CI1238 - Otimização

Segundo Trabalho

3 de junho de 2023

1 Introdução

O trabalho consiste em modelar e implementar, por *Branch & Bound*, o problema *Separação de grupo minimizando conflitos*, descrito na Seção 2.

A resolução do problema, ou seja, a descrição do problema, da modelagem e da implementação, deve estar em um texto claro em formato de um artigo em pdf. Além disso, deve ser feita uma análise com duas funções limitantes (*bounds*) diferentes, sendo uma dada pelo professor (Seção 3) e outra escolhida pelos alunos (que deve ser melhor que a dada). A função limitante dos alunos deve ser a default na implementação e a dada pelos professores é escolhida usando uma opção da linha de comando (`-a`).

O texto deve conter o nome dos autores (alunos), uma introdução com o problema, a modelagem e sua explicação (de por que essa modelagem resolve o problema), detalhes da implementação (com exemplos de uso), e uma análise do uso das funções limitantes. Nesta análise devem ser feitas e comparadas contagens de número de nós da árvore e tempo de execução (com relatório gerado pelo programa). Outras métricas também podem ser usadas.

Todas as referências que forem usadas devem estar citadas corretamente no texto.

Para facilitar a análise, o seu programa deve ter a opção de fazer ou não os cortes por viabilidade e otimalidade. Ou seja, com a opção de linha do comando `-f` os cortes por viabilidade não devem estar ativos; e com a opção de linha do comando `-o` os cortes por otimalidade são desligados.

Você pode usar bibliotecas para estruturas de dados (como listas, conjuntos etc), mas não para o algoritmo de resolução principal do problema. O seu programa deve compilar e executar nas servidoras do DINF.

O trabalho deve ser entregue com um `makefile` de forma que ao digitar o comando `make` o executável `separa` seja construído.

Resumindo, o texto deve ter:

- identificação;
- explicação do problema;
- modelagem;
- análise das funções limitantes;
- detalhes da implementação.

A implementação:

- deve ter executável de nome `separa`;
- na opção default todos os cortes (viabilidade e otimalidade) estão ativos e a função limitante é a criada pelos autores;
- com a opção `-f` na linha de comando deve desligar os cortes de viabilidade;
- com a opção `-o` na linha de comando deve desligar os cortes de otimalidade;
- com a opção `-a` na linha de comando deve usar a função limitante dada pelos professores;
- deve gerar relatório (na saída de erro padrão, `stderr`) com número de nós da árvore e tempo gasto (sem contar o tempo de entrada e saída).

Você deve entregar um arquivo compactado (`tar.gz`) com os seguintes arquivos no diretório corrente:

- texto (em pdf);
- os fontes (podem estar em subdiretórios);
- makefile;
- exemplos usados na análise (podem estar em subdiretórios).

A entrega deve ser feita por e-mail para `andre@inf.ufpr.br` em um arquivo compactado com todos os arquivos do trabalho, com assunto "Otimização-trabalho 2" (exatamente).

2 O problema

Separação de grupo minimizando conflitos

Em um certo mundo existe um grupo de n super-heróis. Alguns dos super-heróis tem conflitos entre si, enquanto outros tem grande afinidade, a ponto de só funcionarem juntos. Como neste mundo apareceram dois vilões, os n super-heróis resolveram se dividir em dois grupos. Esta separação tem que ser tal que todos os pares de super-heróis com grande afinidade DEVEM ficar no mesmo grupo, e deve também minimizar os pares com conflito dentro de um mesmo grupo.

Considere que cada super-herói é indexado pelos números de 1 a n. Seja C o conjunto de pares (a_i, b_i), com $1 \leq i \leq k = |C|$, tais que a_i e b_i tem conflito, e seja A o conjunto de pares (a_i, b_i), com $1 \leq i \leq m = |A|$, tais que a_i e b_i tem grande afinidade.

Dados n, C e A, queremos encontrar uma separação em dois grupos não vazios de tal forma que para todo par $(x, y) \in A$, x e y estão no mesmo grupo, e que minimiza o número de pares $(u, v) \in C$ tais que u e v estão no mesmo grupo[1].

2.1 Formato de entrada e saída

Os formatos de entrada e saída, são descritos a seguir e devem ser usados a entrada e a saída padrão (`stdin` e `stdout`).

A entrada é formada de um conjunto de números inteiros. Os números podem estar separados por 1 ou mais espaços, tabs ou fim de linha.

Entrada: Inicia com os valores de n (número de itens), k (número de conflitos) e m (número de afinidades) na primeira linha (separados por espaço). Em seguida temos $k + m$ linhas com dois números (separados por espaço) representando os k conflitos e as m afinidades.

Saída: O número de conflitos que não foram evitados devem aparecer na primeira linha. Na segunda linha devem estar os elementos do grupo que contém o primeiro super-herói, em uma mesma linha, em ordem crescente, separados por espaço (simples) e sem espaço no começo nem no fim da linha.

2.2 Exemplos

Exemplos de entrada e saída.

[1] Dica: enumere os subconjuntos que representam um dos lados.

2.2.1 Exemplo simples com $n = 4$, $k = 2$ e $m = 2$

Entrada:

```
4 2 2
1 3
2 4
1 2
3 4
```

Saída:

```
0
1 2
```

2.2.2 Exemplo simples com $n = 3$, $k = 3$ e $m = 0$

Entrada:

```
3 3 0
1 2
1 3
2 3
```

Saída:

```
1
1 2
```

3 Função limitante dada

Dados o conjunto de super-heróis com grupos já escolhidos (E), sabendo que $g(i)$ é o grupo do super-herói i (já escolhido), definimos o conjunto C_E como sendo o conjunto dos conflitos que envolvem apenas super-heróis com grupos escolhidos.

Um triângulo em um conjunto $C' \subseteq C$ é uma tripla (x, y, z) tal que $(x, y), (x, z), (y, z) \in C'$.

Seja t_E o número de triângulos em $C \setminus C_E$ que não compartilham nenhum par de super-heróis.

Podemos então definir a função $B_{dada}(E)$ por:

$$B_{dada}(E) = |(x, y) \in C_E \mid g(x) = g(y)| + t_E.$$

Ou seja, $B_{dada}(E)$ é o número de conflitos onde os dois super-heróis envolvidos já foram colocados em um mesmo grupo mais o número de vezes que um triângulo de conflitos aparece no conjunto de conflitos ainda não decididos.

4 Dicas

Lembrem que uma função limitante em um problema de minimização deve ser sempre menor ou igual ao valor ótimo do subproblema, para garantir que nenhuma solução ótima seja cortada.