Trabalho Prático 1
Um Novo Sistema de Seleção Unificada

Algoritmos I

Entrega: 16/04/2019

1 Introdução

O Sistema de Seleção Unificada (Sisu) é o sistema informatizado, gerenciado pelo Ministério da Educação (MEC), pelo qual instituições públicas de educação superior oferecem vagas a candidatos participantes do Exame Nacional do Ensino Médio (Enem). Ao efetuar a inscrição, um candidato c_i com uma nota n_i apresenta uma lista de preferência L_i de até duas opções entre as universidades disponíveis no Sisu. Cada universidade u_j possui uma quantidade máxima de vagas q_j e uma nota mínima de corte nc_j (a nota mínima que o aluno deve ter para poder ser aceito), além disso, todas as universidades ordenam os candidatos pela suas respectivas notas, ou seja, as universidades tem interesse maior em alunos com notas maiores.

No atual processo de seleção do Sisu, é possível mudar as opções de curso no decorrer da semana, com base nas notas de corte parciais que são divulgadas por cada universidade. Ao final da etapa de inscrição, o sistema seleciona automaticamente os candidatos mais bem classificados em cada curso, de acordo com suas notas no Enem. O candidato selecionado em sua primeira ou segunda opção só terá uma oportunidade de fazer sua matrícula, já que, independentemente de efetuar ou não sua matrícula na instituição de ensino, não poderá manifestar interesse em participar da lista de espera de outra instituição. Após a chamada regular do processo seletivo, o Sisu disponibiliza às instituições participantes uma lista de espera com os candidatos com uma nota superior a nota de corte para ser utilizada para o preenchimento das vagas eventualmente não ocupadas.

Esse processo de admissão apresentado pelo Sisu pode introduzir vários problemas para os candidatos bem como para as universidades. Supondo que um estudante tenha sido aceito pela sua segunda opção e caso por algum motivo, venha a surgir uma nova vaga para a primeira instituição o estudante ficara inibido de participar da lista de espera. Além disso, o processo atual exige inúmeras rodadas até todos estarem alocados. Essa situação poderia fazer com que o processo seletivo fosse finalizado com candidatos insatisfeitos com a alocação, uma vez que a mudança de opção seria vantajosa tanto para o estudante quanto para a universidade (porque há vagas e o aluno atende o corte mínimo). Com isso, o governo brasileiro está refazendo o sistema de oferta de vagas para o Sisu e pediu a sua ajuda para desenvolver um algoritmo capaz de, dada uma lista de vagas em instituições com uma nota de corte e uma lista de candidatos com uma nota no Enem e com uma lista de prioridade de faculdades definida pelos candidatos, encontrar a melhor alocação para os alunos nessas instituições, satisfazendo as restrições impostas pela nota de cada candidato, o número de vagas e a nota mínima de cada instituição.

2 O que fazer?

O objetivo deste trabalho será distribuir os candidatos entre as vagas de forma a satisfazer ambos os grupos, candidatos e universidades. Considere um processo seletivo no qual n candidatos devem ser designados para m universidades, onde q_i é a quantidade de vagas na i-ésima universidade. Cada candidato informa uma lista de universidades L_i, $0 < L_i \leq m$, por ordem de preferência (sem empates), excluindo apenas aquelas em que não deseja estudar em hipótese alguma. Da mesma forma, cada universidade u_j deve listar os candidatos (dentre os que aplicaram para ela) ordenados pela nota do Enem, excluindo aqueles estudantes que estão abaixo da nota mínima estipulada. A quantidade de vagas na universidade pode ser maior, menor ou igual a quantidade de aplicações.

A condição essencial que deve ser satisfeita para uma designação de candidatos para universidades é que ela não apresente **"instabilidades"**. Em outras palavras, queremos que a satisfação dos candidatos seja a maior possível, respeitando toda essas restrições do problema (ordem das notas dos candidatos, a nota mínima de cada instituição e o número de vagas). Uma alocação será instável e inválida quando:

- Exista um candidato c_1 que esteja alocado para uma universidade α e um candidato c_2 que esteja alocado para uma universidade β mas c_1 prefira a universidade β e c_2 prefira a universidade α
- Um candidato c_1 esteja alocado para uma universidade α, mas há um aluno não alocado que tenha nota maior que c_1 e que tenha marcado α como interesse
- Um candidato está alocado para uma universidade α, entretanto a universidade β tem uma prioridade maior em sua lista, possui vagas livres e ele atende a nota mínima
- Há um candidato sem alocação mas existe vagas em uma das universidades na sua lista e ele atende a nota mínima

Além disso a alocação deve ser **simultaneamente a melhor possível para todos os candidatos**, respeitando as restrições descritas anteriormente. Ou seja, a alocação será ótima do ponto de vista dos candidatos. Todas as universidades ordenam alunos por nota (universidades preferme alunos com notas maiores), logo, em essencia, as preferências de cada universidade são as mesmas, mas as notas de corte são diferentes. Com isso, um aluno que tem uma nota menor que a aceitável não pode ser alocado para aquela universidade. Por fim, um aluno não pode ser alocado para uma universidade que não está em sua lista de prioridades. Será adotada como premissa que, quando outros aspectos forem iguais, os candidatos com menor identificador terá prioridade.

3 Os arquivos de entrada

O problema terá como entrada dois arquivos de texto contendo a relação das universidades e dos candidatos. Os alunos terão que ler os arquivos e armazená-los em uma estrutura de dados para utilizar na resolução do problema. Cada arquivo contém as seguintes informações:

3.1 Universidades

A primeira linha contém um número m inteiro, $0 < m \leq 10^4$, relacionado ao número total de universidades. Em sequência, cada uma das m linhas a seguir deverá contém uma tupla com o número de vagas que a universidade m_i está oferecendo e a nota mínima que a universidade aceita.

3.2 Candidatos

A primeira linha contém um número n inteiro, $0 < n \leq 10^4$, relacionado ao número total de candidatos. Para cada um dos n candidatos as duas seguintes linhas devem ser preenchidas. A primeira contém uma tupla contendo, respectivamente, o número de aplicações n_a que o candidato c_i fará, $0 < c_i \leq m$, e a nota que ele obteve. A segunda linha deve conter uma lista L_i de universidades que o candidato está interessando, sendo a primeira entrada a universidade que o candidato está mais interessado e a última a universidade que o candidato está menos interessado.

Note que há três cenários possíveis: há mais vagas que aplicações, há um número igual de vagas e aplicações ou há menos vagas que aplicações. A seguir está listado um exemplo de entrada e sua respectiva saída.

Arquivo contendo as universidades

```
3                                       // <Número de universidades>
2 70                                    // <Vagas> <Nota-corte> (Universidade 1)
1 80
1 80
```

Arquivo contendo candidatos

```
3                                       // <Número de candidatos>
3 66                                    // <Número de Aplicações> <Nota> (Candidato 1)
2 3 1                                   // <Lista prioridade> (Candidato 1)
2 95
2 3
3 88
1 2 3
```

4 O arquivo de saída

A saída deve ser impressa na tela (*stdout*) e seguir o seguinte padrão: A primeira linha deve conter a string "Grupos com alocacao" seguido de tuplas <Número do Candidato> <Número da Faculdade>, correspondendo ao id do candidato e ao id da universidade que ele foi alocado. Em sequência a string "Candidatos nao alocados" deve ser impressa seguido dos identificadores de cada candidato que não conseguiu nenhuma vaga.

A impressão desses grupos deve ser ordenado pelo identificador do candidato. Lembrando que a identificação ocorre pela posição do candidato ou instituição no arquivo (1 para o primeiro candidato/instituição). Fique atento para imprimir exatamente como foi descrito pois a correção do algoritmo será feita de forma automática.

Saída Esperada

```
Grupos com alocacao
2 2                                    // <Candidato> <Universidade>
3 1
Candidatos nao alocados
1                                      // <Candidato>
```

5 Avaliação Experimental

Para a avaliação experimental o aluno deverá criar entradas com certas restrições para avaliar como cada uma das três métricas seguintes variam em função dos parâmetros de entrada (número de candidatos, número de vagas nas universidades, notas mínimas, etc).

As métricas sugeridas são:

$$\text{Taxa de alocação} = \frac{\text{Vagas Alocadas}}{\text{Total de Vagas}} \tag{1}$$

$$\text{Média da taxa de preenchimento de vagas nas universidades} = \frac{\sum_{i=1}^{m} \frac{vp_i}{vt_i}}{m} \tag{2}$$

onde vp_i é o número de vagas preenchidas na universidade i e vt_i é o número total de vagas na universidade i e m o número de universidades

$$\text{Satisfação dos candidatos} = \frac{\sum_{i=1}^{n} sat(p_i)}{n} \tag{3}$$

$$sat(p_i) = \begin{cases} \dfrac{1}{log_2(p_i + 1)}, & \text{se o candidato foi alocado} \\ 0, & \text{caso contrario} \end{cases}$$

onde p_i é a posição da universidade na lista de prioridade que o candidato i foi alocado e n o número de candidatos

A partir dessas métricas, espera-se que os alunos tirem conclusões sobre o novo sistema construído, como:

- no novo sistema, os alunos estão limitados a escolher apenas duas instituições na sua lista de preferência, aumentar o tamanho dessa lista aumenta a satisfação dos candidatos? Para verificar é necessário executar o algoritmo com entradas diferentes, variando o número máximo da lista de prioridade de cada aluno, e calcular o grau de satisfação em cada execução
- Permitir que as listas de prioridade sejam maiores aumenta a taxa de alocação?
- Como varia a taxa de alocação de acordo com o número total de vagas fornecidas? Quanto mais vagas melhor a taxa de alocação?
- Como varia a taxa de alocação de acordo com o número total de vagas candidatos?

- Como varia a média da taxa de preenchimento de vagas nas universidades de acordo com a nota mínima? E a quantidade de vagas fornecidas?

É recomendado o uso de gráficos e/ou tabelas para visualizar a variação de cada métrica em relação a uma mudança na entrada.

6 O que deve ser entregue

Você deve submeter um arquivo compacto (zip ou tar.gz) no formato **seu_nome_sua_matrícula** via Moodle contendo:

- todos os arquivos do código *.c* e *.h* que foram implementados,
- um arquivo *makefile* **que crie um executável tp1**,
- sua documentação.

Sua documentação deve ter até 10 páginas contendo:

- uma breve descrição do problema,
- explicações das estruturas de dados e dos algoritmos utilizados para resolver o problema. Para tal, artifícios como pseudo-códigos, exemplos ou diagramas podem ser úteis. Note que documentar uma solução não é o mesmo que documentar seu código. Não inclua textos de códigos em sua documentação.
- análise de complexidade da solução proposta (espaço e tempo). Cada complexidade apresentada deverá ser devidamente justificada para que seja aceita.
- avaliação experimental.

O seu TP deverá ser entregue de acordo com a data especificada no moodle. A penalidade em porcentagem para os TPs atrasados é dada pela fórmula $2^{d-1}/0.16$.

7 Implementação

7.1 Linguagem, Ambiente e Parâmetros

O seu programa deverá ser implementado na linguagem **C** e poderá fazer uso de funções da biblioteca padrão da linguagem. Trabalhos que utilizem qualquer outra linguagem de programação e/ou que façam uso de bibliotecas que não a padrão não serão aceitos. Além disso, certifique-se que seu código compile e execute corretamente nas máquinas **Linux** dos laboratórios do **DCC**.

Os arquivos da entrada devem ser passados como parâmetros para o programa através da linha de comando (e.g., *$ ./main universidades.txt candidatos.txt*)

7.2 Testes

A sua implementação passará por um processo de correção automática, o formato da saída de seu programa deverá ser idêntico aquele descrito nas seções anteriores. Saídas diferentes serão consideradas erro para o programa. Para auxiliar na depuração do seu trabalho, será fornecido um pequeno conjunto de entradas e suas saídas. É seu dever certificar que seu programa atenda corretamente para qualquer entrada válida.

7.3 Qualidade do código

É importante prestar atenção para a qualidade do código, mantendo-o organizado e comentado para não surgir dúvidas na hora da correção. Qualquer decisão que não estiver clara dada a documentação e a organização do código será descontada na nota final.

8 Consideração Final

Assim como em todos os trabalhos dessa disciplina é estritamente proibido a cópia parcial ou integra de códigos, seja da internet ou de colegas. Seja honesto! Você não aprende nada copiando código de terceiros. Se a cópia for detectada, sua nota será zerada e o professor será informado para que as devidas providências sejam tomadas.

Appendices

A Exemplo 1

3
5 70
5 80
1 80

5
2 66
3 1
2 95
3 1
2 88
3 1
2 99
3 1
2 80
3 1

Grupos com alocacao
2 1
4 3
Candidatos nao alocados
1
3
5

B Exemplo 2

3
1 70
5 80
5 80

5
3 76
3 1 2
3 95
3 1 2
3 88
3 1 2
3 99
3 1 2
3 80
3 1 2

Grupos com alocacao
1 1
2 3
3 3
4 3
5 3
Candidatos nao alocados