Universidade Federal de Minas Gerais
Departamento de Ciência da Computação
Algoritmos e Estruturas de Dados III
2º Semestre de 2013

Trabalho Prático 2 - Festa na Piscina

1 Introdução

*Z*ambis quer dar uma festa para inaugurar sua nova piscina e quer chamar todos os seus amigos. Porém ele não quer perder tempo chamando cada um deles. Para resolver isso, ele sabe que existem pessoas de sua rede de amizade que se conhecem. Então ele quer pedir para o menor número de amigos chamarem quem eles conhecem até que todos os amigos do Zambis sejam convidados. Vale lembrar que os amigos do Zambis só irão chamar quem eles conhecem. Quantas pessoas o Zambis vai precisar chamar para que todos seus amigos sejam convidados?

2 Soluções

Para este trabalho, você deverá apresentar duas soluções: uma que apresente a solução ótima (solução exata) e outra solução - não necessariamente ótima - obtida através de uma heurística. Caso você consiga desenvolver - e provar - que sua heurística é um algoritmo aproximativo, você ganhará um ponto extra!

3 Entrada

O arquivo de testes conterá diversas instâncias de teste - a primeira linha do arquivo contém um inteiro com o número de instâncias presentes no teste. Na primeira linha de cada instância terá um inteiro A que mostra quantos amigos o Zambis tem. A seguir a explicação das próximas A linhas:

- Os amigos são identificados por um número inteiro único C começando pelo 1;
- A linha A_c indica os amigos da pessoa com identificador c;
- Vale lembrar que se o amigo identificado como 1 ($c = 1$) for amigo de 2, então 2 também é amigo de 1. Sendo assim, a amizade entre 1 e 2 só será mostrada no primeiro amigo (neste caso em 1).

Após isso, começará uma nova instância. Veja o exemplo abaixo:

```
2
10
1 2 3 4 5
2 3 4
3 4
4 5 6
5
6 7 10
```

7 8 9
8 9
9
10
7
1 2 5
2 3 5
3 4 5
4 5 6 7
5
6 7
7

4 Saída

Para cada instância de teste, a saída deverá mostrar o número de amigos que o Zambis precisou chamar para que todos os seus amigos fossem convidados. A saída de cada instância deve estar em uma linha. Entre duas instâncias distintas NÃO se deve possuir espaço em branco.

Veja a saída do exemplo anterior:

3
2

5 Entrega

- A data de entrega desse trabalho é **11/10/2013**.
- A penalização por atraso obedece à seguinte fórmula $2^{d-1}/0.32\%$, onde d são os dias úteis de atraso.
- Submeta apenas um arquivo chamado <número matricula>_<nome>.zip. Não utilize espaços no nome do arquivo. Ao invés disso utilize o caractere '_'.
- Não inclua arquivos compilados ou gerados por IDEs. **Apenas** os arquivos abaixo devem estar presentes no arquivo zip.
  - Makefile
  - Arquivos fonte (*.c e *.h)
  - Documentacao.pdf
- Os executáveis gerados devem ser nomeados como *tp2e* para o algorítmo exato e *tp2h* para a heurística.
- Não inclua **nenhuma** pasta. Coloque todos os arquivos na raiz do zip.
- Siga rigorosamente o formato do arquivo de saída descrito na especificação. Tome cuidado com whitespaces e formatação dos dados de saída

- NÃO SERÁ NECESSÁRIO ENTREGAR DOCUMENTAÇÃO IMPRESSA!
- Será adotada **média harmônica** entre as notas da **documentação e da execução**, o que implica que a nota final será 0 se uma das partes não for apresentada.

6 Documentação

A documentação não deve exceder 10 páginas e deve conter pelo menos os seguintes itens:

- Uma **introdução** do problema em questão.
- **Modelagem e soluções** propostas para o problema. Os algoritmos devem ser explicado de forma clara, possivelmente através de pseudo-código e esquemas ilustrativos. Nesse tópico, caso exista, deve possuir a explicação do porque sua heurística é um algoritmo aproximado.
- **Análise de complexidade** de tempo e espaço das soluções implementadas. É importante que você compare as duas soluções propostas.
- **Experimentos** variando-se o tamanho da entrada e quaisquer outros parâmetros que afetem de forma significante a execução das soluções. Aqui, novamente, é importante que você compare as duas soluções propostas.
- Especificação da(s) **máquina(s) utilizada(s)** nos experimentos realizados.
- Uma breve **conclusão** do trabalho implementado.

7 Código

O código deve ser obrigatoriamente escrito na **linguagem C**. Ele deve compilar e executar corretamente nas máquinas Linux dos laboratórios de graduação.

- O utilitário **make** deve ser utilizado para auxiliar a compilação, um arquivo *Makefile* deve portanto ser incluído no código submetido. O utilitário deverá gerar dois executáveis com o nome **tp2e** e **tp2h** que deverão **obrigatoriamente** ser capazes de receber o nome de um arquivo de entrada - de onde serão lidas as instâncias do problema - e o nome de um arquivo de saída - onde serão gravadas as soluções. O comando para executá-lo deverá seguir o exemplo abaixo:
  /.tp2e input.txt output.txt
  /.tp2h input.txt output.txt
- As estruturas de dados devem ser **alocadas dinamicamente** e o código deve ser **modularizado** (divisão em múltiplos arquivos fonte e uso de arquivos cabeçalho .h)
- **Variáveis globais** devem ser evitadas.

- Parte da correção poderá ser feita de forma automatizada, portanto siga rigorosamente **os padrões de saída especificados**, caso contrário sua nota pode ser prejudicada.
- **Legibilidade e boas práticas** de programação serão avaliadas