ORGANIZAÇÃO DOS PROJETOS

Serão desenvolvidos alguns projetos durante o semestre. Todos serão organizados como descrito a seguir.

Basicamente, os programas lerão 2 arquivos: um arquivo que descreve um conjunto de dados a ser armazenados em alguma estrutura de dados e um arquivo descrevendo operações sobre o primeiro conjunto. As operações podem causar a delição, modificação, inserção de dados ao primeiro conjunto.
O resultado final do processamento de cada conjunto é gravado em um ou mais arquivos de saída.

ENTRADA DE DADOS

A entrada de dados, via de regra, ocorrerá por meio de um ou mais arquivos. Estes arquivos estarão sob um diretório, referenciado por **BED** neste texto.[1]

SAIDA DE DADOS

Os dados produzidos serão mostrados na saída padrão e/ou em diversos arquivos-texto. Alguns resultados serão gráficos no formato SVG. Os arquivos de saída serão colocados sob um diretório, referenciado por **BSD** neste texto.[2]

ORGANIZAÇÃO DA ENTREGA

O trabalho deve ser submetido no formato **ZIP**, cujo nome deve ser curto, mas suficiente para identificar o aluno ou a equipe.[3] Este arquivo deve estar organizado como descrito à frente.

PROCESSO DE COMPILAÇÃO E TESTES DO TRABALHO

Organização do ZIP a ser entregue

A organização do zip a ser entregue pelo aluno deve ser a seguinte:

[abreviatura-nome]	*Por exemplo, josers.*
`LEIA-ME.txt`	*colocar a matrícula e o nome do aluno. Atenção: O número da matrícula de estar no início da primeira linha do arquivo. Só colocar os números; não colocar qualquer pontuação.*
`*`	*Outros arquivos podem ser solicitados a cada fase.*
`/src`	*(arquivos-fonte)*
`makefile`	*deve ter target para a geração do arquivo objeto de cada módulo e o target solicitado em cada trabalho (p.e.,* **`t1)`** *que produzirá o executável de mesmo nome dentro do mesmo diretório* **`src`***. Os fontes devem ser compilados com as opções -fstack-protector-all. -fstack-check*

1 Indicado pela opção -e.
2 Indicado pela opção -o.
3 Por exemplo, josers.zip (se aluno se chamar José Roberto da Silva), josers-mariabc.zip (para uma equipe com dois alunos. Evite usar maiúsculas, caracteres acentuados ou especiais.

	** adotamos o padrão C99. Usar a opção -std=c99.*
`*.h e *.c`	***Atenção:** não devem existir outros arquivos além dos arquivos fontes e do makefile*

Organização do diretório para a compilação e correção dos trabalhos (no computador do professor):

[HOME_DIR]	
`*.py`	*scripts para compilar e executar*
`\t`	*diretório contendo os arquivos de testes*
`*.geo *.qry`	*arquivos de consultas, talvez, distribuídos em alguns outros sub-diretórios*
`\alunos`	*(contém um diretório para cada aluno)*
`\abrnome`	*diretório pela expansão do arquivo submetido (p.e., josers)*
	outros subdiretórios para os arquivos de saída informados na opção `-o`

Os passos para correção serão os seguintes:

1. O arquivo `.zip` será descomprimido dentro do diretório alunos, conforme mostrado acima
2. O makefile provido pelo aluno será usado para compilar os módulos e produzir o executável. Os fontes serão compilados com o compilador gcc em um máquina virtual Linux. Os executáveis devem ser produzidos no mesmo diretório dos arquivos fontes O professor usará o GNU Make. Serão executadas (a partir dos scripts) o seguinte comando:
   - **`make t2`**
3. O programa será executado automaticamente várias vezes: uma vez para cada arquivo de testes e o resultado produzido será inspecionado visualmente pelo professor. Cada execução produzirá (pelo menos) um arquivo `.svg` diferente dentro do diretório informado na opção **-o**. Possivelmente serão produzidos outros arquivos `.svg` e `.txt`.

APENDICE

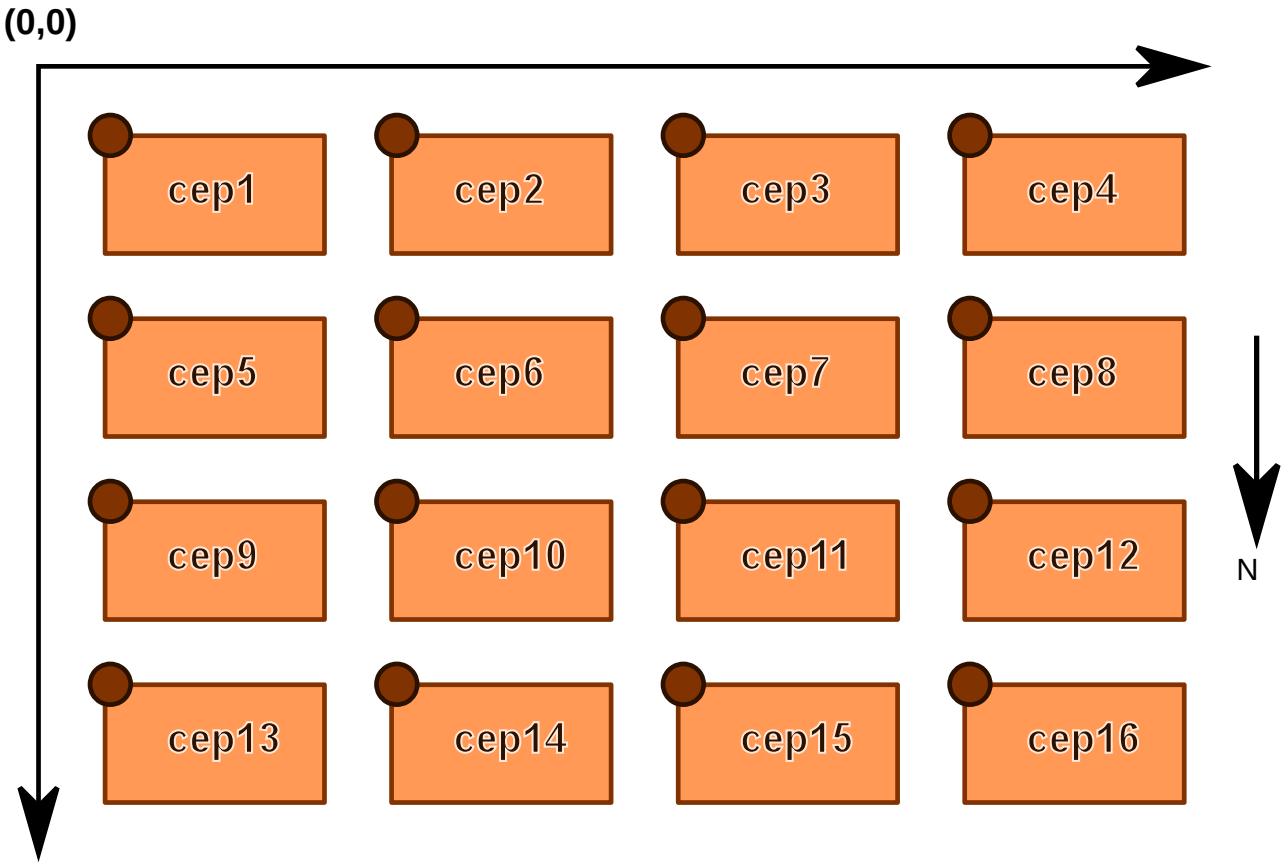
https://www.gnu.org/software/make/manual/make.html
http://opensourceforu.com/2012/06/gnu-make-in-detail-for-beginners/

TRABALHO 2

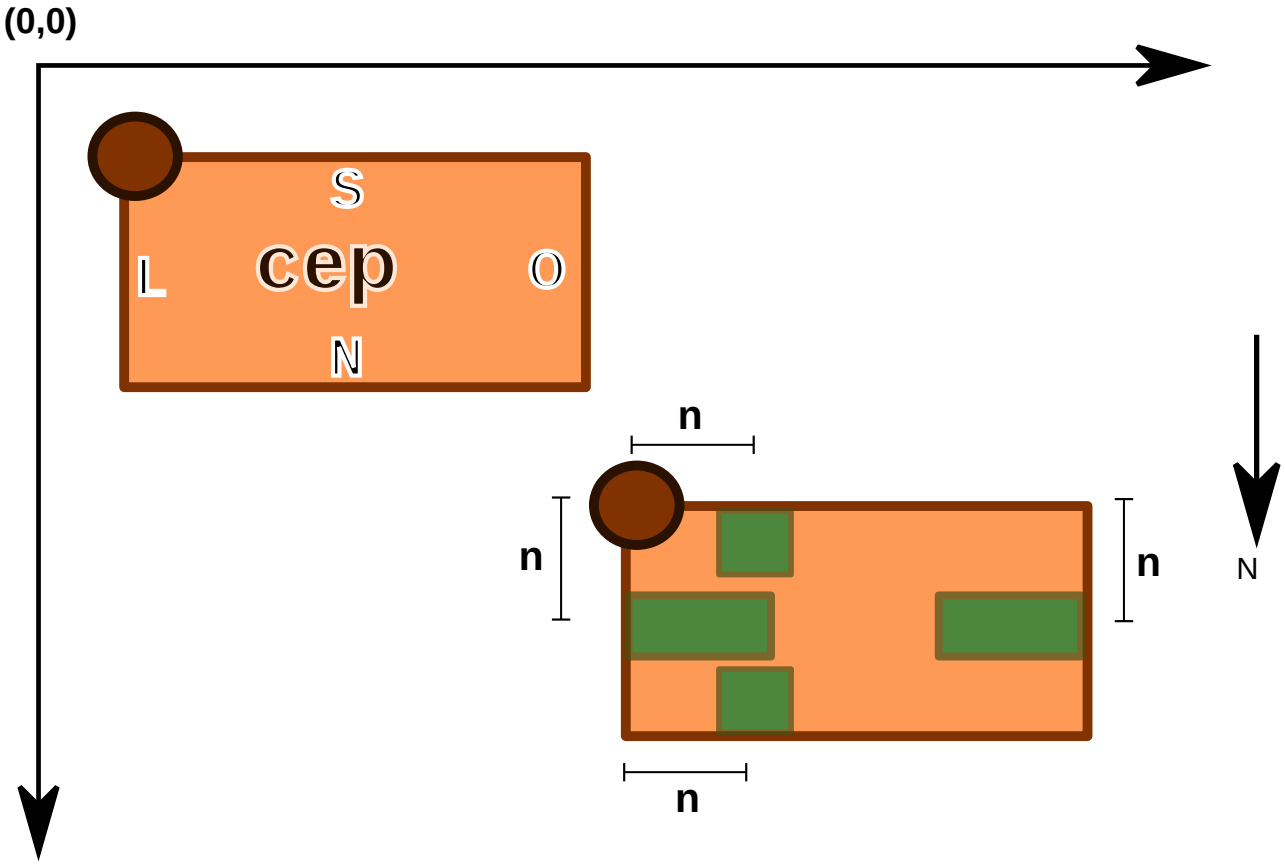
Um Sistema de Informações Geográficas (SIG), para a nossa finalidade, é um sistema que contém (não exclusivamente) dados geo-referenciados, isto é, dados com algum atributo de localização espacial (uma coordenada).

O sistema manipulará o mapa de uma cidade e algumas informações relacionadas.

O mapa de uma cidade é, basicamente, composto por um conjunto de retângulos que representam as quadras.



A cidade exemplificada acima chama-se **Bitnópolis** e possui 16 quadras. O sistema de endereçamento de Bitnópolis é inspirado no de nossa capital federal. Cada **quadra** possui 4 **faces** (N,S,L,O) e é identificada por um **CEP** alfanumérico. O número de uma casa ou estabelecimento comercial é a **distância** da frente da casa até uma projeção do ponto de ancoragem do retângulo que representa a quadra (veja figura abaixo). Assim, um endereço é da forma CEP/Face/número, por exemplo, cep15/S/45. O ponto de ancoragem do retângulo é o canto sudeste da quadra.



Queremos determinar trajetos na cidade. Uma possível consulta seria:

"*Qual é o melhor (mais curto, mais rápido) entre os endereço X e Y*?".

O resultado das consultas são textuais e pictóricas. Um exemplo de uma resposta textual poderia ser:

> *Siga na direção norte na Rua Xxxx até o cruzamento com a Rua Yyyy. Siga na Rua Yyyy na direção sul.....*

O resultado pictórico é mostrado no arquivo `.svg` produzido, evidenciando o caminho a ser percorrido. Algumas consultas podem interditar trechos de ruas ou regiões da cidade.

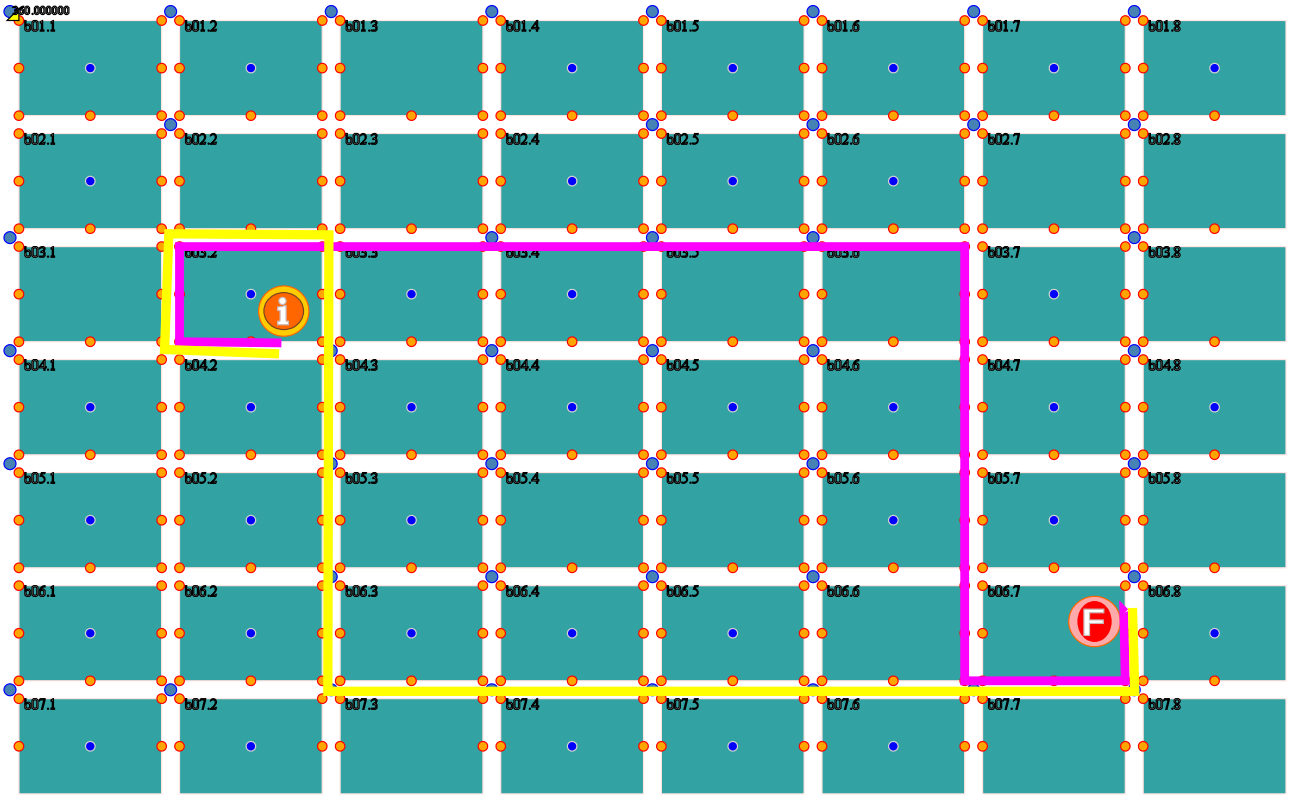


Figura 1: Exemplo de saída da consulta p?. Note o percurso mais rápido e o mais curto.

Algumas consultas podem resultar nas condição do tráfego. Estas alterações impactam o cálculo do percurso, a saber: a operação catac interdita completamente trechos de ruas e a operação rv reduz a velocidade média de trechos de ruas.

Note que estas alterações podem fazer com que certas regiões da cidade tornem-se inacessíveis. Neste projeto, vamos considerar que uma região é inacessível se: efetivamente não existem caminhos entre elas (tipicamente, trechos de ruas interditadas por operações **`catac`**) ou, se o acesso a elas só seja possível por trechos de ruas com uma velocidade média muito baixa (por exemplo, imagine congestionamentos).

A Entrada

Bitnópolis tem quadras e um sistema viário.

Cidade

A cidade é descrita pelo arquivo **`.geo`**:

comando	parâmetros	
q	`cep x y w h`	*Insere uma quadra (retângulo e cep)*
cq	`sw cfill cstrk`	*Cores do preenchimento (`cfill`) e da borda (`cstrk`) das quadras, espessura da borda (`sw`) (a partir deste comando)*
Novos comandos do arquivo .geo		

O Sistema Viário

A entrada de dados será feita arquivo-texto (**arquivo**.**via**) com a descrição do sistema viário da cidade. A maior parte das novas consultas indagam sobre o melhor percurso (menor distância, menor tempo estimado) entre uma origem e um destino.

A origem e o destino são referências geográficas: endereços de origem (**`@o?`**) e o endereço de destino (**`p?`**)

A resposta esperada é a descrição textual do percurso e a representação pictórica do percurso do caminho mais curto e do caminho mais rápido.

Atenção: algumas consultas fazem com que alguns segmentos de rua fiquem intransitáveis.

O sistema viário da cidade é representado por um grafo direcionado. Algumas consultas se referem a subgrafos particulares: a árvore geradora de custo mínimo e componentes conexos.

O Mapa Viário

O mapa viário é um grafo direcionado: os vértices representam os extremos de um segmento de rua e os arcos representam um segmento de rua e indicam o sentido do tráfego.

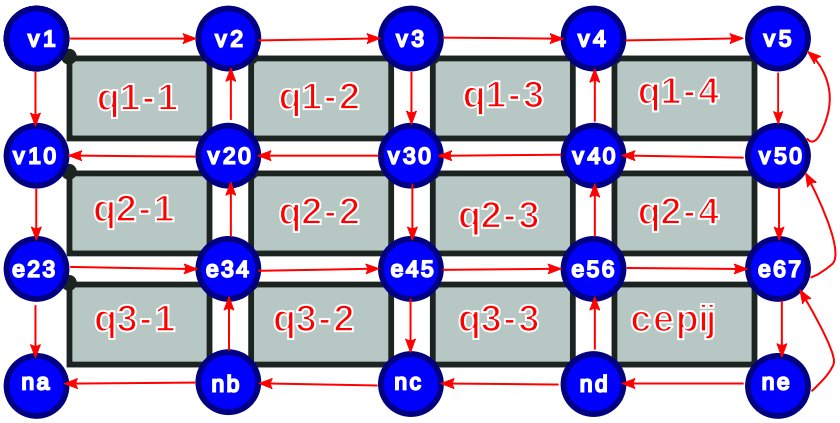


Figura 2: Representação do mapa viário: vértices nas esquinas, arestas indicando o sentido do tráfego.

Os vértices e os arcos possuem alguns atributos:

Atributo	Descrição
Vertices	
id	Um identificador (string)
x,y	Uma coordenada do segmento de rua
Arestas	
nome	Nome da rua a qual pertence o segmento
ldir	Cep da quadra que está do lado direito do segmento de rua
lesq	Idem para o lado esquerdo
cmp	comprimento (em metros) do segmento de rua
vm	Velocidade média (m/s) que os carros trafegam neste segmento de rua

O Arquivo do Mapa Viário

O arquivo do mapa viário possui o seguinte formato (os campos são separados por um espaço em branco):

`nv`	*A primeira linha do arquivo contém um número inteiro (número de vértices do grafo)*
`v id x y`	*cria o vértice* id *posicionado nas coordenadas* [x,y]
`e i j ldir lesq cmp vm nome`	*cria a aresta (i,j) e associa as outras informações à aresta. Caso a aresta não possua quadras em algum de seus lados, esta ausência é indicada por um hífen (-)*

Abaixo, um exemplo deste arquivo:

```
153
v v1 10.0 10.0
v v2 110.0 10.0
v v3 210.0 10.0
v v4 310.0 10.0
v v5 410.0 10.0
v v6 10.0 70.0
v v7 110.0 70.0
v v8 210.0 70.0
...
e v1 v6 - cep1 70.0 3.5 Rua_Belo_Horizonte
e v7 v8 cep6 cep2 100.0 4.0 Av_Higienopolis
e v8 v7 cep2 cep6 100.0 5.0 Av_10_de_Dezembro
...
```

mapa-viario1.via

Note que a especificação dos vértices sempre precedem as das arestas.

Consultas

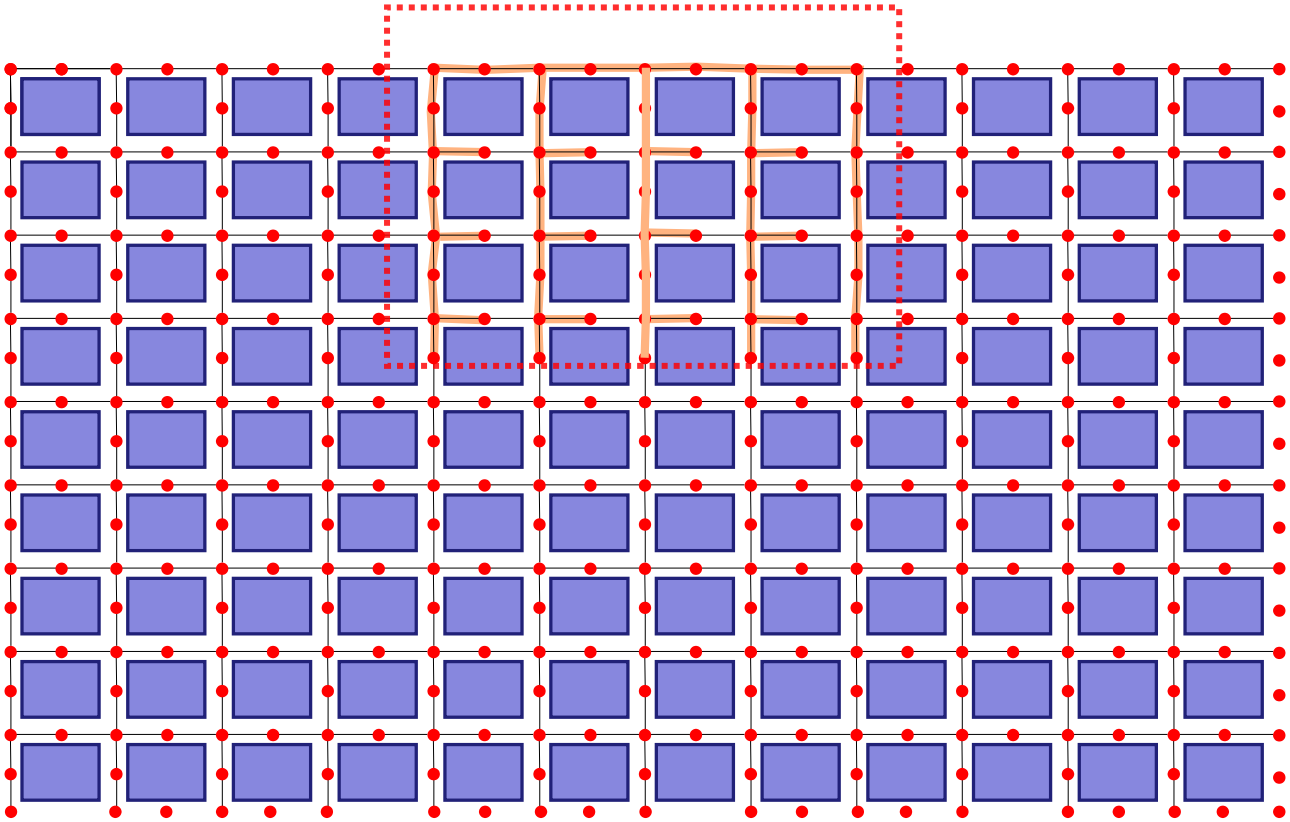


Figura 3: Exemplo de saída de um comando rv. Note as arestas da AGM destacadas.

comando	parâmetros	
`@o?`	`cep face num`	*Armazena no registrador r a posição geográfica do endereço cep/face/num* **SVG**: linha vertical com identificação do registrador.
`catac`	`x y w h`	*Remover as quadras contidas na região delimitada pelo retângulo* `x,y,w,h`. *Arestas incidentes em vértices dentro do retângulo especificado devem ser removidas.* ***TXT:*** *imprimir identificadores e dados associados a tudo o que foi removido.* ***SVG:*** *elementos removidos não devem aparecer. Desenhar sob o mapa o referido retângulo com cor de preenchimento #AB37C8, cor de borda #AA0044 e com 50% de transparência.*

rv	`x y w h f`	*Calcule a árvore geradora mínima do subgrafo com os vértices que estão dentro da região especificada pela retângulo x,y,w,h. A velocidade dos trechos relativos às arestas da árvore geradora mínima deve ser reduzida, a cada nível da árvore, de acordo com o fator f (0.0 < f < 1.0), até, no mínimo, 0.01 (ou seja, 1%). Por exemplo, se f = 0.05, a velocidade deve ser reduzida de 5% da velocidade original nas arestas que ligam a raiz da árvore aos seus filhos; de 10% nas arestas que ligam os filhos aos netos; de 15%, nas que ligam os netos aos bisnetos e assim sucessivamente.* ***TXT**: reportar arestas da AGM, incluindo seus atributos (busca em largura)* ***SVG**: pintar as arestas da AGM com um traço bastante largo e de uma cor clara. Delimitar a região x,y,w,h com um retângulo de bordas grossas e tracejadas. **ATENÇÃO**: Indicar a raiz da AGM.*
cx	`limiar`	*Calcule as regiões isoladas. Considere inacessível as regiões não conectas ou se estiverem conectadas apenas por trechos com velocidades médias abaixo do limiar especificado.* ***TXT**: reportar os identificadores dos vértices de cada região* ***SVG**: pintar de vermelho e com traço grosso os trechos com velocidade abaixo do limiar. Colocar sob os vértices, elipses de cores diferentes para cada região desconectada.*
p?	`cep face num cmc cmr`	*Qual o melhor trajeto de carro entre a origem (`@o`) e o destino especificado pelo endereço cep/face/num . `o` percurso na representação pictórica deve indicar o trajeto mais curto na cor `cmc` e o trajeto mais rápido com a cor cmr.* ***TXT**: descrição textual do percurso* ***SGV** : traçar (linhas) dos percursos com as cores correspondentes. Cuidado para que uma percurso não esconda o outro. Indicar o início e o fim do percurso. As linhas devem ser grossas, mas menos espessas que as arestas da AGM.*
Comandos do arquivo .qry		

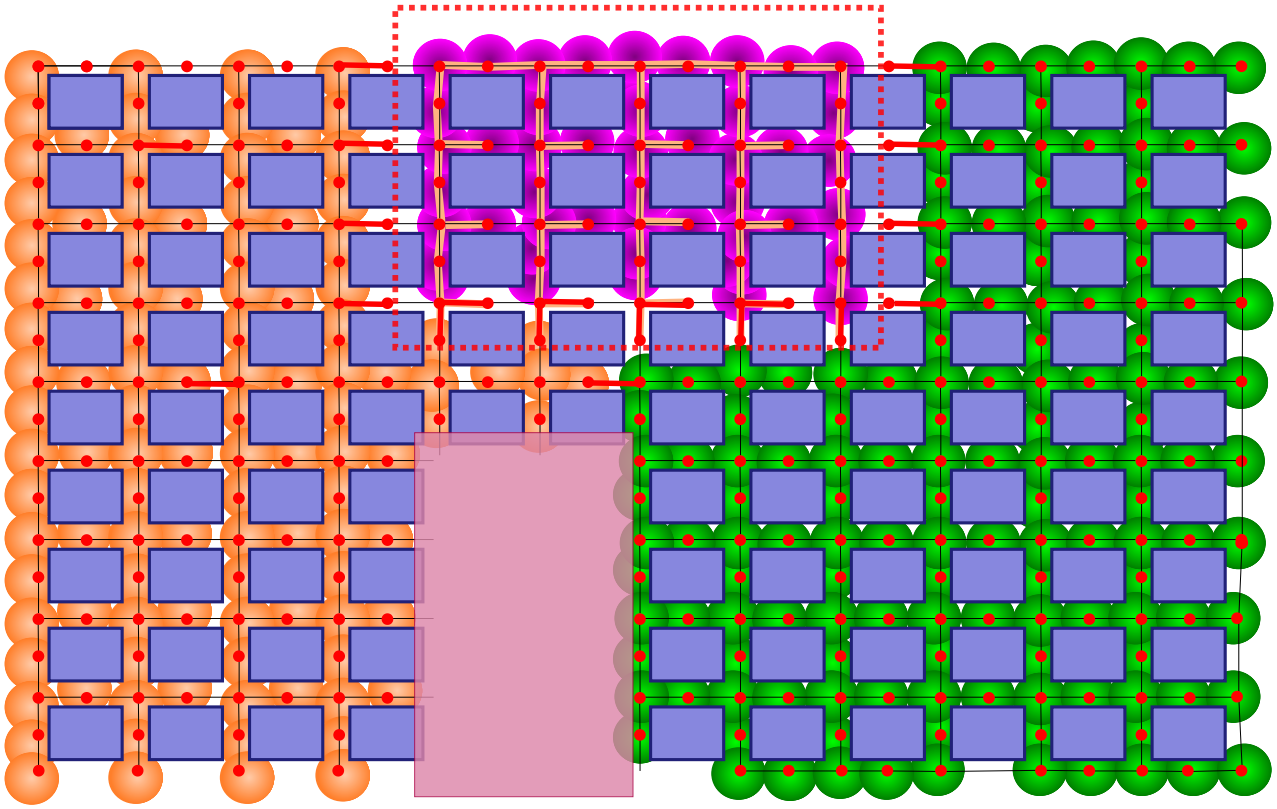


Figura 4: Resultado de operação cx

A Figura 4 mostra o resulta da operação **`cx`**. Este caso de teste é composto por uma operação **`rv`** (parte superior central) que diminuiu a velocidade média alguns trechos, por um **`catac`** (parte inferior central) que interditou outros trechos. Além disso alguns trechos originalmente já possuiam velocidade média inferior ao limiar estabelecido pelo comando **`cx`**.

A Implementação

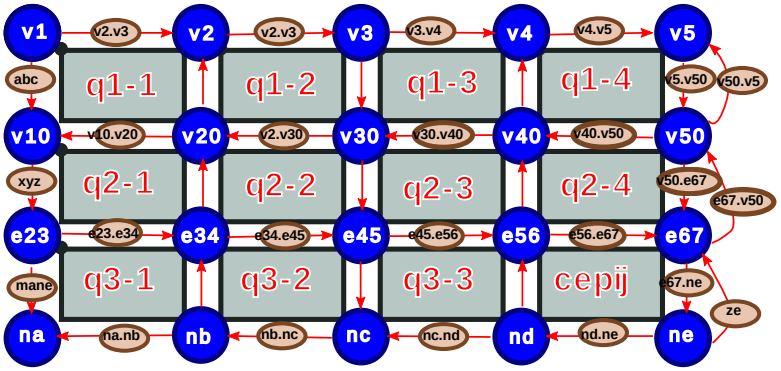
Para a determinação de caminhos mínimos **deve** ser utilizado o algoritmo de **Dijkstra**. Para a construção da Árvore Geradora Mínima **deve** ser usado o algoritmo de Kruskal.O TAD de grafo direcionado apresentado em aula **deve** ser completamente implementado com modificações que se façam necessárias. O grafo **deve** ser implementado como listas de adjacências. Uma sugestão: provavelmente, também colocar os vértices do grafo numa árvore facilite consultas espaciais.

Para determinação de regiões desconectadas, usar o algoritmo de cálculo de componentes conexos.

As diversas estruturas de dados devem ser implementadas em módulos separados e bem documentados. Continua expressamente proibido declarar structs em arquivos .h.

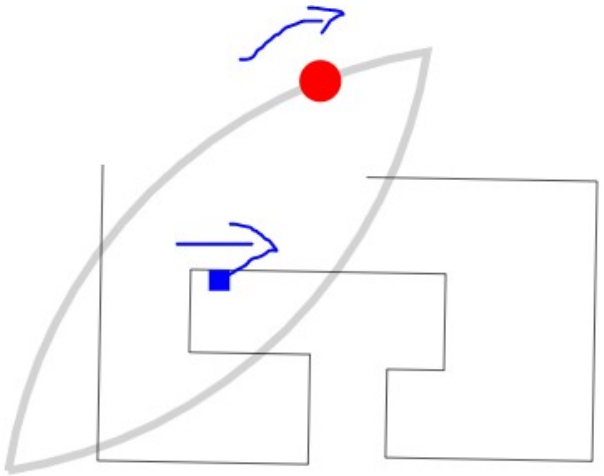
Para facilitar a implementação, alguns fatos podem ser considerados:

- Existem vértices do grafo posicionado no ponto médio das intersecções de ruas.
- Normalmente, existirão vértices no ponto médio de um segmento de rua. Isto é, o segmento de rua referente a uma face de uma quadra poderá ser, na prática, representado por duas (ou mais) arestas: (esquina1,meioquadra) e (meioquadra,esquina2). A figura abaixo ilustra onde serão posicionados os vértices extras (em lilás). Isto não altera o cálculo do percurso, apenas facilita posicionar o início e o fim do percurso.



Os percursos devem ser animados. Isto é, alguma figura (retângulo, círculo, etc)[4] deve transitar entre o início e o fim do percurso. A figura abaixo mostra um exemplo. Note que **<path/>** descreve o caminho a ser percorrido. Note o elemento **<animateMotion/>** dentro do elemento a ser animado (o círculo e o retângulo no nosso exemplo) .

ATENÇÃO: Os percursos só devem ser recalculados caso, dentro do mesmo arquivo de consulta, a origem ou o grafo tenha sido alterados.



```
<path d="M10,110 A120,120 -45 0,1 110 10 A120,120 -45 0,1 10,110"
   stroke="lightgrey" stroke-width="2"    fill="none" id="theMotionPath"/>

<path  d="m 32.539299,36.987292 -1.233889,70.589218 50.211412,0.87769 0.458653,-26.239002 -
28.795792,-0.503346 0.345472,-19.763936 60.819315,1.063112 -0.40189,22.991614 -13.58416,-0.23745 -
0.36862,21.088178 49.3119,0.86196 1.16814,-66.828091 -54.976546,-0.960982"    id="path10" />

<circle cx="" cy="" r="5" fill="red">
        <animateMotion dur="6s" repeatCount="indefinite">
           <mpath xlink:href="#theMotionPath"/>
        </animateMotion>
</circle>

<rect x="" y="" width="5" height="5" fill="blue">
        <animateMotion dur="6s" repeatCount="indefinite">
           <mpath xlink:href="#path10"/>
        </animateMotion>
</rect>
```

4 Para os valentes: desenhar um carro.

A Saída

O programa deverá produzir um arquivo **.svg** e um arquivo **.txt** ambos com o mesmo nome base do arquivo **.geo**.

Se o arquivo **.geo** (a01.geo, no nosso exemplo) for processado em conjunto com um arquivo **.qry** (q1.qry, no exemplo), além de a01.svg, deverá ser produzido o arquivo a01-q1.svg, contendo as quadras acrescidas aos resultados da consulta.

Também produzir um arquivo-texto (a01-q1.txt, no exemplo) contendo o resultado textual de todas as consultas. Neste arquivo deve ser copiado em uma linha o texto da consulta e, na linha seguinte, o seu resultado.

A Avaliação

Espera-se uma atitude pró-ativa para a aquisição dos conhecimentos (i.e., estudo) para resolver o problema proposto.

Espera-se que todos os membros trabalhem assiduamente na execução do trabalho. Em caso de dúvidas sobre a efetiva participação de um membro, o professor poderá convocar a equipe para uma entrevista. Caso o aluno não demonstre efetivo domínio do trabalho produzido, sua nota poderá ser diferente da do outro membro da equipe (que também pode ser reduzida). **Muita atenção**, neste caso, sua nota provavelmente será muito baixa (não excluída a nota nula).

A avaliação consistirá da execução dos testes e da inspeção de código. Além disso, deverá ser produzido um vídeo de, aproximadamente, 5 minutos no qual apresenta o sistema. O aluno deve ter por objetivo convencer o avaliador que: (a) o programa funciona; (b) o programa foi bem implementado. (c) o programa cumpre os requisitos especificados; (d) todos membros da equipe participaram assiduamente no desenvolvimento. Seria aconselhável mostrar alguma execução e fazer alguns comentários sobre os problemas enfretados e como foram resolvidos. O vídeo deve ser colocado no Youtube e o seu link deve estar anotado no arquivo LEIA-ME.TXT.

O Programa

O nome do programa deve ser **t2** e aceitar até cinco parâmetros:[5]

```
t2 [-e path] -f arq.geo [-q consulta.qry] [-v arqvias.via] -o dir
```

O primeiro parâmetro (**-e**) indica o diretório base de entrada. É opcional. Caso não seja informado, o diretório de entrada é o diretório corrente da aplicação. O segundo parâmetro (**-f**) especifica o nome do arquivo de entrada que deve ser encontrado sob o diretório informado pelo primeiro parâmetro. O terceiro parâmetro (**-q**) é um arquivo de consultas. O último parâmetro (**-o**) indica o diretório onde os arquivos de saída (***.svg** e ***.txt**) devem ser colocados. Note que o nome do arquivo pode ser precedido por um caminho relativo; ***dir e path*** é um caminho absoluto ou relativo (ao diretório corrente).

5 Novos parâmetros são acrescentados no decorrer do ano

A seguir, alguns exemplos de possíveis invocações de **t2**:

- `t2 -e /home/ed/testes/ -f t001.geo -o /home/ed/alunos/aluno1/o/`
- `t2 -e /home/ed -f ts/t001.geo  -o /home/ed/alunos/al1/o`
- `t2 -f ./tsts/t001.geo -e /home/ed -o /home/ed/alunos/aluno1/o/`
- `t2 -o ./alunos/aluno1/o -f ./testes/t001.geo`
- `t2 -o ./alunos/aluno1/o -f ./testes/t001.geo  -q ./t001/q1.qry`
- `t2 -e ./testes -f t001.geo -o ./alunos/aluno1/o/ -q ./q1.qry`

O Que Entregar

Submeter no Classroom o arquivo .zip com os fontes , conforme descrito anteriormente.

RESUMO DOS PARÂMETROS DO PROGRAMA SIGUEL

Parâmetro / argumento	Opcional	Descrição
`-e` ***path***	S	Diretório-base de entrada (**BED**)
`-f` ***arq*`.geo`**	N	Arquivo com a descrição da cidade. Este arquivo deve estar sob o diretório **BED**.
`-o` ***path***	N	Diretório-base de saída (**BSD**)
`-q` ***arqcons*`.qry`**	S	Arquivo com consultas. Este arquivo deve estar sob o diretório **BED**.
`-v` ***arq*`.via`**	S	Arquivo de vias (para construção do grafo)

RESUMO DOS ARQUIVOS PRODUZIDOS

-f	-q	comando com sufixo	arquivos
arq.geo			arq.svg
arq.geo	***arqcons***.qry		arq.svg arq-arqcons.svg arq-arqcons.txt
arq.geo	***arqcons***.qry	***sufx***	arq.svg arq-arqcons.svg arq-arqcons.txt arq-arqcons-sufx.[svg|txt][6]

ATENÇÃO:
* os fontes devem ser compilados com a opção `-fstack-protector-all`.
* adotamos o padrão C99. Usar a opção `-std=c99`.

6 Podem ser produzidos os respectivos arquivos .svg e/ou .txt, dependendo da especificação do comando.