Trabalho Prático 2: Uaibucks

Algoritmos e Estruturas de Dados III – 2017/2
Entrega: 20/11/2017

1 Introdução

Nubby e seus amigos decidiram abrir uma startup de consultoria. O primeiro cliente da startup é uma famosa cafeteria chamada Uaibucks, a qual deseja abrir coffee shops em determinadas esquinas de uma nova cidade. Entretanto, Uaibucks já recebeu muitas reclamações de suas filiais e críticas de seus clientes de outras cidades de que o número de cafeterias é muito excessivo, muitas vezes com cafeterias em esquinas próximas. Assim, a Uaibucks decidiu impor algumas restrições para a implementação de novas filiais, conforme descrito a seguir.

Suponha que a cidade alvo possua N esquinas, e duas esquinas são ditas vizinhas se existe uma rua interligando-as. Cada uma das esquinas possui uma demanda de clientes esperada $d_i \in \mathbb{N}$. Dentre todas as esquinas da cidade a serem consideradas, a empresa deseja escolher um subconjunto de esquinas para abrir novas filiais de modo que duas filiais não estejam em esquinas vizinhas para evitar reclamações de suas filiais e clientes e a escolha das esquinas deve maximizar a demanda esperada.

Após Nubby pensar muito sobre o problema ele acabou desistindo, pois ele não conseguiu encontrar um algoritmo polinomial (determinístico) para resolver o problema. Bryan, após observar o esforço de seu amigo, indicou a Nubby que o problema é NP-Difícil, conteúdo que Nubby ainda não aprendeu em seu curso online de algoritmos. Como Nubby está ocupado com a gerência da startup, ele atribuiu a você um dos membros da startup a tarefa de modelar o problema utilizando teoria dos grafos, provar que o problema a ser resolvido pode ser modelado como um problema de decisão e sua versão de decisão é um problema NP-Completo e implementar um algoritmo exato e uma heurística para o problema.

2 Entrada e saída

Entrada A primeira linha contém dois inteiros positivos: N indicando o número de esquinas na determinada cidade ($1 \leq N \leq 30$) para o algoritmo exato e $1 \leq N \leq 10^5$ para a heurística) e M ($1 \leq M \leq N(N-1)/2$) indicando o número de pares de esquinas vizinhas. As esquinas são identificadas por ids de 1 a N, assim, cada uma das N linhas seguintes possui um único número natural natural d_i ($d_i \geq 0$) indicando a demanda de clientes esperada na i-ésima esquina, ou seja, esquina com id i. Depois disso, M linhas, cada uma contendo dois inteiros x e y ($1 \leq x \leq N$ e $1 \leq y \leq N$ e $x \neq y$) indicando que as esquinas x e y são vizinhas.

Exemplo de entrada

```
7 8
6
2
1
2
2
1
4
1 2
6 3
6 7
7 5
4 1
1 3
3 7
5 6
```

Saída A primeira linha deve conter um inteiro k indicando quantas esquinas foram selecionadas e um número natural r, o qual representa o soma das demandas esperadas das esquinas selecionadas. Em seguida, na segunda linha deve-se imprimir k inteiros separados por espaço onde cada inteiro é o id de cada esquina selecionada. Em caso de múltiplas soluções ótimas imprima a que possuir o maior número de esquinas selecionadas. No exemplo abaixo foram selecionadas as esquinas com ids 1 e 7, com demanda total 10.

Exemplo de saída

```
2 10
1 7
```

3 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.

Documentação Deverá ter no máximo 10 páginas e seguir tanto os critérios de avaliação discutidos na Seção 4.1, bem como as diretrizes sobre a elaboração de documentações disponibilizadas no *moodle*. Além disso, a documentação deverá conter modelagem do problema utilizando teoria de grafos, formulação do problema como um problema de decisão, demonstração de que o problema é NP-Completo, análise teórica das complexidades de tempo e espaço dos algoritmos, análise experimental validando a análise teórica e comparação entre os resultados obtidos utilizando o algoritmo exato e a heurística proposta.

Implementação O código fonte do seu TP deve estar separado em arquivos ċ e ḣ, **por exemplo** crie um arquivo *exato.*{*c/h*} e *heuristica.*{*c/h*}.

Makefile Inclua um *makefile* na submissão que permita compilar o trabalho. **Deverão ser produzidos dois executáveis** chamados *exato* e *heuristica*.

É obrigatório o uso das *flags*: **-Wall -Wextra -Werror -std=c99 -pedantic -O2**. **Pode-se assumir que bibliotecas padrões da linguagem serão automaticamente linkadas pelo compilador**.

Execução A entrada e a saída devem ser **lidas e impressas utilizando a entrada/saída padrão (stdin/stdout)**. Um exemplo de execução utilizando o executável *heuristica* para um arquivo de entrada *in_1.in* e imprimindo o resultado em *out_1.out* será da seguinte forma:

$$./heuristica < in\_1.in > out\_1.out$$

4 Avaliação

Uma lista **não exaustiva** dos critérios de avaliação que serão utilizados é apresentada a seguir.

4.1 Documentação

Introdução Inclua uma breve explicação do problema que está sendo resolvido no seu trabalho.

Solução do Problema 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 torna o detalhamento interessante.

Análise de Complexidade Inclua uma análise de complexidade de tempo e espaço dos principais algoritmos e estrutura de dados utilizados. Cada complexidade apresentada deverá ser devidamente **justificada** para que seja aceita. Faça uma análise para cada tipo de algoritmo implementado. Mostre vantagens e desvantagens de cada um deles.

Avaliação Experimental Sua documentação deve incluir os resultados de experimentos que avaliem o tempo de execução de seu código em função de características da entrada. Cabe a você gerar entradas para esses experimentos. Para tal, um gráfico mostrando o tempo de execução em função do tamanho da entrada pode ser interessante. Você também deve interpretar os resultados obtidos. Comente sobre cada gráfico ou tabela que você apresentar mostrando o que é possível concluir a partir dele.

4.2 Implementação

Linguagem & Ambiente 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 ou utilizem outras bibliotecas que não seja padrão ou variáveis globais serão zerados. Além disso, é recomendado que se certifique que seu código compile e funcione corretamente nas máquinas **Linux** dos laboratórios do DCC.

Casos de teste A sua implementação passará por um processo de correção automatizado, portanto, o formato da saída do seu programa deve ser idêntico aquele descrito nessa especificação. Saídas com qualquer divergência serão consideradas erradas, mesmo que as divergências sejam *whitespaces*. e.g.

espaços, *tabs*, quebras de linha, etc. Para auxilia-lo na depuração do seu código, será fornecido um pequeno, **não-exaustivo**, conjunto de entradas e suas respectivas saídas. É seu dever certificar-se que seu código funciona corretamente para qualquer entrada válida.

Alocação Dinâmica Algoritmos e estruturas de dados deverão fazer uso de memória alocada dinamicamente (`malloc()` ou `calloc()`). Certifique-se que seu programa utiliza essas regiões de memória corretamente, pois os monitores penalizarão implementações que realizam *out-of-bounds access* e que tenham vazamento de memória (não desalocar memória dinâmica).
A alocação dinâmica deverá fazer uso das funções malloc() ou calloc() da biblioteca padrão C, bem como liberar tudo o que for alocado utilizando free(), para gerenciar o uso da memória. **DICA:** Utilize `valgrind` antes de submeter o seu TP.

Qualidade do código Seu código também será avaliado no quesito de legibilidade, dando atenção, porém não limitando-se, aos seguintes items: (i) **INDENTAÇÃO**; (ii) nomes de variável e função descritivos e claros; (iii) Modularização adequada; (iv) Comentários dentro de funções, explicando o que certos trechos mais complicados fazem; (v) Comentários fora de funções, explicando, em alto-nível, o que as funções mais importantes fazem; (vi) funções concisas que desempenham somente uma tarefa; (vii) **Proibido uso de variáveis globais**.

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

$$\Delta_p = \frac{2^{d-1}}{0.32}\,\%$$

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: $N_f = 70 \cdot (1 - \Delta_p) = 52.2$. Note que a penalização é exponencial e 6 dias de atraso resultam em uma penalização de 100%.

5 Considerações Finais

Assim como em todos os trabalhos dessa disciplina **é estritamente proibida a copia parcial ou integra 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.