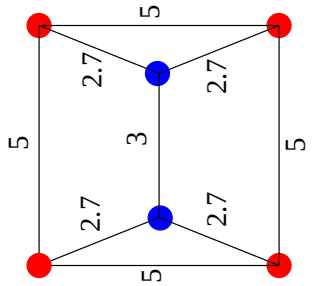
Projeto e Análise de Algoritmos II
Trabalho Prático
Prof. Eduardo C. Xavier

Você trabalha como consultor na área de otimização e algoritmos. Um dia é chamado por uma empresa local que faz projetos para a instalação de cabeamento (de telefones, TV paga etc). O gerente de produção começa lhe explicando o problema que desejam otimizar: Dado um conjunto de caixas de ligação distribuídas na cidade, deseja-se em um determinado momento fazer a ligação entre algumas destas caixas de tal modo que todas estas caixas, chamadas de terminais, estão ligadas entre si por cabeamentos. Para fazer a conexão entre todos os terminais você pode, mas não é necessário, usar outras caixas de ligação existentes como pontos intermediários por onde passará um cabo. O gerente de produção lhe diz então que deseja construir uma rede que conecte todos os terminais, de tal forma que o total gasto com o cabeamento seja mínimo.

Você começa a pensar sobre o problema e a primeira idéia que vem a sua cabeça é construir uma árvore geradora mínima considerando apenas os terminais. Algum tempo depois você percebe que isto não funciona. Você constrói um exemplo (abaixo) onde os nós vermelhos são terminais. Uma árvore mínima considerando apenas as ligações com terminais possui custo 15. Se pudermos usar os nós intermediários (azuis) teremos uma solução com custo $4 \cdot 2.7 + 3 = 13.8$.



Você diz para o gerente que vai pensar sobre o problema. Mais tarde no seu escritório, após algumas tentativas de projeto de algoritmos polinomiais você começa a desconfiar que o problema é NP-Difícil. Após algumas cabeçadas na mesa você se lembra que este problema nada mais é do que uma versão do problema *Steiner Tree*: Dado um grafo $G = (V, E)$ com custos positivos nas arestas, e um conjunto de terminais $T \subset V$, deseja-se achar um subgrafo conexo de custo mínimo que conecte todos os vértices de T. Para tanto você pode usar vértices (conhecidos como *steiner*) que não pertencem a T para tentar reduzir o custo do subgrafo. É claro que o subgrafo de custo mínimo será uma árvore (não é necessariamente uma árvore geradora de todos os vértices!). Este problema é NP-Difícil!

Neste trabalho vocês devem implementar uma heurística para o problema descrito anteriormente. Alguns detalhes sobre o trabalho:

- O trabalho deve ser feito em grupos de 3 ou 4 pessoas.
- A sua heurística deve ser implementada em C, Java ou Python padrões (C compilável gcc, Java com java da SUN).

- Vocês devem **entregar via email até o dia 10 de junho** um único arquivo com nome RAxxxxx-RAxxxxxRAxxxxx.zip onde são especificados os RAs dos alunos do grupo. Dentro deste arquivo compactado deve constar:

  1. Diretório chamado **codigo**: contém todos os arquivos da sua implementação bem como um arquivo makefile para compilar o código. O arquivo executável gerado deve ter o nome stein (stein.py no caso de Python).
  2. Arquivo pdf que é um relatório do trabalho: Vocês devem escrever um relatório de no máximo 10 páginas, contendo pelo menos: 1) Os nomes e RAs dos integrantes do grupo 2) Organização e breve descrição dos arquivos código fonte 3) Heurística utilizada 4) Descrição em alto nível da sua implementação mostrando o funcionamento geral da heurística; quais estruturas de dados foram usadas (para representar soluções para acelerar geração de vizinhança etc); quais operadores foram usados; qual a vizinhança utilizada; descrição de setup de parâmetros da heurística; etc 5) Uma seção com resultados computacionais descrevendo testes realizados, soluções encontradas e tempo de execução (use tabelas e gráficos para mostrar os resultados).
  3. No email enviado para o professor deve constar no *subject* apenas o texto: **trabalho mc548**.
  4. A heurística a ser implementada por cada grupo será definida pelo professor e estará disponível na página da disciplina.

  No endereço `http://www.ic.unicamp.br/~ecx/mc548/inst.zip` vocês encontram algumas instâncias para o problema de *Steiner*, obtidas da SteinLib (`http://steinlib.zib.de/steinlib.php`). Nestas instâncias os grafos são completos e não direcionados. Em todas as instâncias vocês podem assumir que o grafo é completo e não direcionado. Os alunos são encorajados a testar suas heurísticas com outras instâncias que podem ser geradas de forma aleatória (crie um grafo completo com pesos aleatórios nas arestas e escolha alguns terminais de forma aleatória). No relatório vocês devem reportar testes com estas instâncias bem como dados sobre a geração de tais instâncias (número de vértices, número de terminais, como foram atribuídos pesos nas arestas, e como foram escolhidos os terminais). Um exemplo de *script* em Python para gerar grafos foi disponibilizado pelo professor dentro da pasta *inst*.

  Cada arquivo de entrada tem o seguinte formato:

```
Nodes  número de vértices
Edges número de arestas
E i j valor
...
E i j valor

Terminals número de terminais
T número do nó
...
T número do nó
```

  As primeiras duas linhas indicam o tamanho do grafo. Em seguida são descritas para cada par de vértices o peso da aresta correspondente. Após descrever os pesos de todas as arestas há uma linha em branco seguido de *Terminals* e o número de terminais da instância. Em seguida em cada linha é especificado o número de um vértice terminal. Exemplo para um grafo com 4 vértices e terminais 1 e 3:

```
Nodes 4
Edges 6
E 1 2 298
E 1 3 285
E 1 4 378
E 2 3 1001
E 2 4 298
E 3 4 567

Terminals 2
T 1
T 3
```

O seu programa deve aceitar um arquivo de entrada pela linha de comando (stein *inst*) e deve gerar uma saída da seguinte forma:

```
Solucao: inst
Valor: valor
Arestas usadas: total
i j
i j
...
```

Na primeira linha é impresso o nome da instância de entrada, em seguida é impresso o valor da solução. Na linha seguinte deve ser impresso em *total* o número de arestas usadas, e em seguida as arestas usadas, uma por linha.

Importante: O seu programa deve executar no máximo 2 minutos para cada instância, e deve então imprimir a melhor solução encontrada. Plágio ou cópias de trabalho não serão aceitas! Vocês podem porém usar idéias descritas em artigos que apresentam heurísticas para este problema (Steiner Tree). Neste caso vocês devem fazer a própria implementação da heurística, e devem citar os artigos de onde tiraram idéias de implementação. Os trabalhos que não satisfaçam as restrições descritas neste texto terão nota 0.

Sobre a nota: A nota de um trabalho entregue varia de 0 a 10, mas 3 pontos serão atribuídos comparando-se os resultados obtidos entre os trabalhos entregues pelos alunos. O professor realizará testes de comparação entre as heurísticas e as melhores receberão as melhores notas. Note que como vocês tem no máximo 2 minutos de tempo de execução, a escolha da linguagem a ser usada na implementação pode fazer diferença.