TP2 - Desemprego

Algoritmos I
Data de entrega: 02/06/2023

1 Objetivo do trabalho

O objetivo deste trabalho é modelar o problema computacional descrito a seguir utilizando estruturas de dados e algoritmos, em particular aqueles vistos na disciplina, que permitam resolvê-lo de forma eficiente.

Serão fornecidos alguns casos de teste bem como a resposta esperada dos casos fornecidos para que o aluno possa testar a corretude de seu algoritmo. Não obstante, recomenda-se que o aluno crie casos de teste adicionais a fim de validar sua própria implementação. A sua solução deve obrigatoriamente ser desenvolvida utilizando algoritmos em grafos e algoritmos gulosos.

O código-fonte da solução e uma documentação sucinta (relatório contendo não mais do que 5 páginas) deverão ser submetidos via Moodle até a data limite de 02/06/2023. A especificação do conteúdo do relatório e linguagens de programação aceitas serão detalhadas nas seções subsequentes.

2 Definição do problema

Willie Gates é o CEO de uma grande rede social chamada LinkOut. No entanto, a rede está enfrentando uma grande crise e ele está buscando maneiras de aumentar o engajamento dos usuários. O principal objetivo da LinkOut é conectar seus usuários às vagas de emprego ideais. Infelizmente, Willie está recebendo muitas reclamações de usuários que não estão sendo recomendados para nenhuma vaga, enquanto seus colegas com qualificações semelhantes estão sendo bombardeados com novas vagas.

Desesperado, Willie contatou você, um especialista em algoritmos, para desenvolver uma solução que pudesse garantir vagas para o máximo de usuários possível, a fim de evitar os problemas relatados pelos usuários da rede. Felizmente, Willie possui um algoritmo muito eficaz que indica se um usuário é apto ou não para determinada vaga. Esses dados podem ser representados como um grafo bipartido, onde uma aresta candidato-vaga existe se o candidato atende aos requisitos da vaga.

Preocupado com a rápida queda de usuários da LinkOut, você sugeriu utilizar uma espécie de *Proof of Concept* para o problema, propondo uma abordagem gulosa, de baixa complexidade, que encontra uma solução rapidamente. Willie aprovou a ideia, desde que, assim que a solução gulosa estiver pronta, você também implemente um outro algoritmo que sempre encontre a melhor solução possível.

Sua missão é implementar dois algoritmos que encontrem pares únicos de usuários e vagas para o maior número possível de usuários em uma rede. Note que a solução gulosa muitas vezes é subótima, e Willie gostaria que você comparasse ambas soluções, apontando para ele as vantagens e desvantagens de cada algoritmo.

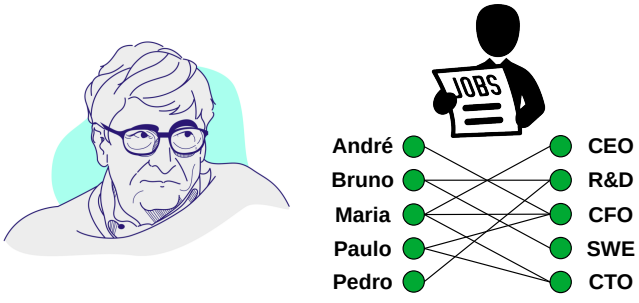

Figura 1: Wille em busca do aumento de engajamento na sua rede social.

3 Exemplo do problema

3.1 Modelagem do problema

Este trabalho prático aborda a parte de grafos e algoritmos gulosos da ementa desta disciplina. Para a resolução do problema a sua modelagem **precisa** usar ambas técnicas e deve ser descrita sucintamente no relatório apresentado.

3.2 Formato da entrada esperada

O seu programa deverá processar um caso de teste em cada execução. A primeira linha de um cenário de teste é composta de três número inteiros U, J e E, representando respectivamente o número de usuários ($2 \leq U \leq 10000$), o número de ofertas de emprego ($1 \leq J \leq 10000$) e o número de qualificações usuário-emprego ($1 \leq E \leq (U * J)$).

Cada uma das próximas E linhas descreve uma qualificação usuário-vaga, representadas pelo nome do usuário seguido pelo nome da vaga, obrigatoriamente nessa ordem. Os dois identificadores são representados por strings, separados por um espaço simples.

3.3 Formato da saída esperada

Para cada caso de teste seu programa deve imprimir duas linhas. A primeira linha deve imprimir o resultado do algoritmo guloso, enquanto a segunda linha deve imprimir o resultado do algoritmo exato. O output deve usar a saída padrão e utilizar o seguinte formato:

Guloso: <RESULT_GREEDY>
Exato: <RESULT_EXACT>

3.4 Casos de teste

Entrada	Saída
5 5 9 Andre CFO Bruno RED Bruno SWE Maria CEO Maria CFO Maria CTO Paulo CFO Paulo CTO Pedro RED	Guloso: 4 Exato: 5

Entrada	Saída
5 4 9 Anna software_engineer Edsel electrical_engineer Edsel senior_php_developer Edsel software_engineer Elisha senior_php_developer Ethelbert c_programmer Ethelbert electrical_engineer Santo c_programmer Santo electrical_engineer	Guloso: 4 Exato: 4

IMPORTANTE: Nos casos de teste, o resultado do algoritmo guloso pode variar bastante de acordo com a estratégia adotada. Por outro lado, a solução ótima (exata) será sempre a mesma. Serão disponibilizados comandos no Makefile (próxima seção) para avaliar tanto o algoritmo exato quanto a

solução gulosa (proximidade do ótimo). Encorajamos o estudante a encontrar a melhor aproximação gulosa possível.

Junto com a descrição do problema, disponibilizaremos um conjunto de casos de teste e as soluções exatas para cada caso de teste.

4 Implementação

O seu programa deverá ser implementado na linguagem C ou C++, e deverá fazer uso apenas de funções da biblioteca padrão da linguagem. Não serão aceitos trabalhos que utilizem qualquer outra linguagem de programação e/ou que façam uso de bibliotecas que não a padrão.

O aluno pode implementar seu programa em qualquer ambiente (Windows, Linux, MacOS, etc...), no entanto, deve garantir que seu código compile e rode nas máquinas do DCC (tigre.dcc.ufmg.br ou jaguar.dcc.ufmg.br ou login.dcc.ufmg.br), pois será neste ambiente que o TP será corrigido. Note que essas máquinas são acessíveis a todos os alunos do DCC com seu login e senha, podendo inclusive ser realizado acesso remoto via SSH. O aluno pode buscar informações no site do CRC (Centro de Recursos Computacionais) do DCC (https://www.crc.dcc.ufmg.br/).

Para facilitar o desenvolvimento vamos fornecer uma estrutura base de arquivos com Makefile já configurado. A pasta TP02-Template-CPP.zip, disponível para download na tarefa do Moodle, contém 2 arquivos: main.cpp e Makefile. O ponto de entrada do seu programa está no arquivo main.cpp. Fique à vontade para criar novos arquivos .hpp e .cpp conforme sua preferência. Para compilar seu programa basta executar o comando "make" no mesmo diretório que o Makefile está. Ao final deste comando, se a compilação for bem sucedida, será criado um arquivo executável chamado "tp02". Esse arquivo pode ser executado pela linha de comando usando "./tp02".

O arquivo da entrada deve ser passado ao seu programa como entrada padrão, através da linha de comando (e.g., $ ./tp02 < casoTeste01.txt) e gerar o resultado também na saída padrão (não gerar saída em arquivo).

Para avaliar automaticamente sua solução em todos os casos de teste disponibilizados, disponiblizamos dois comandos:

- "make eval": avalia apenas os resultados referentes ao algoritmo exato.
- "make eval_greedy": avalia o quanto o resultado guloso é próximo do ótimo.

Nota: Esses comandos foram testados apenas em ambientes Linux.

5 O que deve ser entregue

Deverá ser submetido um arquivo .zip contendo apenas uma pasta chamada tp2, esta pasta deverá conter: (i) Documentação em formato PDF e (ii) Implementação.

5.1 Documentação

A documentação deve ser sucinta e não ultrapassar 5 páginas. Você deve descrever cada solução do problema de maneira clara e precisa, detalhando e justificando os algoritmos e estruturas de dados utilizados. 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 é necessário incluir trechos de código em sua documentação nem mostrar detalhes de sua implementação, exceto quando estes influenciem o seu algoritmo principal, o que se torna interessante. É essencial que a documentação contenha ao menos:

1. **Identificação:** Nome e Matrícula.
2. **Introdução:** Breve resumo do problema com suas palavras.
3. **Modelagem:** Modelos adotados bem como detalhamento e justificativa dos algoritmos e estruturas de dados escolhidos. É necessário também realizar uma análise comparativa dos resultados dos dois algoritmos.

Observações:

- É necessário que você realize a análise de complexidade do seu código no relatório;
- Soluções muito lentas não serão consideradas, uma vez que terão complexidades muito acima da adequada para este problema. Espera-se que solução proposta tenha tempo de execução em torno de 1 segundo por entrada, e soluções que obtiverem tempos muito discrepantes em relação a esse valor serão desconsideradas.

5.2 Implementação

O código fonte submetido deve conter todos os arquivos fonte e o Makefile usado para compilar o projeto. Lembre que seu código deve ser **legível**, então **evite variáveis com nomes não descritivos** (int a, aa, aaa;) e lembre-se de **comentar seu código.** Já estamos fornecendo uma implementação base com os arquivos necessários, então indicamos que você só o altere se for necessário.

5.3 Atrasos

Trabalhos poderão ser entregues após o prazo estabelecido, porém sujeitos a uma penalização regida pela seguinte fórmula:

$$\Delta_p = \frac{2^{(d-1)}}{0.32}\% \tag{1}$$

Nesta fórmula d representa dias de atraso. Por exemplo, se a nota dada pelo corretor for 70 e você entregou o TP com 4 dias corridos de atraso, sua penalização será de $\Delta_p = 25\%$ e, portanto, a sua nota final será: $N_f = 70(1 - \Delta_p) = 52.2$. Note que a penalização é exponencial e 6 dias de atraso resultam em uma penalização de 100%.

6 Considerações finais

Assim como em todos os trabalhos desta disciplina, é estritamente proibida a cópia parcial ou integral de códigos, seja da internet ou de colegas. Utilizaremos o algoritmo MOSS para detecção de plágio em trabalhos, seja honesto. Você não aprende nada copiando código de terceiros nem pedindo a outra pessoa que faça o trabalho por você. Se a cópia for detectada, sua nota será zerada e os professores serão informados para que as devidas providências sejam tomadas.