

PROJETO 2020

O projeto a ser implementado durante o ano letivo é um sistema de informação geográfica simplificado, como definido abaixo.

A geographic information system (GIS) is a computer system for capturing, storing, checking, and displaying data related to positions on Earth's surface. GIS can show many different kinds of data on one map. This enables people to more easily see, analyze, and understand patterns and relationships.

Fonte: <http://www.nationalgeographic.org/encyclopedia/geographic-information-system-gis/>

O sistema será implementado incrementalmente e em fases. É importante enfatizar que em cada fase o sistema evolui, isto é, novas funcionalidades são acrescentadas, requisitos de implementação podem ser mudados, porém, as funcionalidades existentes **devem continuar funcionais**.

LEMBRETE: AVALIAÇÃO (PROGRAMA DO CURSO)

A avaliação será feita por meio de provas (P) e de trabalhos individuais (Ti) e em equipe (Te). Serão atribuídas notas numéricas às avaliações (0-10). A avaliação poderá ser feita considerando apenas o artefato produzido (texto, programa, etc), como também considerando sua apresentação oral. Espera-se que no caso de trabalhos em equipe, todos os membros trabalhem assiduamente na execução do trabalho. Em caso de dúvidas sobre a efetiva participação de um membro, o professor poderá argui-lo durante a apresentação do trabalho ou convocar a equipe para uma entrevista. Caso o aluno não demonstre efetivo domínio do trabalho produzido, sua nota poderá ser diferente da dos outros membros da equipe. Muita atenção, neste caso, sua nota provavelmente será muito baixa (não excluída a nota nula).

O peso das atividades individuais (provas e trabalhos) será 2. O peso das atividades em equipe será 1. A última atividade do ano letivo terá peso maior, visando diminuir a possibilidade de o aluno ser aprovado sem entregar/realizar a última atividade do ano. O peso da última atividade será 6, seja ela individual ou em equipe.

O que é esperado do aluno:

1. frequência em todas as aulas e pontualidade. Isto é muito importante, pois as aulas são planejadas como uma sequência de atividades que incrementalmente farão os alunos adquirirem os conhecimentos e habilidades desejadas.
2. Participação efetiva nos trabalhos. Isto também é muito importante, pois muitas habilidades e conteúdos só serão adquiridos pela prática. A forma de organização da equipe deve possibilitar que todos os membros da equipe tenham o mesmo nível de aprendizado.
3. Estudo sério da parte teórica. Esta é uma disciplina de Ciência da Computação em uma Universidade. Espera-se que o aluno dedique algumas horas semanais para o aprofundamento da teoria exposta. Espera-se que o aluno estude a parte teórica antes de que inicie a implementação da parte prática.
4. Que o aluno não use apenas as transparências para estudar. O professor disponibilizará as transparências usadas apenas como forma de orientar os estudos. As transparências, propositalmente, contém apenas o esqueleto do conteúdo e figuras para facilitar a explicação do professor. Espera-se que o aluno faz uso efetivo dos livros listados na bibliografia.

IMPORTANTE: não será tolerado nenhum tipo de fraude nas provas ou nos trabalhos. Qualquer fraude detectada durante ou após sua realização será punida com nota zero a todos os envolvidos.

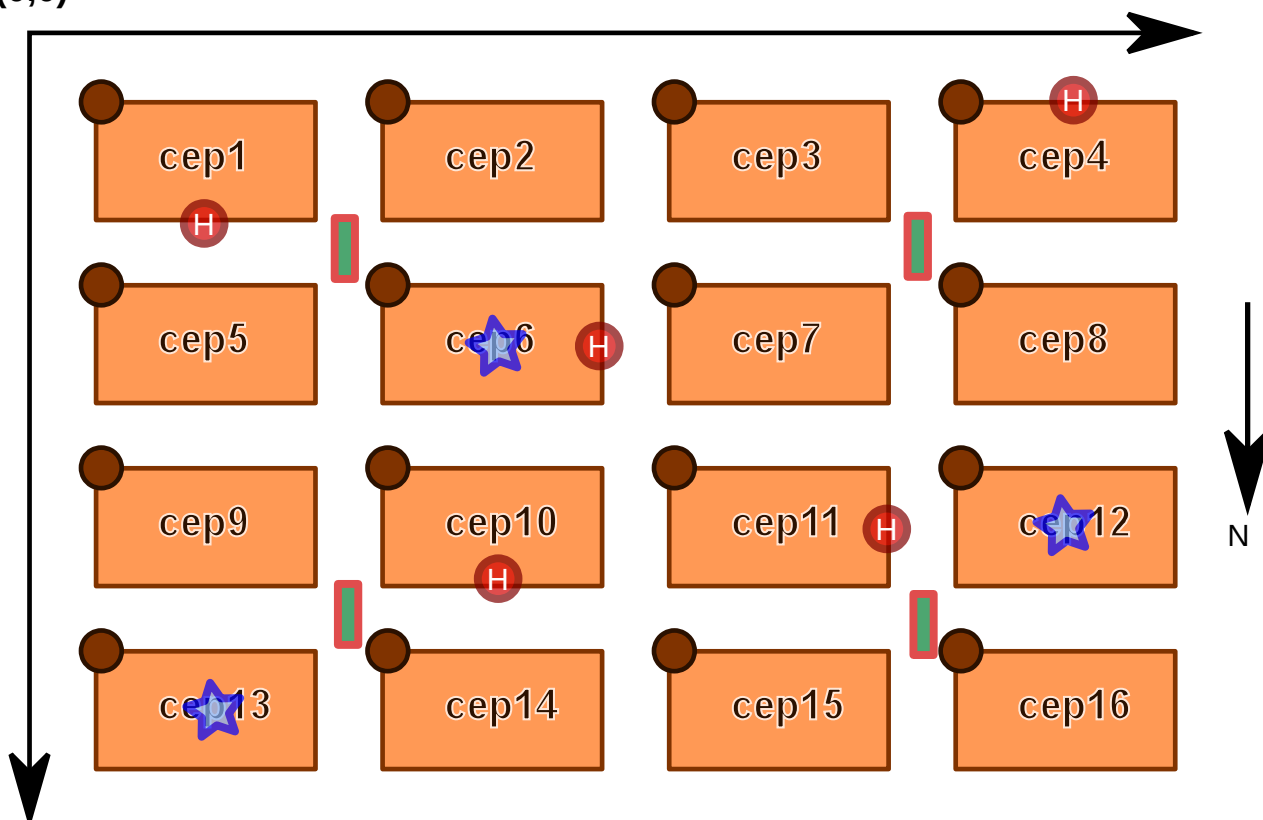
DESCRIÇÃO

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).

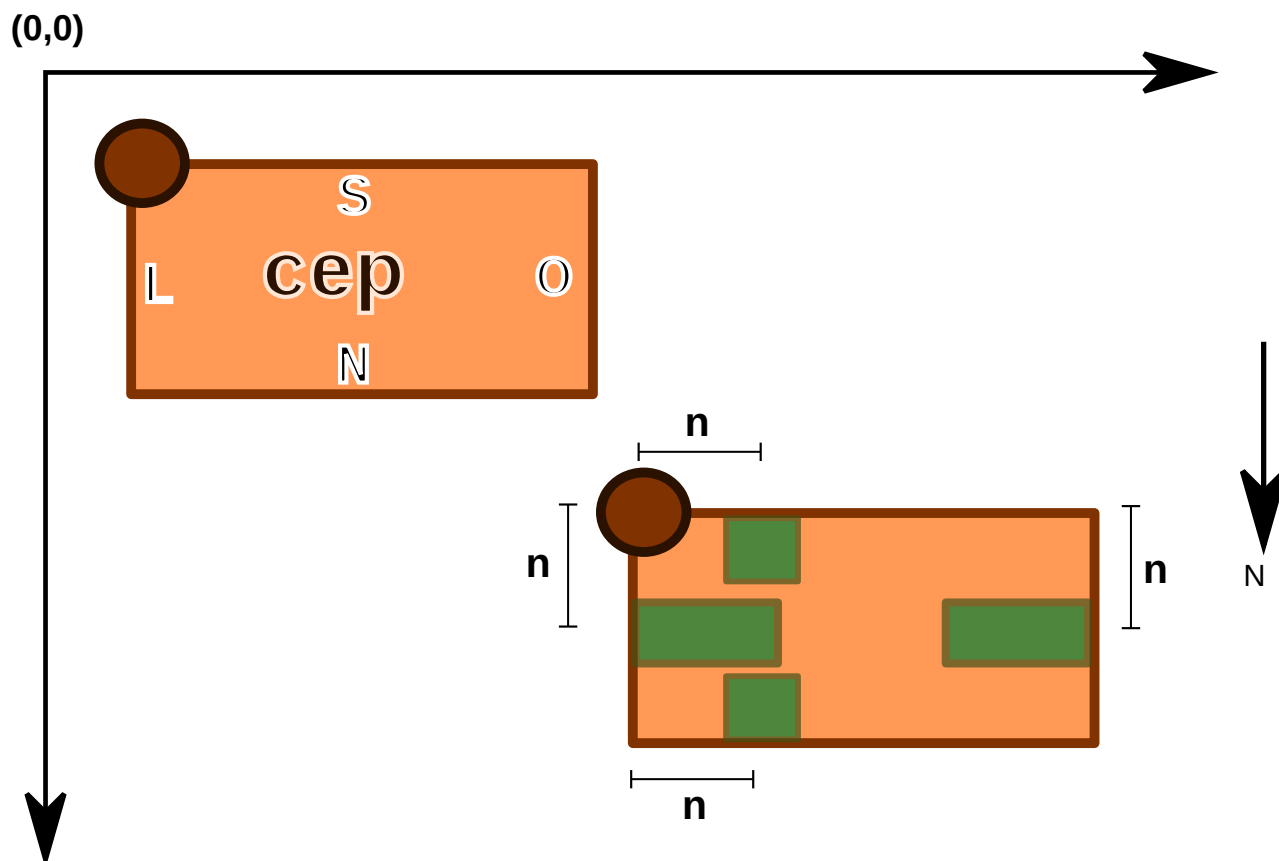
O sistema manipulará o mapa de uma cidade e algumas informações relacionadas.

O mapa de uma cidade é composto por um conjunto de retângulos que representam as quadras; e, um conjunto de equipamentos urbanos (hidrantes, semáforos, torres de celular, pontos de ônibus, etc). Cada equipamento urbano é localizado no mapa por um único ponto, conforme o exemplo abaixo.

(0,0)



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.



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

O 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:

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.

[abreviatura-nome]

LEIA-ME.txt

*

/src

makefile

*.h e *.c

*Por exemplo, josers.**colocar 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.**(arquivos-fonte)**deve ter target para a geração do arquivo objeto de cada módulo e o target **siguel** que produzirá o executável de mesmo nome dentro do mesmo diretório **src**. Os fontes devem ser compilados com a opção **-fstack-protector-all**.*** adotamos o padrão C99. Usar a opção **-std=c99**.****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 siguel**
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/>

FASE I

A Entrada

A entrada do algoritmo será basicamente um conjunto de retângulos e círculos dispostos numa região do plano cartesiano e, possivelmente, algumas consultas, por exemplo, que indagam se duas das formas geométricas se sobrepõem. Os comandos estão contidos num arquivo .geo e as consultas num arquivo .qry.

Considere a Ilustração 1. Cada forma geométrica é definida por uma coordenada âncora e por suas dimensões. A coordenada âncora do círculo é o seu centro e sua dimensão é definida por seu raio. A coordenada âncora do retângulo é seu canto inferior esquerdo⁴ e suas dimensões são sua largura e sua altura. As coordenadas que posicionam as formas geométricas são valores reais e estão contidas dentro da região delimitada pelos cantos $(0,0)$ e (x_{\max}, y_{\max}) . Cada forma geométrica é indentificada por um número inteiro.

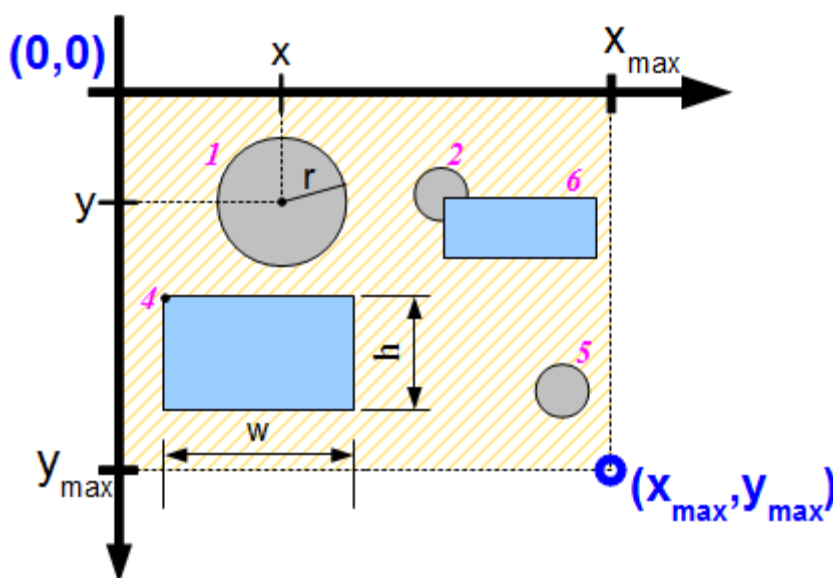


Ilustração 1

A tabelas abaixo mostram os formato dos arquivos de entrada (.geo e .qry). Os arquivos de entrada são compostos, basicamente, por conjunto de comandos (um por linha), a saber: **c** (desenhe um círculo), **r** (desenhe um retângulo), **t** (escreva um texto), etc.

Cada comando tem um certo número de parâmetros. Os parâmetros mais comuns são:

- i, j, k : número inteiro, maior ou igual a 1. Identificador de uma forma geométrica criada pelos comandos **c** ou **r**.
- r : número real. Raio do círculo.
- x, y : números reais. Coordenada (x,y) .
- cor : string. Cor válida dentro do padrão SVG.⁵

⁴ Note que o plano cartesiano está desenhado "de ponta-cabeça" em relação à representação usual.

⁵ <http://www.december.com/html/spec/colorsvg.html>.

<https://www.w3.org/Graphics/SVG/IG/resources/svgprimer.html>

comando	parâmetros	descrição
nx	i	<i>Altera o número máximo de círculos e retângulos (i.e., $c + r + t$) criados no arquivo. O valor default é 1000.</i>
c	i r x y corb corp	<i>desenhar círculo. corb é a cor da borda e corp é a cor do preenchimento</i>
r	i w h x y corb corp	<i>desenhar retângulo: w é a largura do retângulo e h, a altura. corb é a cor da borda e corp é a cor do preenchimento</i>
t	i x y corb corp txto	<i>desenha o texto txto nas coordenadas (x,y) e com a cores indicadas. corb é a cor da borda e corp é a cor do preenchimento</i>
comandos .geo		

comando	parâmetros	descrição
o?	j k	<i>As formas geométricas cujos identificadores são j e k se sobrepõem?⁶ Saída: .txt: copiar o comando e, na linha seguinte escrever uma mensagem informando o tipo das formas j e k e se sobrepõe ou não .svg: traçar um retângulo que envolva ambas figuras: de bordas tracejadas, se não se sobrepõem; linha cheia, se se sobrepõem</i>
i?	j x y	<i>O ponto (x,y) é interno à j-ésima forma geométrica?⁷ Saída: .txt: copiar o texto da consulta e, na linha seguinte, informar o tipo da forma geométrica e se é interno ou não. .svg: colocar um ponto (pequeno círculo) nas coordenadas (x,y) e pintá-lo de azul se for interno e magenta se for externo. Colocar uma linha (da mesma cor do ponto) ligando o ponto ao centro de massa da figura j.</i>

6 A borda da figura pertence à figura. Assim, as figuras que coincidem apenas nas bordas também se sobrepõem.

7 Um ponto na borda da figura pertence à figura, **mas não é interno** à figura.

comando	parâmetros	descrição
pnt	j corb corp	Mudar as cores da borda (corb) e do preenchimento (corp) da forma/texto de identificador j. <i>Saída: .txt:</i> copiar o texto da consulta e, na linha seguinte, informar as coordenadas originais da forma/texto. <i>.svg:</i> a figura afetada é mostrada com as novas cores
pnt*	j k corb corp	Semelhante ao comando pnt, aplicado às formas/texto entre os identificadores j e k, inclusive.
delf	j	Remove a forma/texto de idenficador j. <i>Saída: .txt:</i> copiar o texto da consulta e mostrar todas as informações sobre esta forma/texto. <i>.svg:</i> o elemento removido não deve aparecer no svg.
delf*	j k	Semelhante ao comando delf, aplicado às formas/texto entre os identificadores j e k, inclusive.
Comandos .qry		

A figura abaixo mostra um exemplo de um arquivo de entrada (consistente com a Ilustração 1). Note que a extensão do arquivo é **.geo**. As primeiras operações desenham círculos e retângulos.

```
c 1 50.00 50.0 30.00 grey magenta
r 6 121.0 46.0 100.0 30.0 cyan yellow
c 2 grey magenta 120.0 45.0 15.0
r 4 10.0 150.0 90.0 40.0 cyan yellow
c 5 230.0 180.0 13.0 grey magenta
```

a01.geo

```
o? 2 6
i? 5 210.0 160.0
```

q.qry

A Saída

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

O arquivo .svg produzido deve mostrar as formas geométricas. Além disso, para o comando

o, caso as figuras identificadas se sobreponham, elas devem ser envolvidas por um retângulo tracejado contendo a palavra "sobrepoe". Existem várias ferramentas que renderizam arquivos .svg. As figuras abaixo mostram um exemplo de arquivo .svg e sua respectiva renderização.

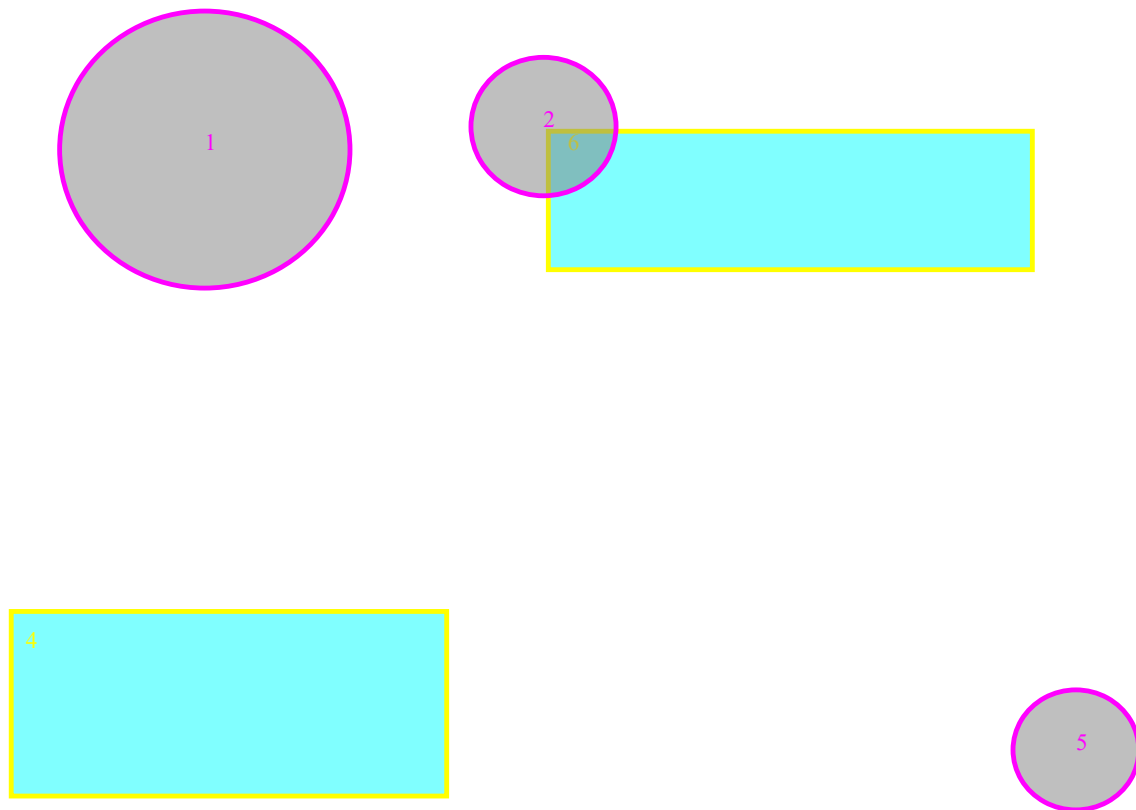


Illustration 2: Arquivo a01.svg

Se o arquivo .geo (a01.geo, no nosso exemplo) for processado em conjunto com um arquivo .qry (q.qry, no exemplo), além de a01.svg, deverá ser produzido o arquivo a01-q.svg, contendo os círculo, retângulos, etc acrescentados aos resultados da consulta.

Também produzir um arquivo-texto (a01-q.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.

```
o? 2 6
2: retângulo 6: círculo SIM

i? 5 210.0 160.0
5: círculo NAO INTERNO
```

Arquivo a01-q.txt

FASE II

Nesta fase serão acrescentados quadras e equipamento urbanos. A nossa cidade começa a tomar forma. Temos quadras, hidrantes, radio-bases de telefonia móvel e semáforos. As quadras são identificadas por seu CEP; os hidrantes, semáforos e rádio-bases são identificadas por um código alfa-numérico (por exemplo, *h-1234.ns_abc23*).

As quadras ocupam uma região retangular, enquanto que os hidrantes, semáforos e rádio-bases são representadas por um ponto (coordenada) no mapa.

Nesta fase serão acrescentados novos comandos ao arquivo **.geo**:

comando	parâmetros	
q	cep x y w h	<i>Insere uma quadra (retângulo e cep)</i>
h	id x y	<i>Insere um hidrante</i>
s	id x y	<i>Insere um semáforo</i>
rb	id x y	<i>Insere uma rádio-base (torre de celular)</i>
cq	sw cfill cstrk	<i>Cores do preenchimento (cfill) e da borda (cstrk) das quadras, espessura da borda (sw) (a partir deste comando)</i>
ch	sw cfill cstrk	<i>Cores do preenchimento e da borda (e sua espessura) dos hidrantes (a partir deste comando)</i>
cr	sw cfill cstrk	<i>Cores do preenchimento e da borda (e sua espessura) das torres de celular (a partir deste comando)</i>
cs	sw cfill cstrk	<i>Cores do preenchimento e da borda (e sua espessura) dos semáforos (a partir deste comando)</i>
sw	cw rw	<i>Espessuras das bordas, respectivamente, dos círculos (comando c) e dos retângulos (comando r) (a partir deste comando)</i>
nx	i nq nh ns nr	<i>Informa o número máximo de formas(círculos e retângulos), quadras, hidrantes, semáforos e rádio-bases, respectivamente, criados no arquivo. O valor default é 1000.</i>
Novos comandos do arquivo .geo		

Alguns comandos de atualização e consulta podem ser colocados em um arquivo **.qry**:

comando	parâmetros	
dq	[#] id r	<p>Remove todas quadras que estiverem inteiramente dentro a uma distância de no máximo r do equipamento urbano identificado por id. O parâmetro # pode estar ausente.</p> <p>No arquivo .svg: Se o parâmetro # estiver ausente, as quadras removidas não devem aparecer; caso contrário, as quadras removidas devem ter cantos arredondados,⁸ ser pintadas de “beige” (fundo) e “olive” (borda). O equipamento urbano em questão deve enfatizado com um anel grosso de duas cores. Desenhar o círculo de raio r, mostrando a região de delição.</p> <p>No arquivo .txt: deve apresentar os ceps das quadras removidas, o id e respectivas informações do equipamento urbano.</p>
del	(cep id)	<p>Remove a quadra, hidrante, semáforo ou torre de identificação id (ou cep).</p> <p>No arquivo .svg: quadra ou equipamento urbano removido não deve aparecer. Deve ser colocada uma linha vertical com início no centro do elemento removido até o topo do mapa. Também colocar (no topo) ao lado da linha vertical o cep ou o id.</p> <p>No arquivo .txt: reportar os dados relacionados da quadra ou equipamento urbano removido.</p>
cbq	x y r cstrk	<p>Muda a cor da borda para cstrk de todas as quadras que estiverem inteiramente contidas dentro do círculo de centro em (x,y) e de raio r.</p> <p>No arquivo .svg: quadras eleitas pintadas conforme descrito.</p> <p>Reporta no arquivo .txt o cep das quadras que tiveram a cor da borda alterada</p>
crd?	(cep id)	<p>Imprime no arquivo .txt as coordenadas e a espécie do equipamento urbano de um determinado cep ou com uma determinada identificação.</p>

⁸ Veja atributo rx de <rect>

comando	parâmetros	
car	x y w h	<p>Calcula a área total das quadras e equipamentos urbanos que estiverem inteiramente dentro do retângulo (x,y,w,h). No arquivo .svg: desenhar o retângulo da consulta e traçar uma linha vertical do ponto (x,y) até o topo do mapa. No topo do mapa, ao lado da linha vertical, escrever a área total. No centro de cada quadra selecionada, escrever a sua respectiva área. No arquivo .txt: escrever os ceps das quadras selecionadas e suas respectivas áreas. Escrever a área total.</p>
Comandos do arquivo .qry		

EXEMPLOS

```

cq 2px blue black
ch 1px red yellow
ct 3px black red
q cep_001-10 37.00 15.00 89.00 40.00
q cep_001-20 137.00 15.00 89.00 40.00
q cep_001-30 237.00 15.00 89.00 40.00
cq 1.5px yellow green
q cep_002-10 37.00 115.00 89.00 40.00
q cep_002-20 137.00 115.00 89.00 40.00
q cep_002-30 237.00 115.00 89.00 40.00
h h-12 320.00 60.00
c 3 29.00 720.00 458.00 green yellow
r 4 5.00 29.00 1049.00 479.00 green blue
sw 3.4px 1.3px
r 5 55.00 42.00 215.00 702.00 green red
c 6 51.00 139.00 635.00 chocolate black

```

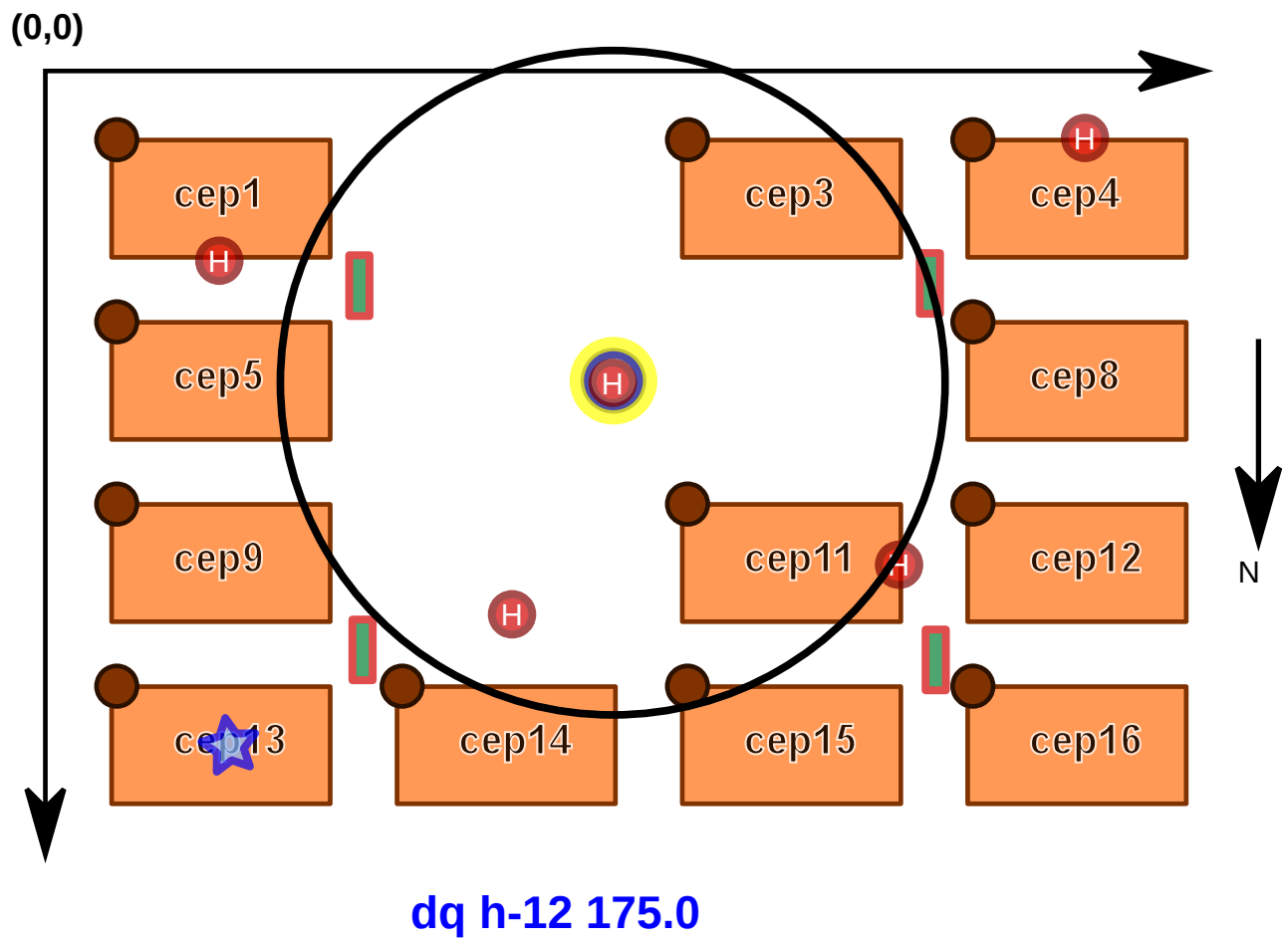
a1.geo

```

dq h-12 120.0
crd? cep_001-10
crd? h-12

```

q1.qry



FASE III

A COVID-19 chegou a Bitnópolis. A fim de evitar a propagação da doença, regiões são categorizadas por grau de incidência(casos/100000 habitantes). Dependendo da categoria, diferentes restrições são aplicadas. As categorias são descritas na tabela abaixo:

Categoria	Incidência (casos/100k hab)	Descrição
A	< 0.1	Livre de Covid. Cor: 00FFFF
B	< 5	Baixa incidência. Cor: 008080
C	< 10	Média incidência. Cor: FFFF00
D	< 20	Alta incidência Cor: FF0000
E	≥20	Catastrófico. Cor: 800080

Para facilitar o atendimento, alguns postos de saúde de campanha foram distribuídos pela cidade. É importante que todas as regiões de categoria E possuam, pelo menos, um posto de campanha em seu interior. Uma de suas tarefas é determinar se as regiões desta categoria efetivamente possuem algum posto de campanha. Em caso negativo, você deve sugerir que um posto seja colocado no centroide da região.⁹

A cidade é dividida em regiões e cada região possui uma determinada densidade demográfica.

comando	parâmetros	
ps	x y	Coloca um posto na coordenada (x,y). <i>SVG: colocar um ponto na coordenada</i>
dd	x y w h d	A densidade demográfica da região delimitada pelo retângulo x,y, w,h é de d habitantes por km ² . ¹⁰
Novos comandos do arquivo .geo		

comando	parâmetros	
cv	n cep face num	Foram detectados n casos de COVID-19 no endereço cep,face,num. <i>SVG: colocar um pequeno quadrado (cor laranja) no endereço com o número n dentro (cor branca).</i>

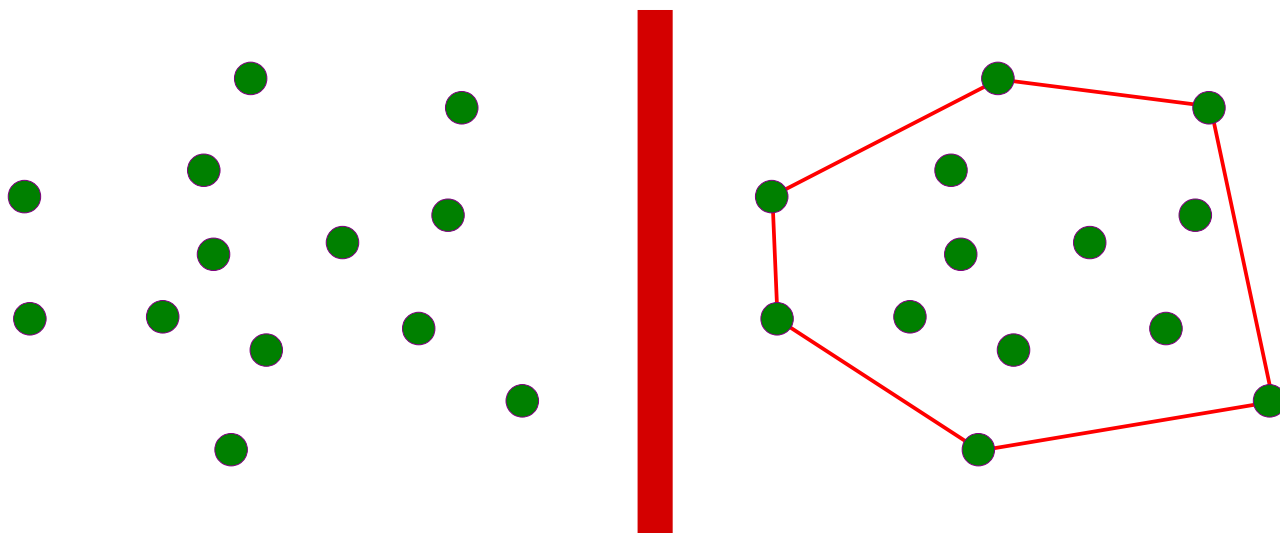
⁹ Veja: <http://dan-scientia.blogspot.com/2009/10/centroide-de-um-poligono.html>

¹⁰ Considerar que as medidas dos comandos equivalem a metros. Por exemplo, se a largura de uma quadra for 100.0, considere 100.0 metros.

soc	k cep face num	<p>Morador do endereço cep,face,num precisa de atendimento. Determine os k postos de atendimento mais próximos.</p> <p>SVG: colocar um pequeno quadrado azul com bordas brancas no endereço. Traçar um segmento de reta tracejada do endereço até cada um dos k postos de atendimento.</p> <p>TXT: escrever as coordenadas dos postos de atendimento</p>
ci	x y r	<p>Determinar a região de incidência relativa aos casos (comando cv) reportados dentro da região delimitada pela circunferência de centro em (x,y) e raio r. Determinar a categoria da região. Se necessário sugerir um posto de atendimento no centróide da região.</p> <p>SVG: Desenhar o círculo sem preenchimento e com borda grossa verde. Desenhar a região de incidência (envoltória convexa). Usar borda grossa na cor vermelha e preencher com fundo transparente na cor relativa à respectiva categoria.</p> <p>TXT: coordenadas dos casos selecionados pelo círculo, o número total de casos, a área dentro da envoltória convexa e a categoria de incidência.</p>
Novos comandos do arquivo .gry		

ENVOLTÓRIA CONVEXA

A envoltória convexa de um conjunto Q de pontos num plano, denotada por $CH(Q)$, é o menor polígono convexo P para o qual cada ponto em Q está no contorno de P ou em seu interior.



O algoritmo para calcular a CH de um conjunto de pontos é muito simples e depende basicamente de uma ordenação particular destes pontos. Para fazer esta implementação, deve ser utilizado o algoritmo de ordenação quicksort.

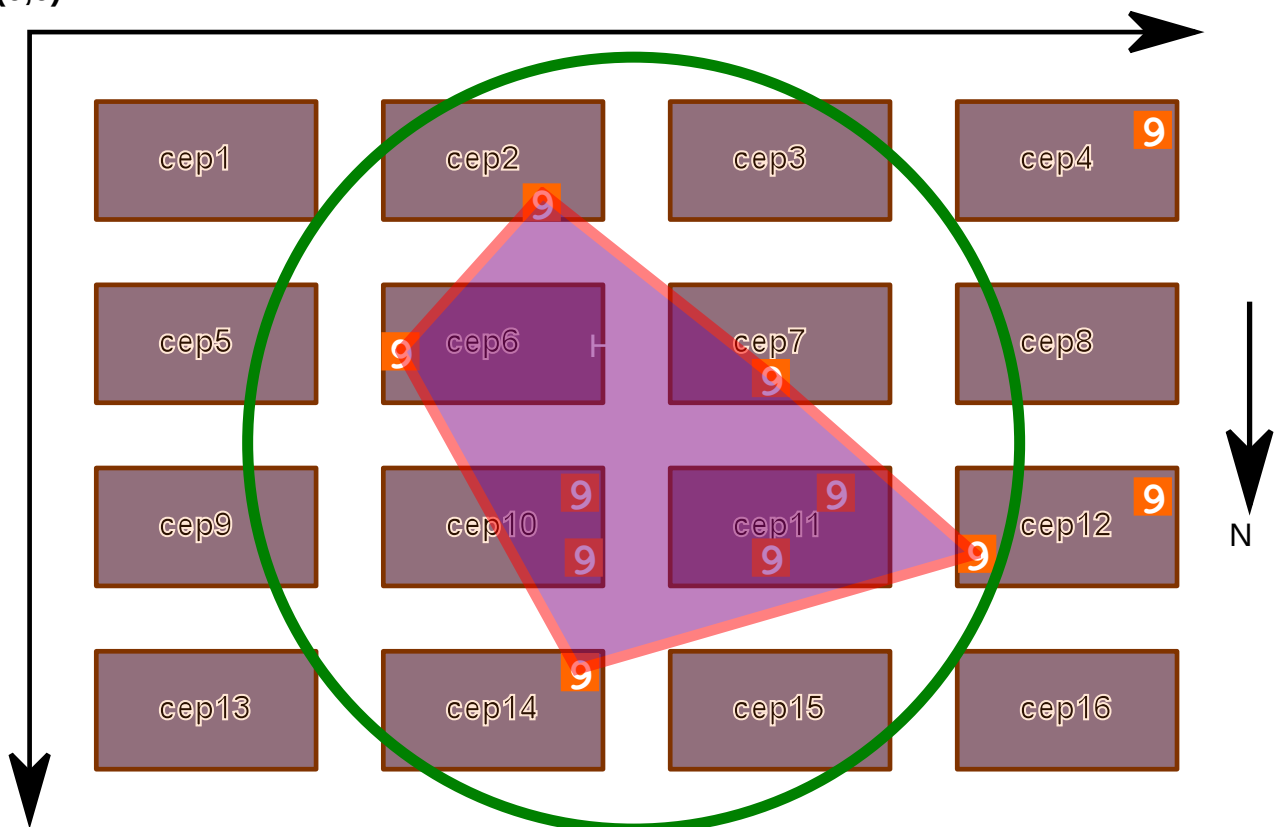
Breves explicações sobre um dos algoritmos (Graham Scan) para o cálculo da envoltória convexa podem ser facilmente encontrados no YouTube usando a consulta *convex hull algorithm, computational geometry, graham scan*.

ÁREA DE POLÍGONOS IRREGULARES

Para determinar a categoria de incidência de uma determinada região é necessário calcular a área daquela região. O algoritmo de Gauss é um algoritmo bastante simples para tal fim.

EXEMPLO DE SAÍDA PARA O COMANDO ci

(0,0)



FASE IV

Nossa cidade já possui quadras e equipamentos urbanos. Agora começaremos a povoá-la com moradores e estabelecimentos comerciais.

Esta fase consiste em trocar as estruturas de dados que armazenam as informações da nossa cidade. Basicamente, as operações de inserção, consulta, modificação e remoção devem ser feitas usando as operações da nova estrutura.

O comando **dd** (densidade demográfica) deve ser alterado. Agora a densidade demográfica passa a se referir às quadras inteiramente dentro da região especificada. Na saída .svg, devem ser incluídas sombras nas quadras.¹¹ A cor da sombra depende da densidade demográfica da quadra, segundo a tabela abaixo.



	habitantes/km ²
#FFFF00	10-500
#FF9955	500-1500
#FF0000	1500-3000
#FF00CC	3000-4500
#6600FF	4500-6000
#A02C5A	Maior que 6000



Desenho 1: Retângulo com Sombra

¹¹ Sombras são facilmente incluídas com a utilização de filtros do tipo `feDropShadow`. Veja <https://developer.mozilla.org/en-US/docs/Web/SVG/Element/feDropShadow>

ARQUIVOS DE ENTRADA

A seguir, são apresentados novos arquivos de entrada que deverão ser processados

Estabelecimentos Comerciais

comando	parâmetros	
t	codt descricao	<i>Define tipo de estabelecimento comercial</i>
e	cnpj cpf codt cep face num nome	<i>Insere um novo estabelecimento comercial de um determinado tipo (codt), localizado em um dado endereço (cep,face,num), que possui um dado cnpj, tem um dado nome e cujo proprietário é identificado pelo cpf.</i>
Comandos do arquivo .ec (-ec)		

Pessoas e moradores

comando	parâmetros	
p	cpf nome sobrenome sexo nasc	<i>Insere pessoa identificada por cpf, nomeada (nome,sobrenome), de um certo sexo (M,F), nascida numa determinada data (dd/mm/aaaa)</i>
m	cpf cep face num compl	<i>Informa que um dada pessoa (cpf) mora num dado endereço (cep,face,num,compl)</i>
Comandos do arquivo .pm (-pm)		

Consultas

comando	parâmetros	
m?	cep	<i>Moradores da quadra cujo cep é cep. Mostra mensagem de erro se quadra não existir. TXT: listar todos os dados dos moradores(nome, endereço,...)</i>
dm?	cpf	<i>Imprime todos os dados do morador identificado pelo cpf. TXT: dados pessoais, seu endereço SVG: colocar uma linha vertical do endereço do morador até a margem superior do mapa. Anotar ao lado da linha o cpf, nome e endereço do morador</i>

de?	cnpj	<p>Imprime todos os dados do estabelecimento comercial identificado por cnpj.</p> <p>TXT: dados do estabelecimento (nome, descrição do tipo, etc) e dados de seu proprietário.</p>
mud	cpf cep face num compl	<p>A pessoa identificada por cpf muda-se para o endereço determinado pelos parâmetros.</p> <p>TXT: Mostrar os dados da pessoa (nome, etc), o endereço antigo e o novo endereço.</p> <p>Arquivo.</p> <p>SVG: Uma linha vermelha grossa do endereço antigo ao endereço novo. Um pequeno círculo vermelho no endereço antigo, outro círculo azul no novo endereço. Ambos círculos com borda branca grossa.</p>
dmprrbt	t sfx	<p>Imprime o estado atual de uma árvore no arquivo <arq>-sfx.svg.</p> <p>t informa qual das árvores:</p> <p>q: quadras; h: hidrantes; s: semáforos; t: torres; p: postos de saúde</p> <p>Cada nó da árvore deve mostrar as coordenadas do elemento armazenado e alguma informação relevante sobre tal elemento (p.ex, o cep da quadra, o identificador do hidrante, etc)</p>
eplg?	[tp *] x y w h	<p>Estabelecimentos comerciais do tipo tp (ou de qualquer tipo, caso *) que estão inteiramente dentro da região retangular especificada.</p> <p>TXT: dados sobre os estabelecimentos comerciais, incluindo o nome de seu proprietário.</p> <p>SVG: Destacar o estabelecimento comercial selecionado.</p>
catac	x y r	<p>Remover as quadras, postos de saúde, elementos urbanos, moradores e estabelecimentos comerciais que estejam inteiramente contidas na circunferência de raio r e centro em x,y</p> <p>TXT: imprimir identificadores e dados associados de tudo o que foi removido.</p> <p>SVG: elementos removidos não devem aparecer. Desenhar sob o mapa o referido círculo com cor de preenchimento #CCFF00, cor de borda #6C6753 e com 50% de transparência.</p>

Novos comandos do arquivo .qry

QUADTREE

A **quadtree** is a tree data structure in which each internal node has exactly four children. Quadtrees are the two-dimensional analog of octrees and are most often used to partition a two-dimensional space by recursively subdividing it into four quadrants or regions. The data associated with a leaf cell varies by application, but the leaf cell represents a "unit of interesting spatial information".

Point quadtree

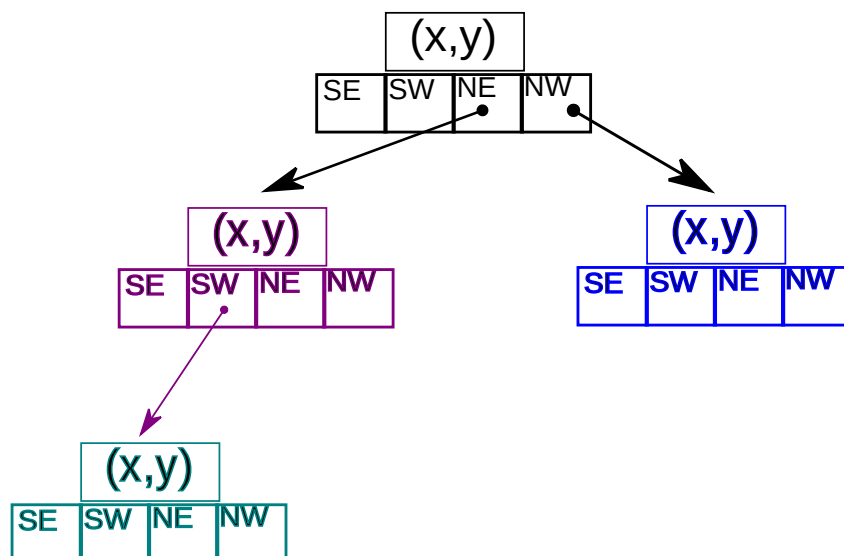
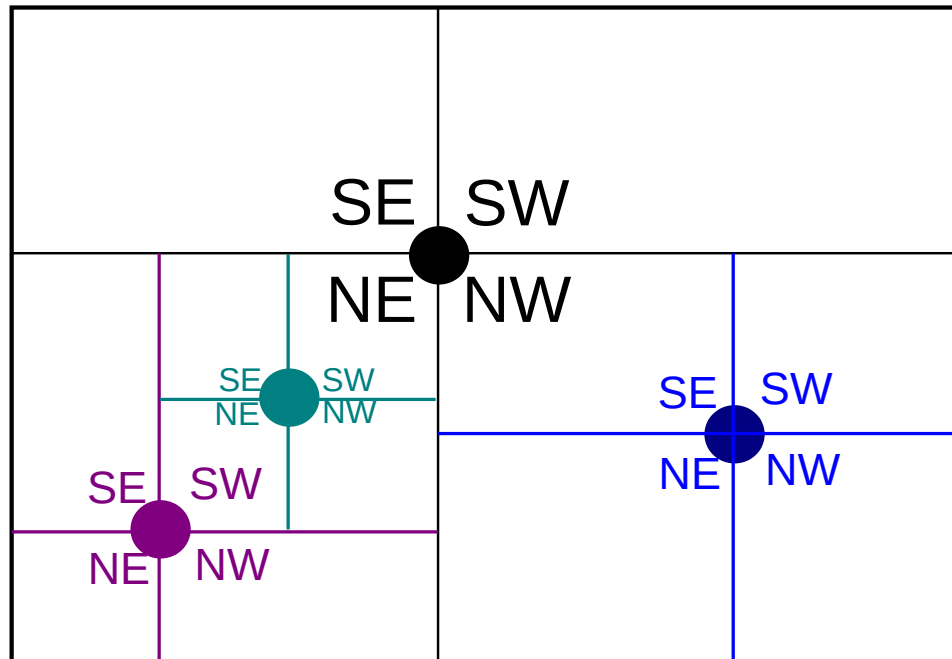
The point quadtree is an adaptation of a binary tree used to represent two-dimensional point data. It shares the features of all quadtrees but is a true tree as the center of a subdivision is always on a point. It is often very efficient in comparing two-dimensional, ordered data points, usually operating in $O(\log n)$ time.

Point quadtrees are constructed as follows. Given the next point to insert, we find the cell in which it lies and add it to the tree. The new point is added such that the cell that contains it is divided into quadrants by the vertical and horizontal lines that run through the point. Consequently, cells are rectangular but not necessarily square. In these trees, each node contains one of the input points.

Since the division of the plane is decided by the order of point-insertion, the tree's height is sensitive to and dependent on insertion order. Inserting in a "bad" order can lead to a tree of height linear in the number of input points (at which point it becomes a linked-list). If the point-set is static, pre-processing can be done to create a tree of balanced height.

Trechos copiados de Wikipedia (<https://en.wikipedia.org/wiki/Quadtree>)

A figura abaixo mostra alguns pontos distribuídos no plano. Um ponto, ao ser inserido, divide uma região do plano em quatro regiões, a saber, sudeste, sudoeste, nordeste e noroeste. A figura também mostra a quadtree relativa àqueles pontos.

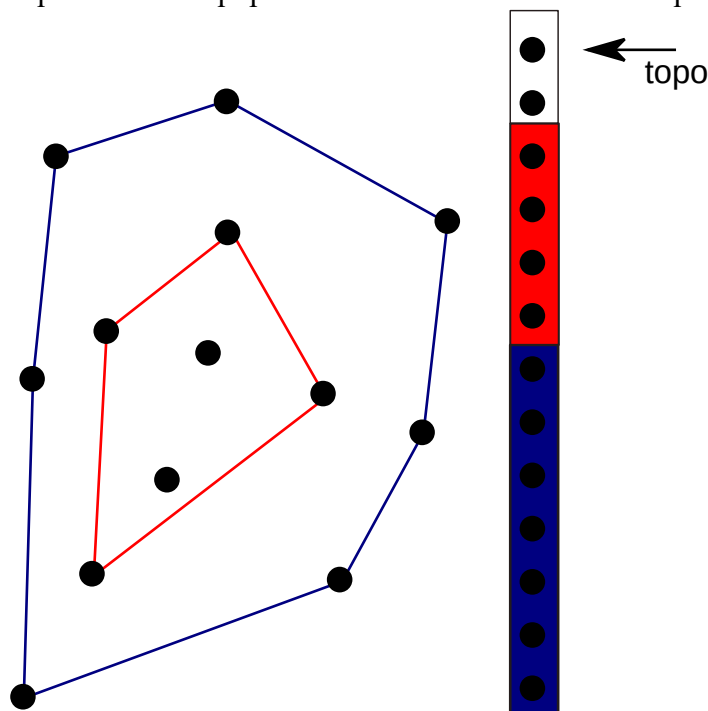


Como mencionado acima, a estrutura de uma quadtree depende da ordem de inserção dos pontos (no exemplo, o ponto preto foi o primeiro a ser inserido). Talvez a estratégia de inserção descrita a seguir possa produzir árvores razoavelmente balanceadas.

Considere a figura abaixo. O polígono azul é a envoltória convexa do conjunto de pontos. O polígono vermelho é a envoltória excluídos os pontos da primeira envoltória convexa, e assim sucessivamente. Note que, se inserirmos na quadtree os pontos na ordem inversa em que fizeram parte de alguma dessas envoltórias, provavelmente produzir-se-á uma quadtree razoavelmente

balanceada.¹²

O aluno deverá avaliar se esta estratégia de inserção é melhor (e em que medida) que simplesmente inserir as quadras e os equipamentos urbanos na ordem em que aparecem no arquivo



ÍNDICES

A fim de facilitar algumas operações e consultas, o sistema utiliza índices. Para o nosso propósito, neste momento, índices são atalhos. Mais precisamente, são informações sobre os dados armazenados que facilitam sua localização. Por exemplo, a figura abaixo mostra duas tabelas. A tabela azul à esquerda (tabela principal) mostra uma agenda telefônica ordenada pelo nome. Neste caso, é fácil encontrar o telefone da "Guilhermina", devido à ordenação. O reverso, isto é, dado um telefone determinar o nome do assinante, é mais difícil.

A tabela à direita (amarela) é uma tabela auxiliar (índice). Ela contém os mesmos números de telefones da tabela principal e uma indicação da linha onde este telefone ocorre naquela tabela. Por exemplo, o telefone 32147530 (primeira linha da tabela amarela) ocorre na linha 5 da tabela principal; assim, determina-se facilmente que tal telefone é da Clarissa.

¹² Você concorda que esta estratégia tende a produzir árvores mais balanceadas? Como você imagina que seria a estrutura de uma quadtree de quadras se elas forem inseridas na ordem em que aparecem no arquivo .geo?

	NOME	FONE		FONE	POS
1	Alvaro	13434365	**	32147530	5
2	Ana	32315623		32315623	2
3	Andre	56783456		32569872	20
4	Carlos	43670987		43670987	4
5	Clarissa	32147530	**	56783456	3
6	Domenica	98795432		58769987	7
7	Eleutério	58769987		75640967	16
8	Fabiana	78654320		76900098	17
9	Fábio	76989876		76989876	9
10	Guilherme	84563589		78654320	8
11	Guilhermina	80453587		80453587	11
12	Hélio	91997204		83434365	1
13	José	91676300		84563589	10
14	Josefina	89099788		89099788	14
15	Laura	89998742		89449922	19
16	Marcos	75640967		89766677	18
17	Maria	76900098		89998742	15
18	Nair	89766677		91676300	13
19	Norberto	89449922		91997204	12
20	Ximena	32569872		98795432	6

A IMPLEMENTAÇÃO

Deve ser implementado um módulo para o tipo abstrato Tabela de Espalhamento. A implementação da tabela de espalhamento deve fazer uso da lista implementada (conforme solicitado) nos trabalhos anteriores.

Todos os elementos georreferenciados devem armazenados em quadrees. Tabelas de espalhamento devem ser usadas como dicionários e como índices. Um dicionário, por exemplo, pode (deve) ser usado para manter a tabela de tipos de estabelecimentos comerciais (tipo X descrição). Vários índices deverão ser usados para facilitar as consultas.

Índices/Dicionários que devem ser mantidos e usados em consultas:

- `cpf X cep`: dado o `cpf`, retorna o `cep` quadra onde a pessoa reside
- `tipo-estabelecimento-comercial X descrição`: dado o código do tipo do estabelecimento, retorna a descrição
- `dados de uma pessoa`: dado um `cpf`, retorna os dados pessoais daquela pessoa
- `cep X quadra`: dado um `cep`, retorna os dados da respectiva quadra

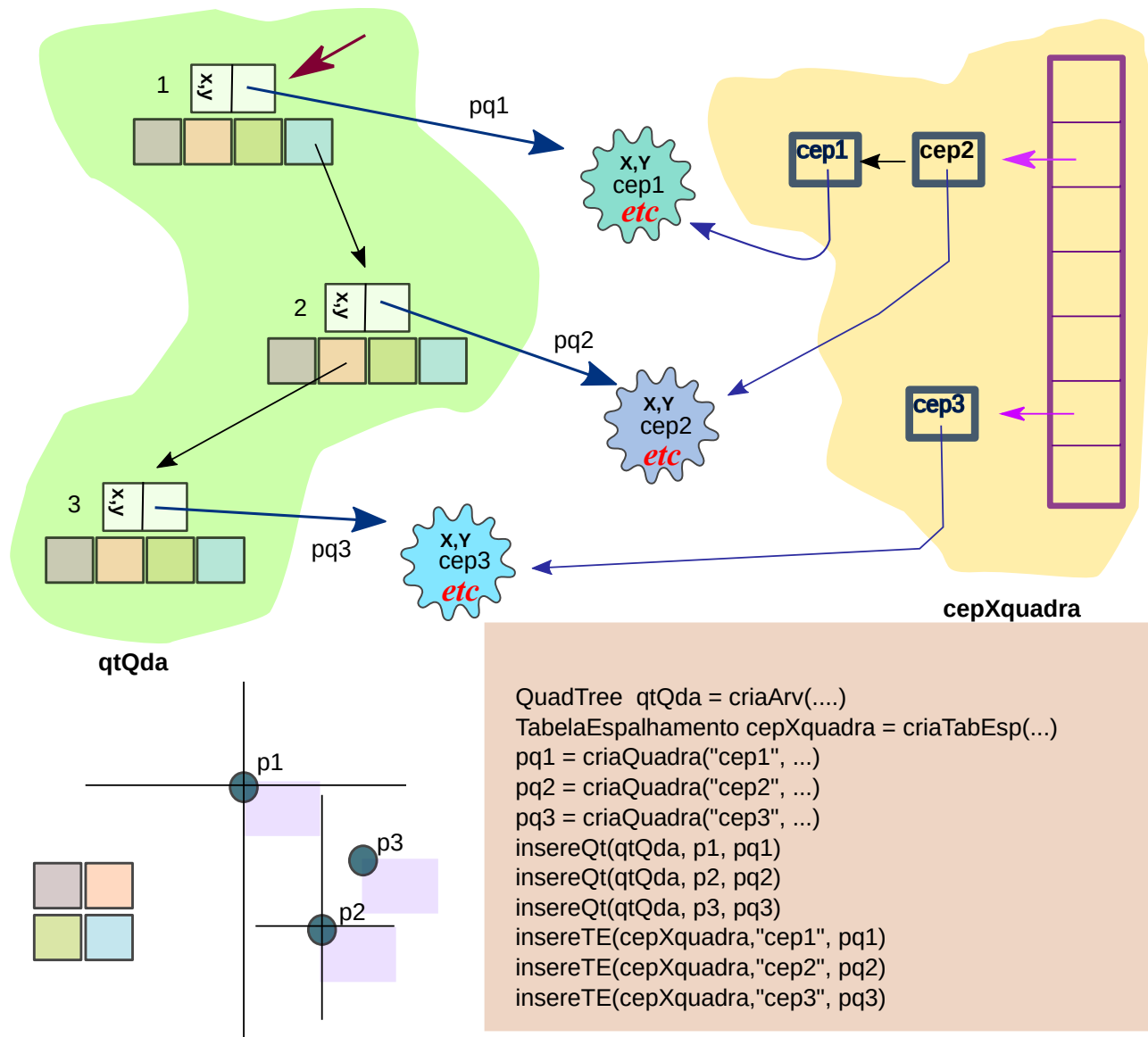
A implementação da QuadTree **deve** estar de acordo com o arquivo `quadtree.h`, disponibilizado pelo professor. Note que esta implementação depende do módulo **Lista** que já foi

implementado e de um módulo Ponto que deverá ser implementado.

Deve existir uma árvore distinta para um dos elementos urbanos. As operações de busca espaciais (região retangular e circular) devem ser feitas sobre a árvore. É **expressamente proibido** fazê-las sobre uma lista. É **expressamente proibido** manter uma lista com todas as quadras, outra lista com todos os textos, outra com todos os hidrantes, etc. Permite-se apenas, que se utilize um vetor auxiliar contendo tais elementos para fins de ordenação.

Todos os TADs devem ser implementados como módulos, conforme descrito em sala de aula (.h, .c). É **expressamente proibida** a declaração de structs em arquivo .h.

O arquivo .h deve ser escrito antes do respectivo arquivo .c. O arquivo .h deve estar bem comentado (lembrar que este arquivo é um contrato).



FASE V

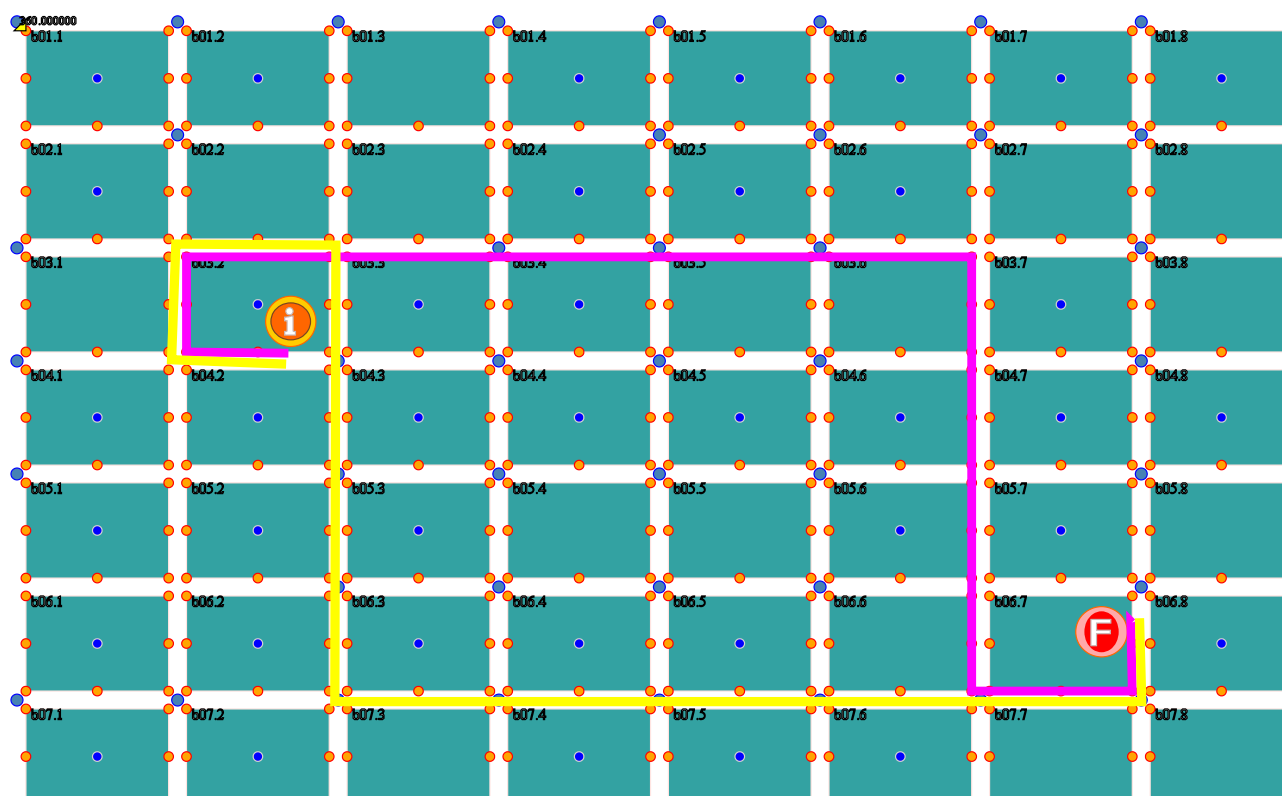
A nossa cidade está completa. Possui quadras, equipamentos urbanos, moradores, e estabelecimentos comerciais. Agora, queremos determinar trajetos na cidade. Uma possível consulta seria:

"Qual é o melhor (mais curto, mais rápido) entre o endereço relativo à posição do equipamento urbano X e o endereço 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.



A cidade também quer implantar ciclovias por toda a cidade. Ciclovias de mão dupla serão colocadas em todas as ruas, quer a rua seja de mão única ou de mão dupla.

Entrada de Dados

A entrada de dados será feita por meio dos arquivos-texto do trabalho anterior e de um novo 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. Referências geográficas são obtidas por meio dos comandos iniciados por @ e armazenadas nos registradores R0 .. R10. A referência geográfica armazenada num registrador pode ser utilizada em outras consultas. Para esta consultas, deve aparecer no SVG um linha vertical da coordenada geográfica determinada até o topo do mapa. No topo do mapa deve ser colocada a identificação do registrador (R0,R1,...)

A resposta esperada é a descrição textual do percurso e a representação pictórica do percurso do caminho mais curto e do caminho mais rápido. A representação pictórica, na consulta é informado o sufixo do arquivo de saída (**nome-arq-sufixo.svg**).

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 um subgrafo particular: a árvore geradora de custo mínimo

Existem ainda outras consultas cujas respostas não são percursos e, portanto, não necessitam dos parâmetros básicos mostrados acima.

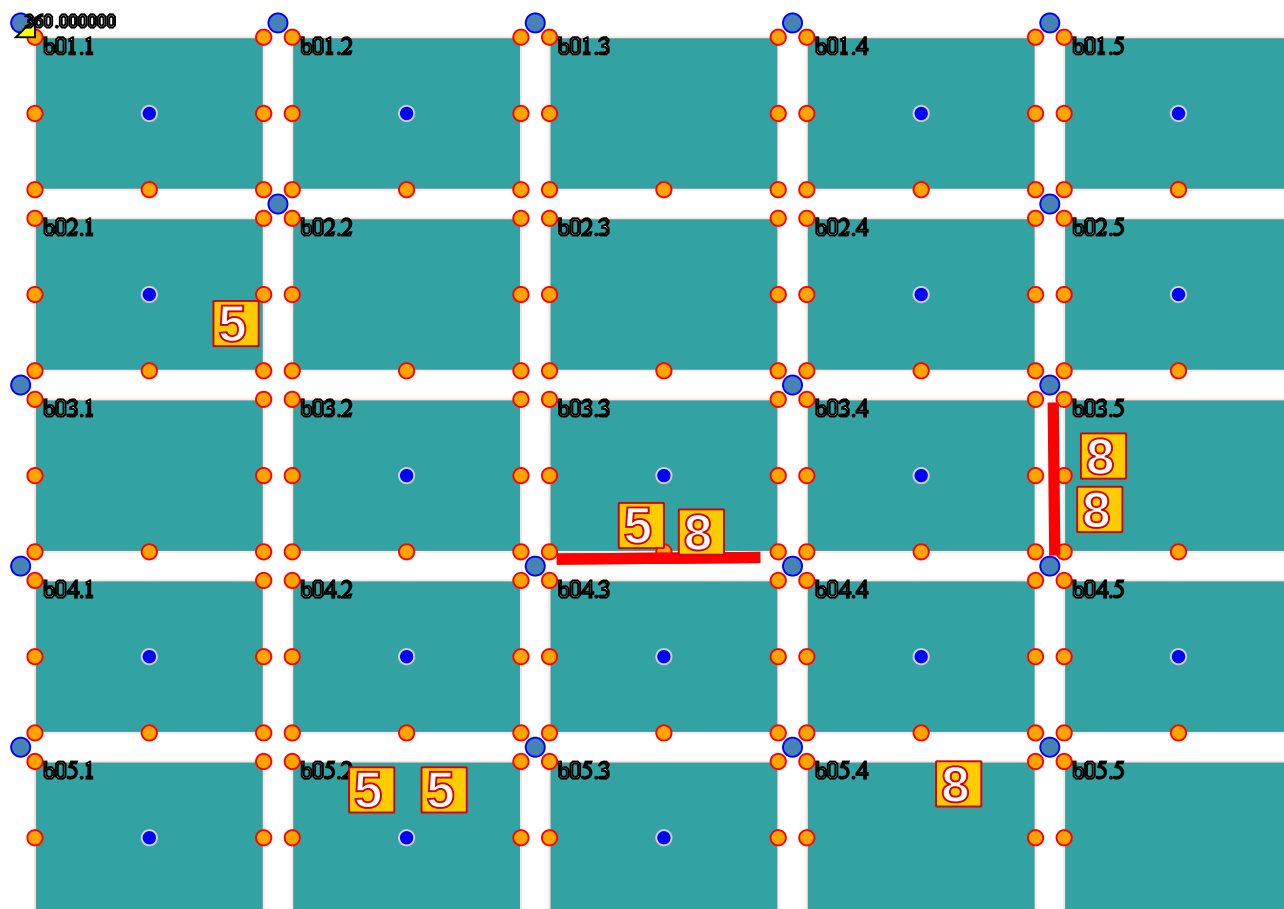
Os novos comandos e seus parâmetros específicos estão listados na tabela abaixo.

ATENÇÃO: *cat* deve remover os vértices do grafo (ruas e ciclovias) internos à região afetada.

ATENÇÃO: *soc* deve ser alterado deforma a determinar (e mostrar) o trajeto ao posto mais próximo.

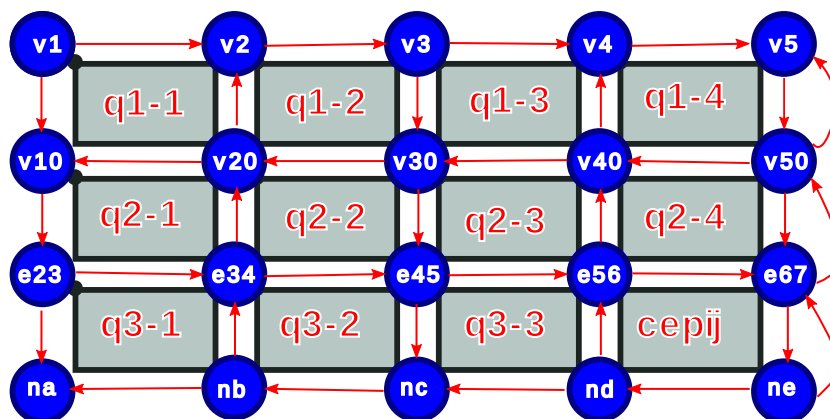
comando	parâmetros	
@m?	r cpf	Armazena no registrador r a posição geográfica da residência do morador de cpf . SVG: linha vertical com identificação do registrador.
@e?	r cep face num	Armazena no registrador r a posição geográfica do endereço cep/face/num SVG: linha vertical com identificação do registrador.
@g?	r id	Armazena no registrador r a posição geográfica do equipamento urbano cujo identificador é id SVG: linha vertical com identificação do registrador.
@xy	r x y	Armazena no registrador r a posição geográfica (x,y) SVG: linha vertical com identificação do registrador.

ccv	sufx	<p>Produza uma árvore geradora mínima que representa as ciclovias implantadas na cidade.</p> <p>SVG (nomearq-sufx.svg): desenhar o grafo, evidenciando as arestas que compõem a árvore geradora mínima.</p>
p?	sufx r1 r2 cmc cmr	<p>Qual o melhor trajeto de carro entre a origem que está no registrador r1 e o destino que está no registrador r2. O percurso na representação pictórica deve indicar o trajeto mais curto na cor cmc e o trajeto mais rápido com a cor cmr.</p> <p>TXT: descrição textual do percurso</p> <p>SGV (nomearq-sufx.svg): traçar (linhas) os percursos com as cores correspondentes. Cuidado para que um percurso não esconda o outro. Indicar o início e o fim do percurso. <i>Caso sufx for -, continuar usando o mesmo arquivo de um comando p? anterior.</i></p>
bf	max	<p>Interditar trechos de ruas (face de quadra) com mais que max casos de covid.</p> <p>TXT: listar cep e face referente ao trecho interditado e o número total de casos de covid naquela face.</p> <p>SGV: desenhar um segmento de reta vermelho e largo ao lado da face da quadra, mas fora dela.</p>
sp?	sufx r1 r2 cmc cmr	<p>Igual a p?, mas evita toda a região com casos de covid. Evitar completamente a região determinada pela envoltória convexa dos casos de covid.</p> <p>TXT: mesma saída de p?</p> <p>SVG: mesma saída de p?. Também desenhar a envoltória convexa com borda grossa, preenchida com amarelo e transparência de 60%</p>
pb?	sufx r1 r2 cmc	<p>Calcula o percurso por ciclovias. Semelhante a p?, mas o cálculo deve considerar a árvore geradora mínima. Como o percurso é por bicicleta, o cálculo deve apenas otimizar apenas a distância a ser percorrida</p>
Novos comandos do arquivo .qry		



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.



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):

v id x y	<i>cria o vértice id posicionado nas coordenadas [x,y]</i>
e i j ldir lesq cmp vm nome	<i>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 (-)</i>

Abaixo, um exemplo deste arquivo:

```
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.

A Implementação

Para a determinação de caminhos mínimos **deve** ser utilizado o algoritmo de **Dijkstra**. 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.

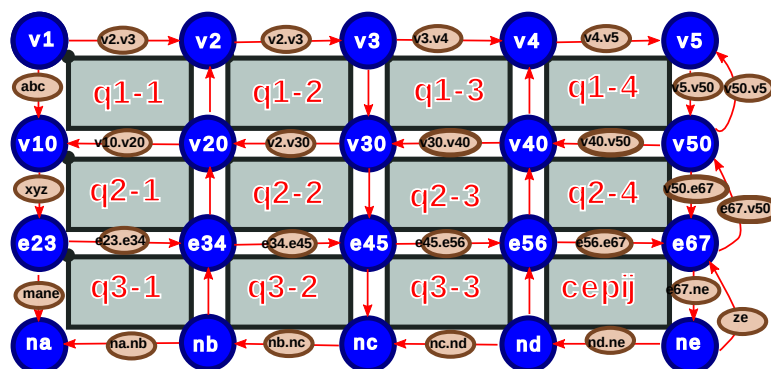
ATENÇÃO: Os dados de pessoas e moradores (arquivo .pm) e de estabelecimentos comerciais (arquivo .ec) devem ser armazenados em **hashfiles**. A implementação do hashfile deve seguir o especificado no arquivo `hashfile.h`. A equipe pode escolher entre **hashfile estático** ou **hashfile extensível**. Os hashfiles podem ser mantidos e usados em execuções subsequentes (veja parâmetros `-k` e `-u`).

IMPORTANTE: Consulta por região: descer apenas em sub-árvores “promissoras”.

Vale relembrar e enfatizar as diversas estruturas de dados (árvore, grafo, tabela de espalhamento, etc) 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:

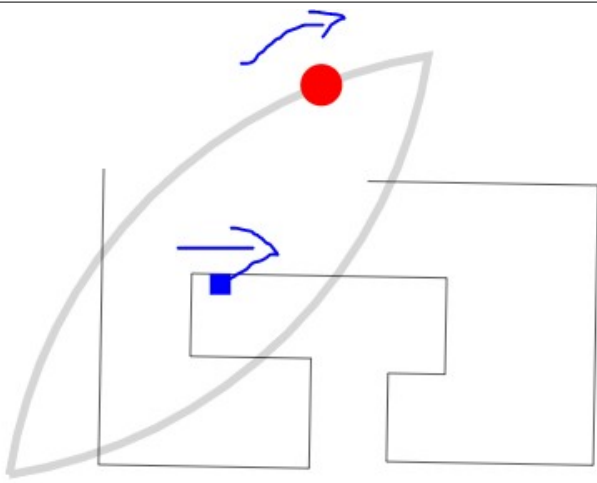
- Existem vértices do grafo posicionado no ponto médio das intersecções de ruas.
- Normalmente, existirão vértices no ponto médio de um segmento de rua. Isto é, o segmento de rua referente a uma face de uma quadra poderá ser, na prática, representado por duas aresta: (esquina1,meioquadra) e (meioquadra,esquina2). 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.



As ciclovias criadas pelo comando `ccv` são, como já dito, dadas pelo cálculo da árvore geradora mínima do grafo viário. Foram apresentados 2 algoritmos. A equipe pode escolher qual preferir.

Os percursos devem ser animados. Isto é, alguma figura (retângulo, círculo, etc)¹³ 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)

¹³ Para os valentes: desenhar um carro.



```

<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" />

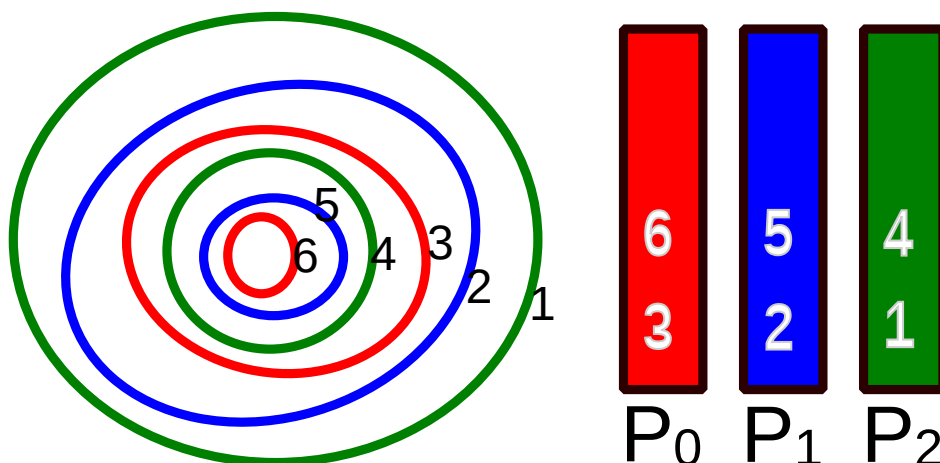
<circle cx="" cy="" r="5" fill="red">
  <animateMotion dur="6s" repeatCount="indefinite">
    <mpath xlink:href="#theMotionPath"/>
  </animateMotion>
</circle>

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

```

OPCIONAL (porém, muito fácil)

Observamos que a estratégia “cebola” de fazer a inserção na árvore não foi tão efetiva quanto esperado, pois as camadas pareciam muito próximas. A pilha pode ser substituída por duas ou mais pilhas, de forma que as camadas são armazenadas alternadamente (e ciclicamente) em cada uma das pilhas. A inserção se dá esvaziando completamente cada uma das pilhas em sequência.



Ordem de inserção: **6,3** , **5,2** , **4,1** .

AVALIAÇÃO

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

Como de costume, a avaliação consistirá da execução dos testes e da inspeção de código. Além disso, a equipe deverá produzir um vídeo de, aproximadamente, 10 minutos no qual apresenta o sistema. A equipe deve ter por objetivo convencer o avaliador que: (a) o programa funciona; (b) o programa foi bem implementado; (c) todos os membros da equipe efetivamente trabalharam no projeto. O vídeo deve ser colocado no Youtube e o seu link deve estar anotado no arquivo leia-me.

O PROGRAMA

O nome do programa deve ser **siguel** e aceitar quatro parâmetros:¹⁴

```
siguel [-e path] -f arq.geo [-q consulta.qry] -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**) deve 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).

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

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

14 Novos parâmetros são acrescentados no decorrer do ano

RESUMO DOS PARÂMETROS DO PROGRAMA SIGUEL

Parâmetro / argumento	Opcional	Descrição
-e <i>path</i>	S	Diretório-base de entrada (BED)
-f <i>arq.geo</i>	N	Arquivo com a descrição da cidade. Este arquivo deve estar sob o diretório BED .
-o <i>path</i>	N	Diretório-base de saída (BSD)
-q <i>arqcons.qry</i>	S	Arquivo com consultas. Este arquivo deve estar sob o diretório BED .
-ec <i>arq.ec</i>	S	Arquivo de estabelecimentos comerciais. Este arquivo deve estar sob o diretório BED .
-pm <i>arq.pm</i>	S	Arquivo de pessoas. Este arquivo deve estar sob o diretório BED .
-v <i>arq.via</i>	S	Arquivo de vias (para construção do grafo)
-k <i>nomebase.dat</i>	S	(keep hashfile) Mantém os hasfiles no disco (no estado que estejam ao final da execução). Para cada hashfile, colocar uma extensão no nome (a sua escolha). Por exemplo: nomebase-pm.dat e nomebase-ec.dat
-u nomebase	S	

RESUMO DOS ARQUIVOS PRODUZIDOS

-f	-q	comando com sufixo	arquivos
<i>arq.geo</i>			arq.svg
<i>arq.geo</i>	<i>arqcons.qry</i>		arq.svg arq-arqcons.svg arq-arqcons.txt
<i>arq.geo</i>	<i>arqcons.qry</i>	<i>sufx</i>	arq.svg arq-arqcons.svg arq-arqcons.txt arq-arqcons-sufx.[svg txt] ¹⁵

ATENÇÃO:

* os fontes devem ser compilados com a opção `-fstack-protector-all`.

* adotamos o padrão C99. Usar a opção `-std=c99`.

¹⁵ Podem ser produzidos os respectivos arquivos .svg e/ou .txt, dependendo da especificação do comando.