versão do MySQL utilizada é a 5.5.37.

PHP entra como linguagem de programação web, para prover a possibilidade de criação de páginas dinâmicas e que possa carregar dados diferentes, de acordo com qual usuário está utilizando-a, e com a interação deste com o sistema. A versão utilizada do PHP no servidor é a 5.3.6.

Contamos ainda, no nosso sistema, com Python, utilizado como linguagem de programação para o back-end do projeto, na camada lógica. A versão do interpretador python no servidor é a 2.7.2.

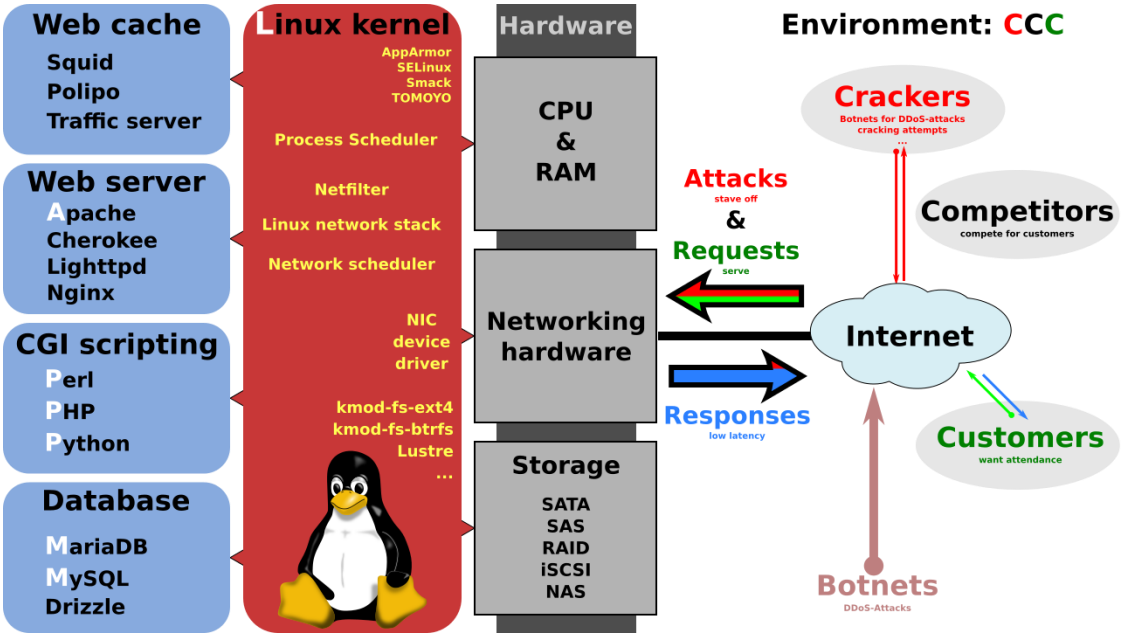

Figura 2: Arquitetura de um servidor web LAMP.

4.2 Arquitetura do Sistema

O sistema foi desenvolvido utilizando-se de um modelo três camadas, também conhecido como 3-Tier. As camadas utilizadas foram:

- Camada de apresentação: Responsável por comunicar-se com o usuário final do sistema. Essa camada foi desenvolvida utilizando-se HTML e CSS para interface, e PHP para comunicação com a camada lógica.
- Camada lógica: Responsável pela obtenção dos dados e manipulação dos mesmos. Comunica-se com a camada de apresentação utilizando-se da camada de dados. Para desenvolvimento do software contido nessa camada, foram utilizadas as linguagens de programação python e C++, sendo a última utilizada apenas quando eficiência é um fator chave.
- Camada de dados: Responsável por armazenar os dados obtidos pela camada lógica, e recuperação eficiente dos mesmos quando necessário. Nessa camada, foi utilizado MySQL como software de banco de dados, que se utiliza da linguagem de programação SQL.

A divisão em camadas foi escolhida pois facilita a organização e manutenção do software, posteriormente. Também facilita a modificação, caso no futuro alguém deseje melhorar o sistema.

Cada camada consiste de um ou mais módulos, sendo um módulo uma parte indivisível do sistema. Cada módulo pertence a exatamente uma camada, mas pode se comunicar com mais que uma delas.

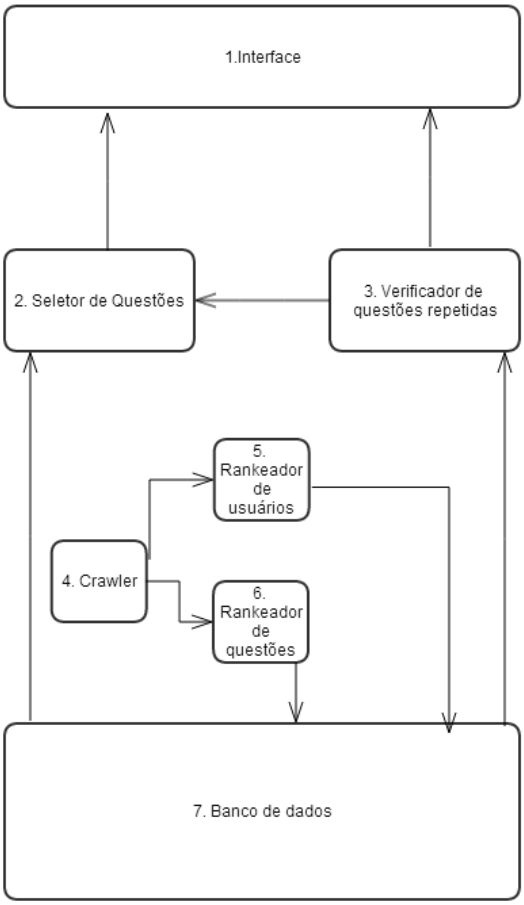

Figura 3: Diagrama de módulos do sistema

O módulo um é o que contém toda a camada de apresentação do sistema, enquanto os módulos 2, 3, 4, 5 e 6 pertencem à camada lógica e o módulo sete contém a camada de dados.

Os módulos 4 5 e 6 contêm duas lógicas diferentes, de acordo com o online judge escolhido na interface para que o contest seja escolhido. Essas diferenças serão analisadas no estudo dos módulos.

4.3.1 Interface

O sistema conta com interface simples, porém completa em termos de acesso de opções pelo usuário. O acesso da interface do sistema se dá a partir da página de edição de contest do Cin Contest System, abaixo do módulo que verifica se existem questões que alguém já fez no contest.

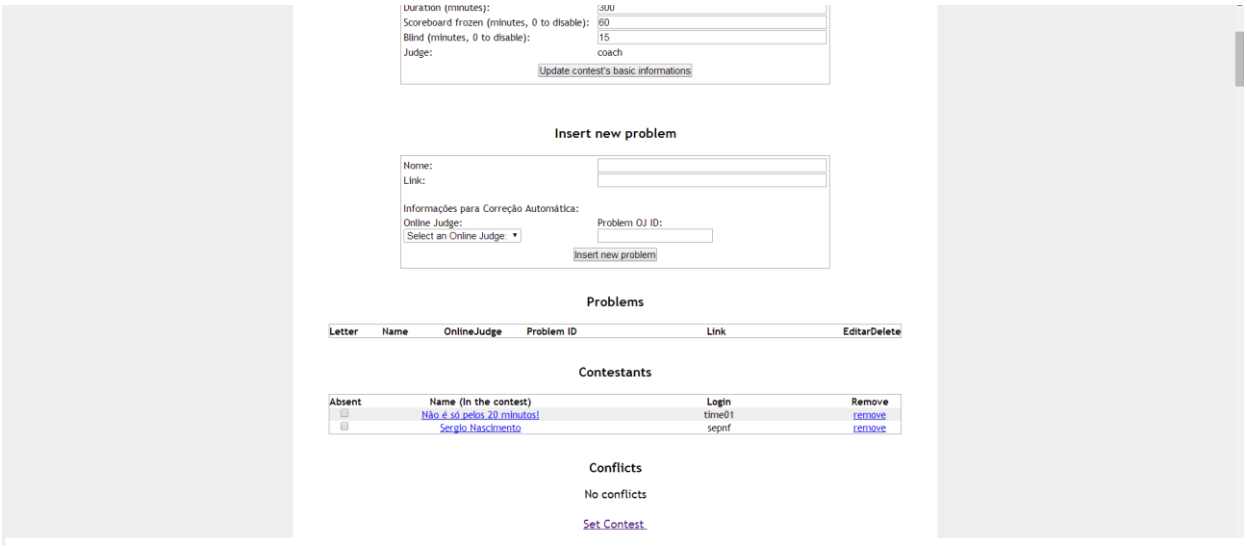

Figura 4: Interface de edição de contest do Cin Contest System

A partir dessa página, clicando em “Set Contest”, o usuário é direcionado a página principal do sistema.

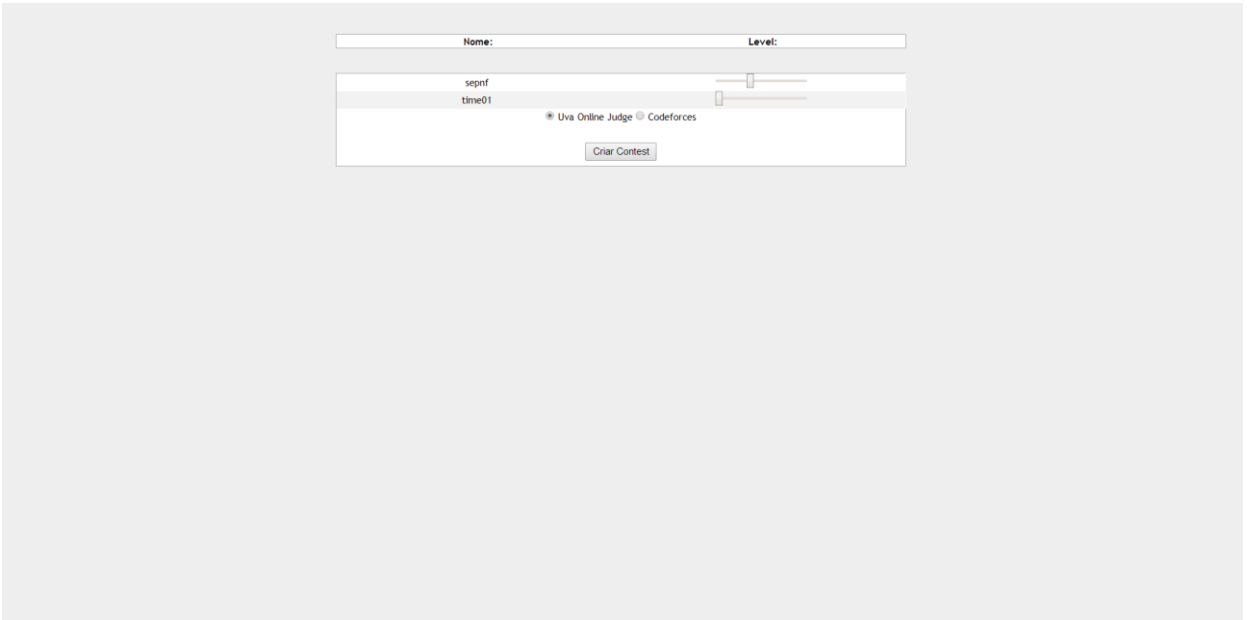

Figura 5: Interface principal do sistema

Nessa página, o setter é capaz de modificar o nível de todos os usuários que pertencem ao contest, e também de selecionar de qual online judge o contest será escolhido, Uva Online Judge ou Codeforces.

Após clicar em create contest, o usuário é automaticamente redirecionado para a página de edição de contests, e o contest já terá sido escolhido e integrado ao sistema.

4.3.2 Seletor de questões

O seletor de questões recebe informações da camada de dados das questões, junto com seus níveis de dificuldade, e do nível dos usuários que pertencem ao contest a ser treinado. Com base nesses valores, escolhe as questões que se enquadrem com o nível dos usuários e que não tenham sido resolvidas por nenhum deles. O sistema realiza isso se utilizando do verificador de questões repetidas. O módulo sempre escolhe um número aleatório de questões, entre 10 e 12, para que possa escolher o contest com esse número de questões.

Para seleção de questões, foi utilizado um algoritmo simples, que consiste em minimizar o módulo da média das distâncias do nível dos usuários para as questões, que é o mesmo que minimizar a soma das distâncias, já que o número de usuários de um dado contest é conhecido. Para todo o texto seguinte, n representa o número de usuários no contest e p o número de problemas.

$$\sum_{i=0}^{n}\sum_{j=0}^{p}(user[i] - question[j])$$

Equação 1: Equação a ser analisada pelo algoritmo de seleção de questões.

Um valor positivo nessa equação significa que o contest está mais fácil que o nível do grupo, enquanto um valor negativo significa que ele está mais difícil que o que o grupo espera. Sendo assim, o objetivo é deixar o valor dessa equação tão próximo de zero quanto possível.

A equação é similar a equação de regressão linear do método dos mínimos quadrados, problema clássico de otimização com solução bem definida. Porém, pelo problema ser de minimização das distâncias lineares, ao invés de minimização de distâncias euclidianas, o problema se torna mais simples, pois matematicamente sempre existem infinitas soluções onde o somatório iguala-

se a zero. Esse fato pode ser provado decorrendo do fato que o somatório é o mesmo que:

$$p * \sum_{i=0}^{n} user[i] - n * \sum_{i=0}^{p} question[i]$$

Equação 2: reescrita da equação 1, de forma que fica evidente como resolver a equação.

Para igualar esse somatório a zero, é sempre possível fazer com que todas as questões estejam tão próximas quanto possível da média aritmética do nível dos usuários, já que é um fato conhecido, e trivialmente provado que:

$$\sum_{i=0}^{n} (user[i] - mean) = 0$$

Equação 3: Somatório de usuários menos média dos níveis dos usuários.

Porém, por motivos técnicos o contest citado acima é inviável, pois seria um treino onde todas as questões teriam o mesmo nível. Isso desestimularia os mais novos no projeto, pois as questões seriam difíceis demais para eles, enquanto não seria um treino adequado para os mais experientes, já que todos os problemas seriam fáceis demais para eles.

Também é fácil provar que, se os níveis forem equidistantes da média aritmética dos usuários, o somatório ainda tende a zero. Sendo assim, é possível construir uma curva normal com média igual a mean, definido acima, e desvio padrão escolhido arbitrariamente, e utilizar a mesma para escolher aleatoriamente as questões do contest.

O método definido acima não garante que o somatório dos níveis será igual a zero, porém garante que a esperança matemática do valor de um contest é igual a zero. Consideramos que a variação de nível dos treinos é positiva para o grupo, sob o argumento de que manter o nível dos treinos sempre igual deixaria os treinos previsíveis e monótonos para os competidores, que resolveriam o mesmo número de problemas toda semana.

Como mostrado anteriormente, a escolha de qualquer desvio padrão é satisfazível para o problema. Sendo assim, utilizamos o desvio padrão como a diferença entre a média aritmética e o valor do usuário melhor ranqueado.

O valor foi escolhido pois aproximadamente 68% dos valores de uma distribuição normal estão na faixa entre [média – desvio padrão, média + desvio padrão]. Como 32% dos valores estão fora dessa faixa, e a normal é simétrica em relação à média, decorre disso que, em média, 16% das questões estarão acima do nível de todos os usuários, e não serão resolvidas durante a competição. Para um contest com 12 questões, esse valor resulta em 1.92, ou seja, em média duas questões não serão resolvidas.

4.3.3 Verificador de questões repetidas

O verificador de questões repetidas busca da camada de dados as informações sobre quais questões já foram resolvidas por cada usuário de uma lista, realiza a união dos conjuntos de questões resolvidas e verifica se uma dada questão está ou não no conjunto criado.

4.3.4 Crawler

O crawler é um módulo dependente de entrada externa, e por isso teve que ser desenvolvido separadamente para o Uva Online Judge e para o Codeforces. Devido à diferença no tamanho das páginas, foi necessária a extração de um número diferente de informações de cada uma delas.

No crawler do Codeforces, são extraídos os dados de todas as submissões realizadas pelos usuários do site. Essa extração foi desenvolvida através do download de todas as submissões das páginas de competições do site. O download das páginas do codeforces foi feito em C++, utilizando-se da ferramenta wget, disponível no Linux, para maior velocidade no download das páginas.

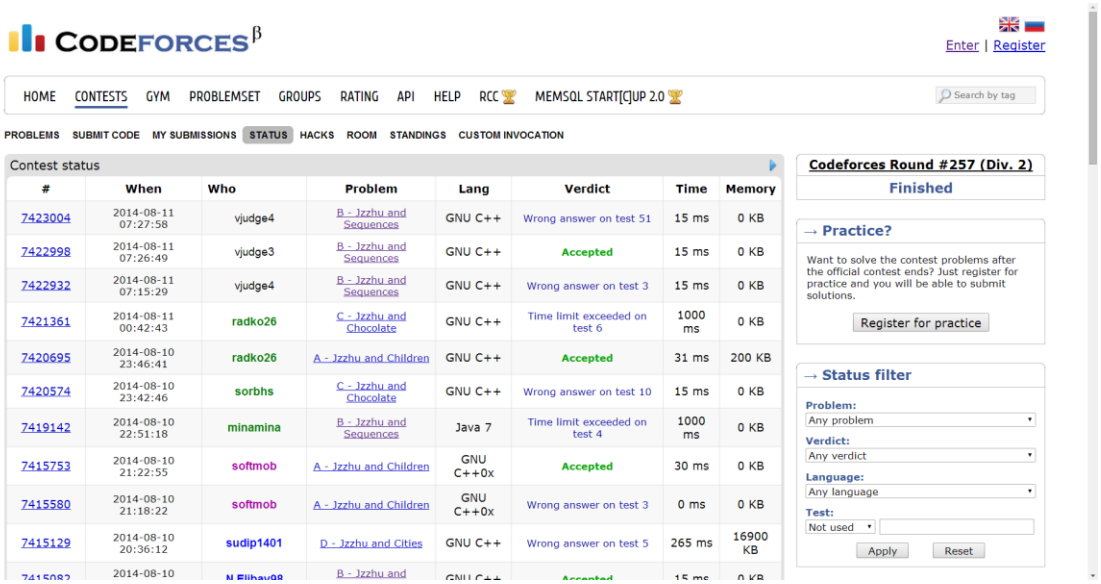

Figura 6: página de submissões do codeforces, baixada pelo crawler

No crawler do Uva Online Judge, porém, são extraídos apenas dados mais básicos das questões. Esses dados consistem de ID da questão, número de vezes que a mesma foi resolvida e percentual de soluções aceitas para o problema. O download das páginas do Uva Online Judge foi feito em python, utilizando-se da biblioteca urllib2.

A escolha de escrever um programa diferente da solução do codeforces deriva-se de dois fatores: O fato do volume de dados ser muito menor, e o fato do site Uva Online Judge ser mais instável. Devido ao último, o uso de uma biblioteca pronta minimiza a chance de erros que possam parar o processo.

A escolha de extrair menos dados foi feita porque, após cálculos simples, notamos inviabilidade da extração da mesma quantidade de dados do Codeforces, já que o Uva Online Judge é um site de maior porte.

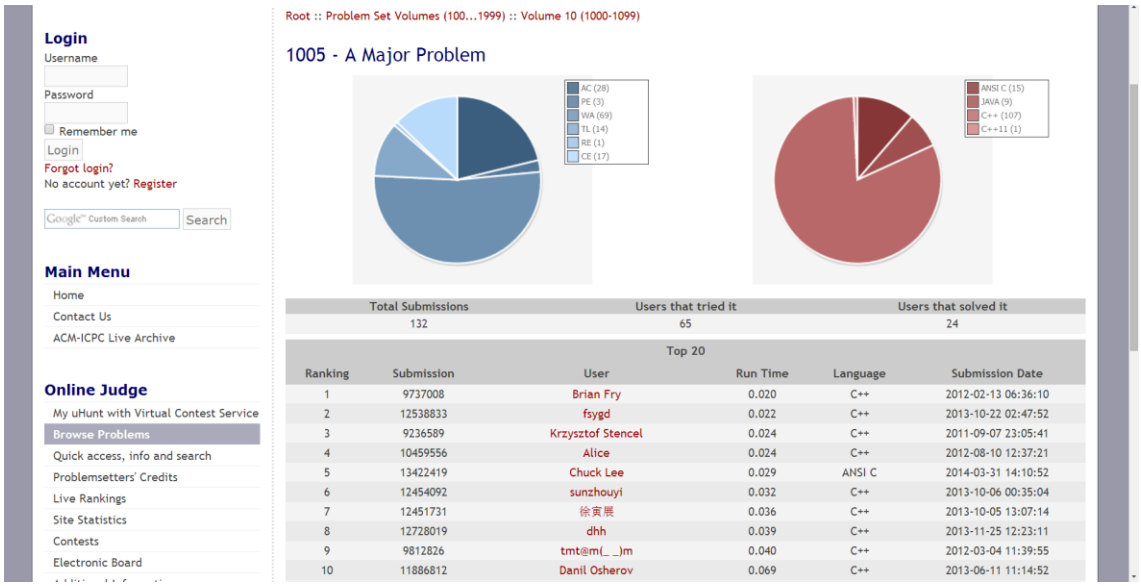

Figura 7: página de submissões do Uva Online Judge, baixada pelo crawler.

Após download das páginas mostradas, ocorre o parsing apropriado para que possam ser extraídos os dados necessários. Todo o parsing foi feito em python, utilizando-se de uma biblioteca chamada BeautifulSoup para gerar a árvore de hierarquia do HTML da página. Tendo o HTML montado como uma árvore, é possível acessar a informação necessária apenas com um passeio na mesma.

4.3.5 Classificador de usuários

O classificador de usuários utiliza-se dos dados das questões classificadas do Uva Online Judge para classificar os usuários. A escolha foi realizada pois os estudantes da UFPE contém, em média, mais questões resolvidas no Uva Online Judge que no Codeforces, assim, conseguimos uma melhor classificação quando utilizado o primeiro.

O classificador utiliza-se de uma média ponderada similar a utilizada no ranqueador de questões do codeforces, onde os pesos decaem como uma exponencial negativa, e as questões mais difíceis resolvidas pelo usuário ganham peso maior.

$$level(x) = \frac{1}{\sum_{i=0}^{n(p)} e^{-i/n(p)}} \sum_{i=0}^{n(p)} e^{-i/n(p)} * level(problem(i))$$

Equação 4: Regra de classificação de usuários no sistema

4.3.6 Classificador de questões

Devido aos crawlers do codeforces e uva buscarem diferentes quantidades de informação, foram necessárias estratégias diferentes para classificação das questões dos dois online judges.

Para atribuição de notas às questões do codeforces, foi utilizado um método de avaliação de documentos conhecido como HITS, enquanto para classificação das questões do Uva Online Judge foi utilizado um método estatístico simples.

Os dois métodos serão estudados com maior profundidade em tópicos separados.

4.3.6.1 Classificador do Codeforces

O classificador do codeforces se utilizou do algoritmo HITS para classificação das questões. O algoritmo HITS pertence à vasta área de análise de dados, mais especificamente se incluindo na subárea de análise de links.

4.3.6.1.1 Análise de dados

Análise de dados é o processo de inspecionar, limpar, transformar e modelar dados com o objetivo de descobrir informações úteis, sugerir conclusões e suportar processos de tomada de decisão. [5] A área surgiu a partir do estudo de integração de dados, que consiste em capturar informações de diferentes fontes e agregá-las para melhor análise do usuário. É correlacionada com as áreas de visualização de dados e disseminação de dados. A essa área, pertencem as subáreas:

- Mineração de dados, que foca em inferência de novo conhecimento a partir de dados bem definidos.
- Business intelligence, utilizada para tomada de decisão em modelos comerciais, onde quantidades enormes de dados são considerados e agregados.
- Análise de links, que foca na inferência de informações a partir de grafos conhecidos.

Análise de dados se divide em duas partes: Análise de dados exploratória, que funciona inferindo novas informações a partir dos dados conhecidos, e Análise de dados confirmatória, que funciona confirmando ou negando hipóteses dadas, dado um conjunto de informações. [5] O algoritmo HITS, estudado nesse capítulo, pertence à área de análise de dados exploratória.

4.3.6.1.2 Análise de Links

Análise de links é uma subárea de análise de dados, responsável por avaliar conexão entre nós de um grafo. A área é utilizada para investigação de análises criminais (como detecção de fraudes, contraterrorismo e inteligência), análise de segurança de sistemas computacionais, otimização de engenhos de busca, pesquisa de mercado e pesquisa médica. [6]

Embora o problema a ser resolvido seja o mesmo para todos os algoritmos da área, os algoritmos se comportam de formas diferentes para grafos diferentes. O grafo da internet, por exemplo, se comporta como um grafo aleatório, quando a função de distribuição de probabilidade é uma função log-normal.

4.3.6.1.3 Algoritmo HITS

O algoritmo HITS é um algoritmo de análise de links que opera sobre grafos bipartidos. O algoritmo foi criado para dar notas de importância a páginas web, sendo o precursor do pagerank. [7]

O algoritmo foi largamente utilizado para dar notas de importância a periódicos científicos, pois para a comunidade acadêmica, ser citado por um artigo importante tem mais valor que ser citado por outro artigo, menos importante. Além disso, até hoje, o Twitter utiliza-se do algoritmo para sugerir aos usuários que outros usuários eles deveriam seguir. [7]

No grafo bipartido, uma das partições é chamada de “authority” e a outra é chamada “hub”. A ideia dos nomes é que a internet é dividida em duas partes: Uma que aponta para muitas páginas, como engenhos de busca e sites do tipo portal de informações, e outra que é apontada por muitas páginas, como sites de referência em área específica. No caso da sua utilização focada na internet,

cada site é dividido em dois nós, um hub e um authority, e o nó é classificado de acordo com seu nó de maior valor final.

O algoritmo funciona como um processo iterativo, onde cada nó começa com valor um em seu peso. Após isso, os nós são atualizados por recursão mútua, processo onde dois objetos computacionais são definidos em termos um do outro. [8]

A atualização do processo se dá via duas regras transição: Regra de atualização de authority e regra de atualização de hub. Ambas podem ser modificadas de acordo com o propósito do algoritmo. No algoritmo proposto originalmente, essas regras são:

$$auth(p) = \sum_{i=0}^{n(p)} hub(adj[p][i])$$

Equação 5: regra padrão de atualização de authorities.

$$hub(p) = \sum_{i=0}^{n(p)} auth(adj[p][i])$$

Equação 6: regra padrão de atualização de hubs.

A intuição sobre as regras é que, caso existam hubs importantes apontando para uma determinada página, ela deve ser um authority importante. Analogamente, quanto mais authorities importantes apontarem para um determinado hub, melhor a qualidade do hub escolhido. Isso não necessariamente é verdade para páginas da internet, e por isso o algoritmo não é vastamente utilizado para classificação das mesmas.

A atualização deve ser feita de forma que, apenas após as duas regras serem aplicadas, os valores dos authorities e dos hubs são atualizados, ou seja, tanto os authorities quanto os hubs utilizam-se dos valores do passo x para gerar o valor do passo x+1. Em nível de exemplo, é demonstrada a execução do algoritmo para um grafo gerado aleatoriamente, exemplificando os primeiros passos do algoritmo.

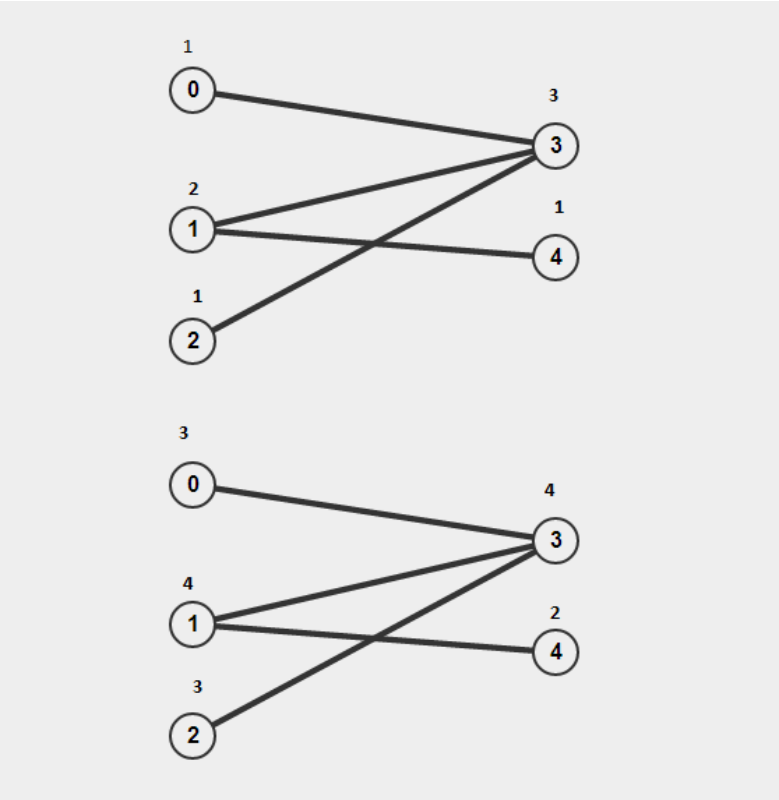

Figura 8: Primeiros passos na execução do algoritmo HITS.

Da forma que as regras são apresentadas, os valores divergem em escala exponencial, tornando inviável a execução do algoritmo. Assim, após cada passo da execução do algoritmo, é recomendado que os valores sejam normalizados.

Com a normalização dos valores, o algoritmo tem convergência garantida para um número grande o suficiente de passos. Empiricamente, encontramos que esse valor cresce na ordem de log(n).

4.6.3.1.4 O classificador

Para classificação de questões, foi montado um grafo bipartido onde em uma partição estão as questões, e na outra estão os usuários do Codeforces. Após isso, o algoritmo HITS foi aplicado ao grafo.

As métricas escolhidas para o algoritmo HITS prever foram nível de dificuldade dos problemas, e nível de habilidade dos usuários. Devido à natureza das métricas escolhidas, as regras de atualização do algoritmo foram modificadas, já que uma questão é mais difícil se foi resolvida por menos usuários, e um usuário é bom apenas se já resolveu questões difíceis (quantidade de questões fáceis é irrelevante).

Para simplicidade na implementação do algoritmo, e para garantia de sua convergência após certo número de passos, foi utilizada a mesma função para ambos, atualização de usuários e atualização de problemas. Anteriormente a execução da função, os nós adjacentes são ordenados por dificuldade.

$$node(p) = \sum_{i=0}^{n(p)} e^{-i/tam} * value[node(i)]$$

Equação 7: Regra para atualização de nós no classificador do codeforces

O uso de uma exponencial negativa para dar pesos aos nós foi decidido de forma empírica, pois foi a equação que demonstrou melhor decaimento de velocidade, entre todas as testadas.

Após vinte passos, o algoritmo apresenta convergência de valores. Embora o algoritmo apresente convergência para número logarítmico de passos como esperado, os valores convergem para um range entre [0.90, 1.0], que é indesejado, pois o módulo de seleção de problemas classificaria todas as questões como difíceis. Com isso, é necessária uma renormalização dos valores, para que eles se encontrem num range mais amigável, após execução do algoritmo.

Após renormalização, o sistema apresenta resultados que comprovam a sua eficiência, sendo demonstrado que o sistema conseguiu clusterizar as questões A, B, C, D e E em níveis diferentes, tendo a D e a E níveis parecidos devido à baixa quantidade de usuários que resolvem questões de ambos os níveis.

	A	B	C	D	E
Média	0.36	0.40	0.45	0.49	0.49
Variância	0.01	0.01	0.01	0.01	0.01

Tabela 1: dados estatísticos sobre o classificador do codeforces

Pode ser identificada a presença de convergência dos valores devido à baixa variância em todos os níveis. O fato de todos os níveis se encontrarem em um pequeno range, de [0.36, 0.49], não tem explicação concreta, mas

acreditamos que o fato se dê devido à existência de contests no formato ACM ICPC no site do codeforces. Esses contests contém maior variação de nível, e provavelmente suas questões ocupam as extremidades dos níveis.

A partir da saída do algoritmo, foi realizada uma análise estatística sobre os dados, sendo assim possível notar que o número de pessoas que resolveram uma dada questão é um fator de alta correlação com a dificuldade da questão. Foi também analisado que o número de submissões erradas em problemas tem baixa correlação com a dificuldade da questão. Esse conhecimento foi de grande utilidade para implementação do classificador de questões do Uva Online Judge, examinado a seguir.

4.3.6.2 Classificador do Uva Online Judge

Devido ao Uva Online Judge ter maior porte que o Codeforces, tanto em número de problemas quanto em número de usuários, baixar todo o grafo de questões x usuários do Uva se torna inviável sem um servidor dedicado. Sendo assim, o classificador deste se baseia apenas no número de usuários que resolveram as questões.

Após vários testes, novamente encontramos que uma exponencial negativa aproximaria melhor o conjunto de questões. Após verificação do comportamento do algoritmo HITS, foram criados pares (número de usuários, dificuldade) que seriam esperados pelo classificador.

Com esses pares, foi realizada uma busca ternária manual para encontrar o valor do tempo da exponencial que melhor se adequaria aos pares criados anteriormente. O resultado é a seguinte função:

$$difficulty(x) = e^{-qt(x)/500.0}$$

Equação 8: Função de atribuição de dificuldade para problemas do Uva Online Judge

O valor 500 é o tempo da exponencial, e significa que uma questão resolvida por 500 usuários recebe dificuldade de aproximadamente 0.36, numa escala entre 0 e 1.

Segue uma tabela de valores de dificuldade de acordo com número de pessoas que resolveram o problema.

Nº soluções	0	20	100	500	1000	2000
Dificuldade	1.0	0.96	0.81	0.36	0.13	0.01

Tabela 2: Número de soluções para problemas x dificuldade atribuída

Recebemos opinião de membros do time UFPE de maratona de programação, e os membros consultados acharam esses valores plausíveis com a realidade. Não existe método para validação formal dos valores, já que não existe classificação oficial das questões.

4.3.6.3 Comparação entre métodos de classificação

Ambos os métodos trouxeram resultados satisfatórios, e são capazes de realizar boa escolha de competições. O método do Uva Online Judge, por consistir apenas de um modelo matemático, ganha em eficiência, pois é capaz de classificar grandes quantidades de informações, já que sua execução é O(Q), sendo Q o número de questões.

O método do Codeforces, por sua vez, tem sua execução em O(n * E), sendo n o número de passos do algoritmo e E o número de arestas do grafo dado. Como o grafo converge em O(log(E)) passos, podemos estimar a complexidade do método em O((V+E) * log(E)), sendo polinomial no tamanho do grafo. O método tem custo consideravelmente maior que o do Uva Online Judge, pois caso o grafo seja denso o número de arestas é quadrático em relação ao número de problemas.

Em qualidade geral de classificação, os dois métodos apresentam resultados de nível semelhante. O método do codeforces, porém, é capaz de classificar “casos de borda”, que são questões onde, ambas foram resolvidas por número parecido de usuários, porém os usuários que resolveram uma tem nível muito maior que aqueles que resolveram outra. Isso é comum quando uma das questões é muito mais antiga que a outra. O classificador do Uva Online Judge apenas classifica ambas as questões com o mesmo nível, ignorando a informação de quem as resolveu. Os casos de borda, porém, são raros, e seus efeitos podem ser ignorados no estudo estatístico dos dados.

4.3.7 Banco de Dados

O banco de dados do sistema é relacional, implementado utilizando-se do sistema de gerenciamento de banco de dados MySQL, serviço gratuito e open source que utiliza SQL como linguagem de programação.

No banco de dados do sistema, existe uma tabela implementada especificamente para esse projeto e uma tabela já existente no sistema modificada para adequar dados da plataforma.

A tabela incluída no banco de dados tem nome Questao_level, e consiste de quatro colunas: id, id_questao, oj e nivel.

- Id é um valor inteiro com auto incremento, para que a recuperação da informação possa acontecer de forma mais eficiente.
- Id_questao representa o identificador único da questão no online judge específico. No caso do Uva Online Judge, o id é sempre um inteiro, enquanto no codeforces o ID é uma string. Por isso, foi escolhido o tipo varchar para id_questao.
- Oj é um varchar que identifica o online judge da questão
- Nivel é um valor de ponto flutuante (double), que representa o nível normalizado do problema, entre zero e um.

Além dessa tabela, na tabela Competidor foram incluídos:

- Um double, com nome “level”, que representa o nível do usuário normalizado entre zero e um.
- Campos de handle_uva, handle_la, handle_spoj e handle_codeforces, para verificação de questões repetidas nos online judges.

A tabela de questões que cada usuário do sistema já resolveu foi julgada um dado muito pequeno para justificar a criação de uma tabela, e esta é mantida em um arquivo de texto pleno.

5. Conclusão

Neste trabalho, foi analisado o problema de seleção de questões para programação competitiva, estudadas ferramentas existentes e como elas solucionam o problema, e proposto um sistema que resolve a questão de forma inovadora, em relação aos seus concorrentes.

O crescimento da maratona de programação no Brasil ocorreu de forma superlinear nos últimos anos, e a metodologia de treinamento das universidades evoluiu junto com o crescimento do projeto. Com metodologia mais aguçada de treinamento, os alunos tendem a se dedicar cada vez mais ao projeto. Como os alunos estão dedicando cada vez mais tempo para esse importante projeto, é importante também que eles aproveitem os seus tempos de forma ótima.

Embora as plataformas atuais tenham uma classificação mais acurada que o projeto proposto, já que são seleções manuais, os alunos mais experientes da maratona de programação já resolveram grande parte das questões compiladas nas outras ferramentas, e quanto acabam as listas, precisam resolver problemas realizando escolhas pior fundamentadas que as do sistema implementado neste projeto.

Como trabalho futuro, é sugerido o uso de técnicas de inteligência artificial, como aquelas utilizadas em sistemas de recomendação. Assim, o classificador de questões poderá ter seu funcionamento melhorado de acordo com o feedback dos usuários do sistema sobre os problemas. Um estudo de viabilidade sobre esse módulo seria necessário, já que com uma base pequena de usuários muitos algoritmos nos quais sistemas de recomendação são baseados não funcionam corretamente.

Ainda como trabalho futuro, sugerimos um módulo similar aos atuais, só que podendo escolher contests antigos completos. Esse módulo é útil já que, quando treinando em time, a simulação de contests completos é uma experiência muito mais valiosa que a escolha de questões avulsas de contests. Para isso, seria necessária a integração do sistema atual, que suporta Uva

Online Judge e Codeforces, com o Live Archive, online judge especializado em contests antigos completos. Devido a modularização do projeto, e a semelhança da página do Live Archive com a página do Uva Online Judge, tal integração se torna simples, sendo necessário apenas que o interessado nessa implementação estude o código e o compreenda.

Referências

1. A2 Online Judge, A2 Online Judge. Disponível em: <http://a2oj.com/>. Acesso em 7 de agosto.

2. Uva Toolkit. Uva Toolkit. Disponível em: < http://uvatoolkit.com/>. Acesso em 7 de agosto.

3. uHunt. uHunt. Disponível em:< http://uhunt.felix-halim.net/>. Acesso em 7 de agosto.

4. WIKIPEDIA. LAMP (Software Bundle). Disponível em < http://en.wikipedia.org/wiki/LAMP_(software_bundle)>. Acesso em 8 de agosto.

5. WIKIPEDIA. Data-analysis. Disponível em < http://en.wikipedia.org/wiki/Data-analysis>. Acesso em 8 de agosto.

6. WIKIPEDIA. Link-analysis. Disponível em < http://en.wikipedia.org/wiki/Link_analysis>. Acesso em 8 de agosto.

7. WIKIPEDIA. HITS algorithm. Disponível em < http://en.wikipedia.org/wiki/HITS_algorithm>. Acesso em 8 de agosto.

8. WIKIPEDIA. Mutual Recursion. Disponível em < http://en.wikipedia.org/wiki/Mutual_recursion>. Acesso em 8 de agosto.