PROJETO 2019

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 atribuidas 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 7. O peso das atividades em equipe será 4. 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.
Caso a última atividade seja individual, o peso será 11; caso seja em equipe, o peso será 6.

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.

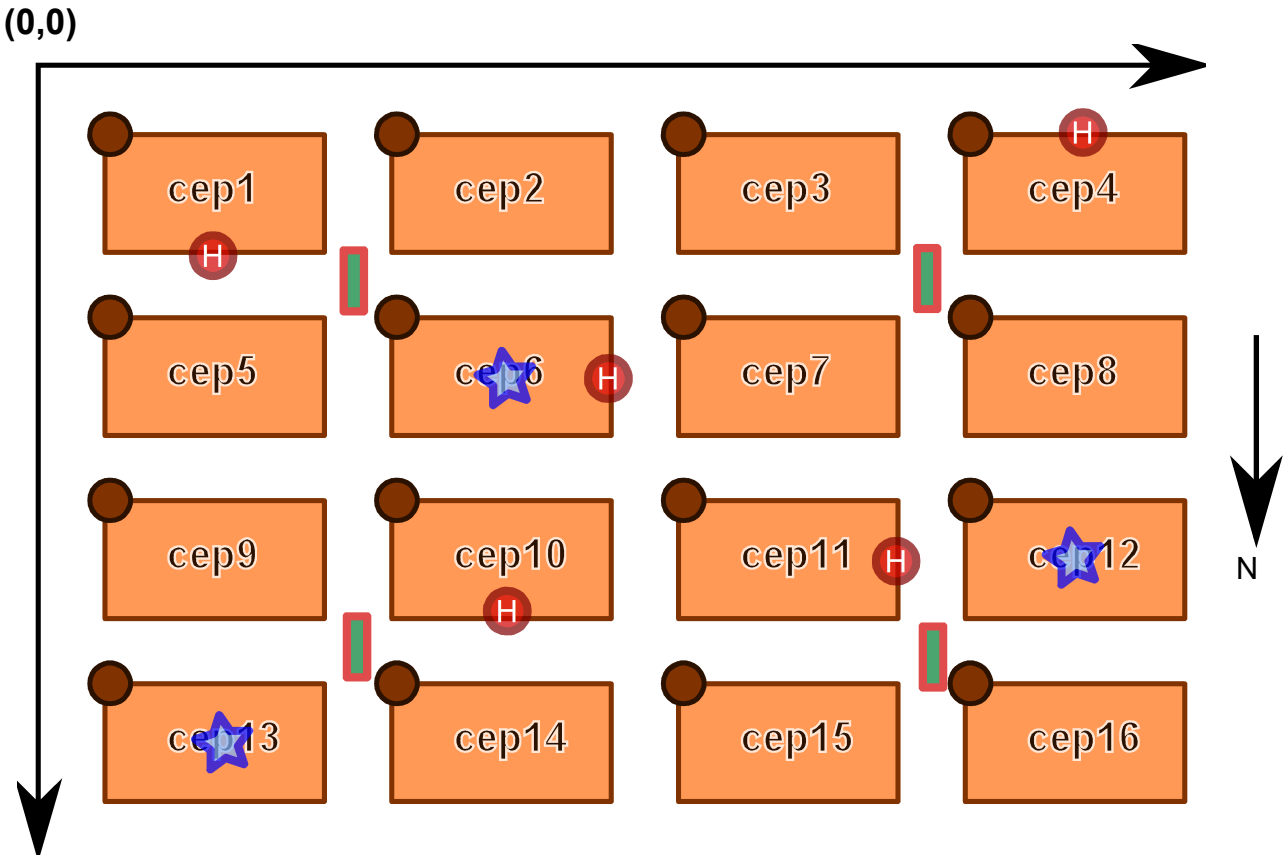
IMPORTANTE: não será tolerado nenhum tipo de fraude nas provas ou nos trabalhos. Qualquer fraude detectada durante ou após sua a realização será punida com nota zero a todos os envolvidos.

DESCRIÇÃO

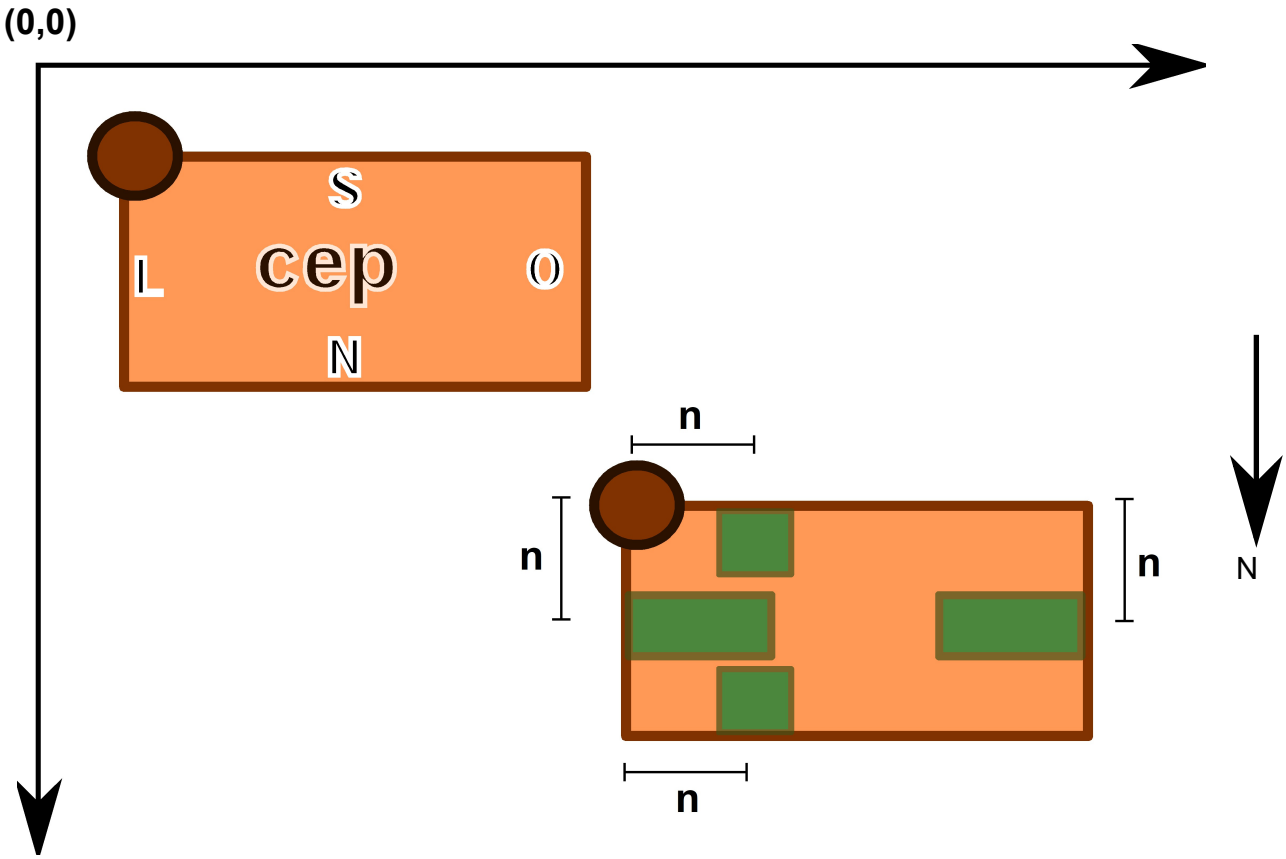
Um Sistema de Informações Geográficas (SIG), para a nossa finalidade, é um sistema que contém (não exclusivamente) dados geo-referenciados, isto é, dados com algum atributo de localização espacial (uma coordenada).

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.



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, jrsilva.zip (se aluno se chamar José Roberto da Silva), jrsilva-mrcarneiro.zip (para uma equipe com dois alunos. Evite usar maiúsculas, caracteres acentuados ou especiais.

[abreviatura-nome]	*Por exemplo, jrsilva.*
LEIA-ME.txt	*colocar matrícula e o nome do aluno. Atenção: O número da matricula 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* **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.*
*.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., jrsilva)*
	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 esquerdo[4] 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.

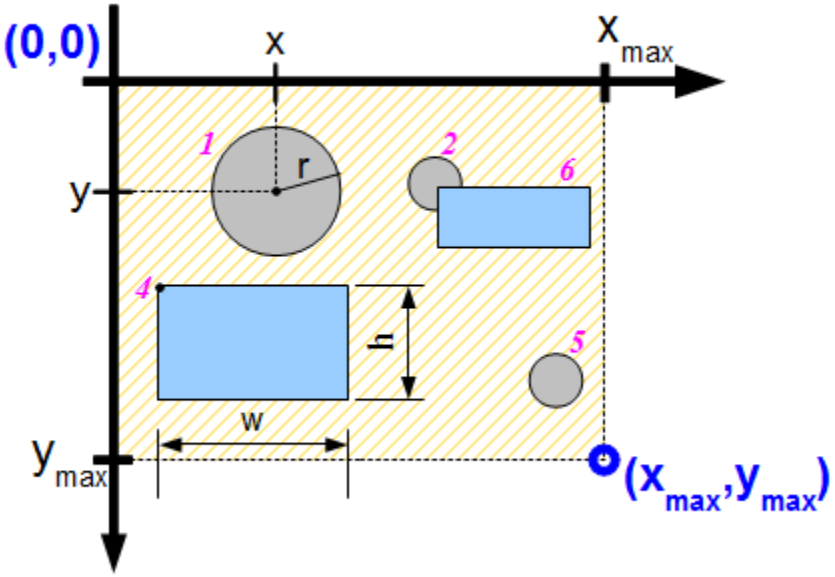


Ilustração 1

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 retâgulo), **`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`	*Altera o número máximo de círculos e retângulos (i.e., c + r) criados no arquivo. O valor default é 1000.*
c	`i r x y cor1 cor2`	*desenhar círculo. cor1 é a cor da borda e cor2 é a cor do preenchimento*
r	`i w h x y cor1 cor2`	*desenhar retângulo: w é a largura do retângulo e h, a altura. cor1 é a cor da borda e cor2, a do preenchimento*
t	`x y texto`	*todo o texto até o final*
comandos .geo		

comando	parâmetros	descrição
o?	`j k`	*As formas geométricas cujos indentificadores são* **`j`** *e* **`k`** *se sobrepõem?*[6] ***Saida*: *.txt:* *copiar o comando e, na linha seguinte escrever uma mensagem informando se sobrepõe ou não*** ***.svg*: *traçar um retângulo que envolva ambas figuras, de bordas tracejadas, se não se sobrepõem; linha cheia, se se sobrepõe***
i?	`j x y`	*O ponto (x,y) é interno à j-ésima forma geométrica?*[7] ***Saída*: *.txt*: *copiar o texto da consulta e, na linha seguinte, informar se é interno ou não.*** ***.svg*: *colocar um ponto (pequeno círculo) nas coordenadas (x,y) e pintá-lo de de verde se for interno e vermelho se for externo. Colocar uma linha ligando o ponto ao centro de massa da figura j.***
d?	`j k`	*Qual é a distância entre os centro de massa das formas geométricas* **`i`** *e* **`j`***?* ***Saída*: *.txt*: *copiar o texto da consulta e, na linha seguinte, informar a distância.*** ***.svg*: *traçar uma reta entre os centros de massa das duas figuras e escrever próximo da metade da linha a distância***

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
bb	sufixo cor	*Cria um novo arquivo svg contendo os círculos e retângulos referentes aos comandos c e r processados até o momento.* ***Para cada círculo****, deve ser criado um retângulo que o envolve ("bounding box").* ***Para cada retângulo*** *deve ser criada uma elipse envolvida por tal retangulo. As figuras produzidas deve ser preenchidas com a cor* **cor**. *O nome do arquivo gerado deve ser* **nomebase-*sufixo.svg***.
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. Na parte parte final do arquivo, estão colocadas duas consultas de sobreposição. A primeira pergunta se o círculo 2 e o retângulo 6 se sobrepõem (de fato, sim) e, a segunda, pergunta se os círculos 1 e 5 se sobrepõem (de fato, não). O último comando solicita que seja produzido um outro arquivo **.svg** (a01-lnhs2.svg) mostrando linhas originárias do centro de massa da figura 2 até cada uma das outras figuras, anotando na respectiva linha o seu comprimento.

```
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
o? 1 5
d? 2 4
i? 5 210.0 160.0
bb suf01 green
```

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 retâgulo 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.

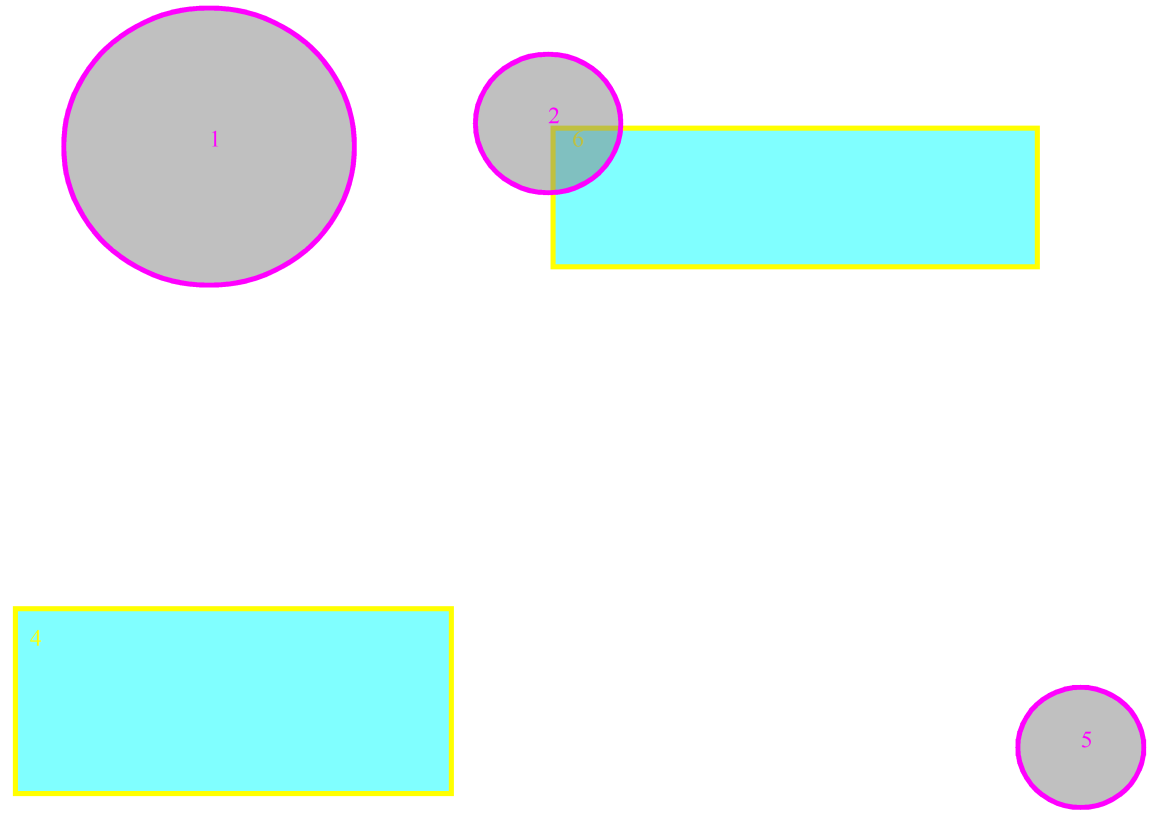
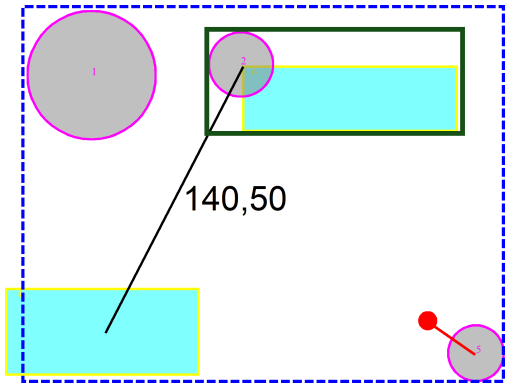
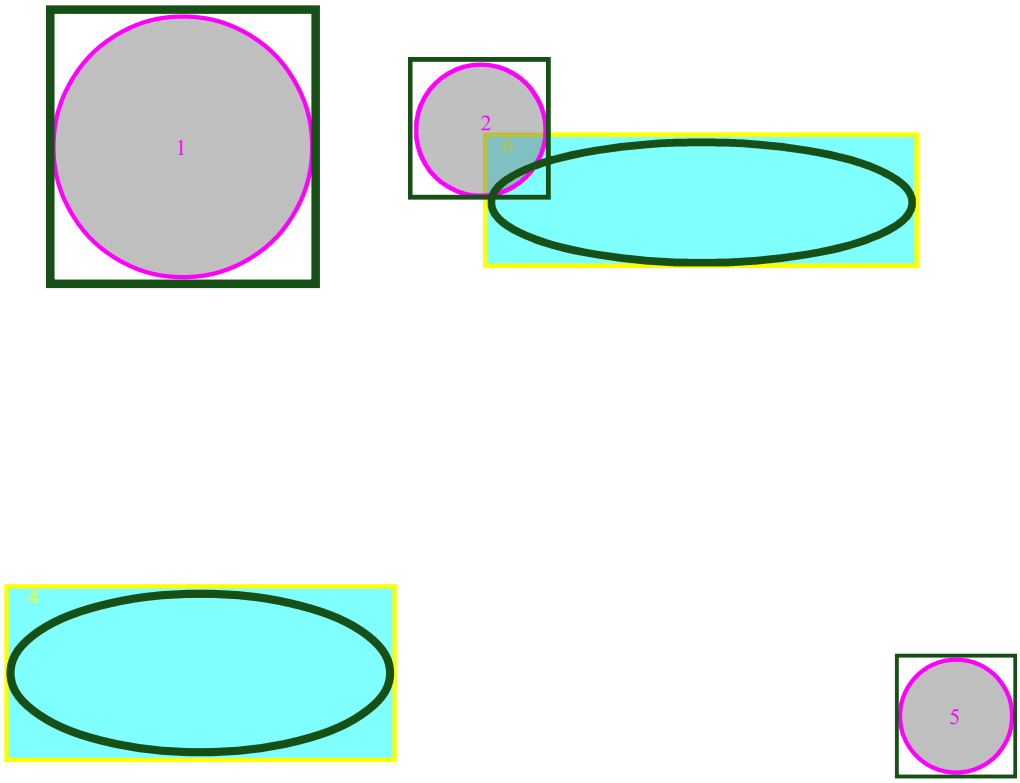


Illustration 2: Arquivo a01.svg

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 acrescidos aos resultados da consulta. Como no exemplo:



Caso o arquivo .qry contiver o comando bb, deverá ser produzido um terceiro arquivo .svg (a01-q-suf01.svg), contendo os círculo, retângulos, etc, acrescido do resultado do comando bb, como no exemplo:



O processamento de um arquivo `.geo` deve 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
SIM

o? 1 5
NAO

d? 2 4
140,50

i? 5 210.0 160.0
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.

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

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

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

comando	parâmetros	
dq	`(L1 | L2) id r`	*remove todas quadras que estiverem inteiramente dentro a uma distância de no máximo r do equipamento urbano identificado por id segundo a métrica* `L1` *ou* `L2`*. determinado pelos parâmetros do comando.* ***No arquivo .svg:*** *as quadras removidas não devem aparecer. O equipamento urbano me questão deve enfatizado com um anel grosso de duas cores.* ***No arquivo .txt****, deve apresentar os ceps das quadras removidas, o id e respectivas informações do equipamento urbano.*
del	`( cep | id )`	*Remove a quadra, hidrante, semáforo ou torre de identificação* `id` *(ou* `cep`*).* ***No arquivo .svg****: quadra ou equipamento urbano removido não deve aparecer.* ***No arquivo .txt****: reportar os dados relacionados da quadra ou equipamento urbano removido.*
cbq	`x y r cstrk`	*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`*.* ***No arquivo .svg:*** *quadras eleitas pintadas conforme descrito.* *Reporta no* ***arquivo .txt*** *o cep das quadras que tiveram a cor da borda alterada*
crd?	`( cep | id )`	*Imprime no* ***arquivo .txt*** *as coordenadas e a espécie do equipamento urbano de um determinado cep ou com uma determinada identificação.*
trns	`x y w h dx dy`	*Desloca todas as quadras e equipamentos urbanos que estiverem inteiramente dentro do retângulo (x,y,w,h) em dx unidades no eixo x e dy unidades no eixo y. Note que dx e dy podem ser medidas negativas.* *No* ***arquivo .svg****: quadras e equipamentos urbanos devem aparecer nas novas posições.* *No* ***arquivo .txt:*** *cep e id das quadras e equipamentos movidos, bem como as respectivas posições anteriores e atualizadas.*
Comandos do arquivo .qry		

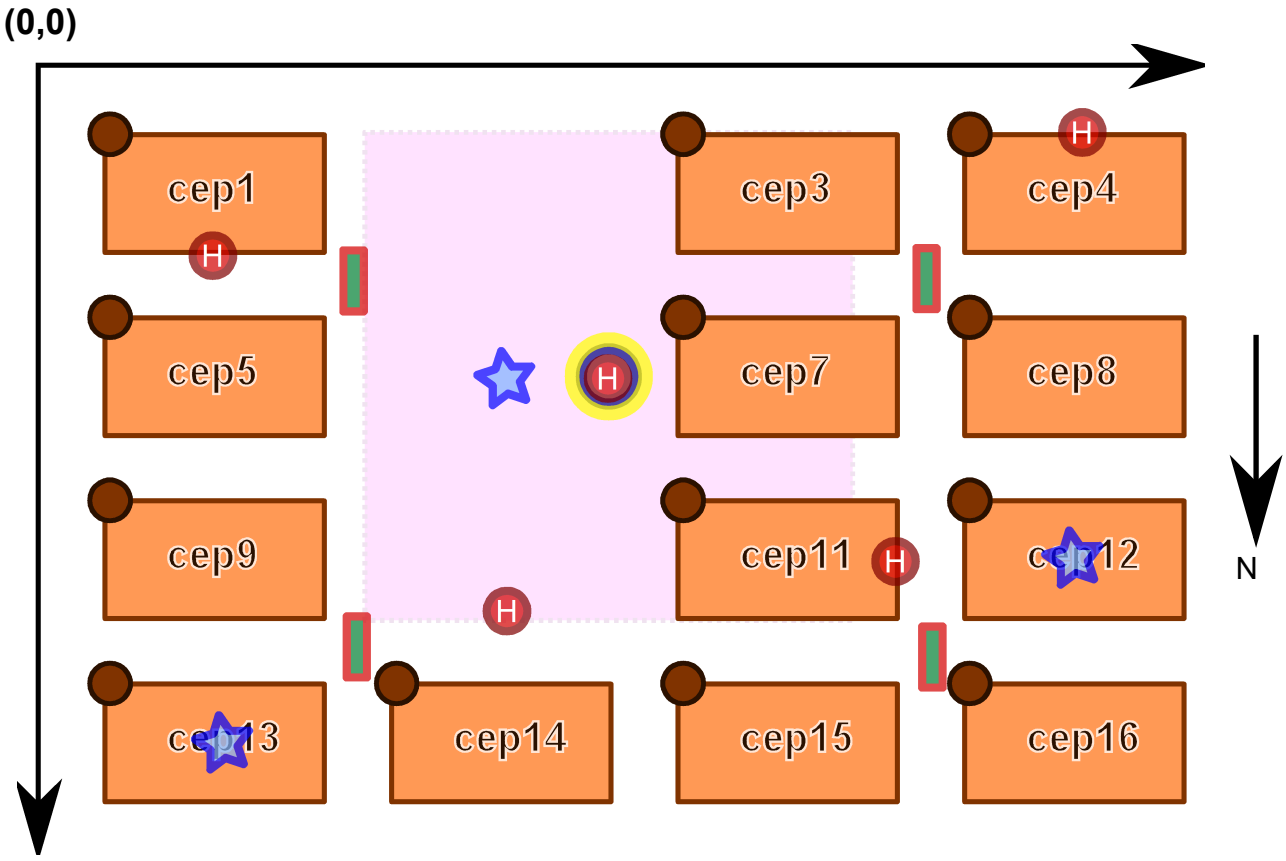
EXEMPLOS

```
cq blue black 2px
ch red yellow 1px
ct black red 3px
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 yellow green 1.5px
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 L1 h-12 120.0
crd? cep_001-10
crd? h-12
```

q1.qry



dq L1 h-12 120.0

FASE III

Nossa cidade está sob ameça de invasão de alienígenas. A estratégia de invasão dos alienígenas consiste em detonar em solo pequenas bombas de radiação luminosa. Todas a superfície exposta a tal radiação fica contaminada por alguns dias. Apesar da radiação ser facilmente bloqueada por qualquer obstáculo, ela é fatal para os seres humanos que se exponham diretamente a ela. Sua tarefa é, depois de cada ataque, delimitar as regiões contaminadas da cidade.

Nossa cidade também possui um sistema de combate a incêndios muito eficiente. Em especial, os hidrantes emitem um grande jato de água provocando uma chuva artificial em suas proximidades, dificultando a propagação do fogo na região afetada (até que os bombeiros possam chegar ao foco do incêndio).

Ao ser detectado um foco de incêndio, várias providências são tomadas automaticamente. Entre elas: alguns semáforos próximos tem sua programação alterada, de forma a desviar o fluxo de carros para fora da região afetada; os hidrantes na região são ativados. Além disso, para maximizar o fornecimento de água na região afetada, os hidrantes mais distantes são desativados. Também, os hidrantes intermediários tem seu suprimento de água reduzido em 50%.

A sua tarefa é determinar quais semáforos devem ser reprogramados e quais hidrantes devem ser ativados.

A ENTRADA

Abaixo, são mostrados os novos comandos de um arquivo .geo. Note que os parâmetros do comando nx foram modificados.

comando	parâmetros	
prd	`cep face num f p mrg`	*Insere prédio numa determinada face da quadra cep. O prédio tem f unidades de frente, p unidades de profundidade. A largura da calçada em frente ao prédio é de mrg unidades.* ***SVG:*** *retângulo representando as dimensões do prédio. Coloca o número no meio da frente do prédio.*
mur	`x1 y1 x2 y2`	*Muro linear entre as coordenadas (x1,y1) e (x2,y2).* ***SVG:*** *uma linha na posição do muro.*
nx	`i nq nh ns nr np nm`	*Informa o número máximo de formas(círculos e retângulos), quadras, hidrantes, semáforos e rádio-bases, prédios e muros, respectivamente, criados no arquivo.* *O valor default é 1000.*

Novos comandos do arquivo .geo

A figura abaixo mostra esquematicamente a semântica do comando prd. Esta figura mostra apenas uma quadra. O prédio foi colocado na face norte da quadra e mede f unidades de frente e p unidades de lado. Note que num é o número do prédio e, também, corresponde a distância da esquina até a metade da frente do prédio.

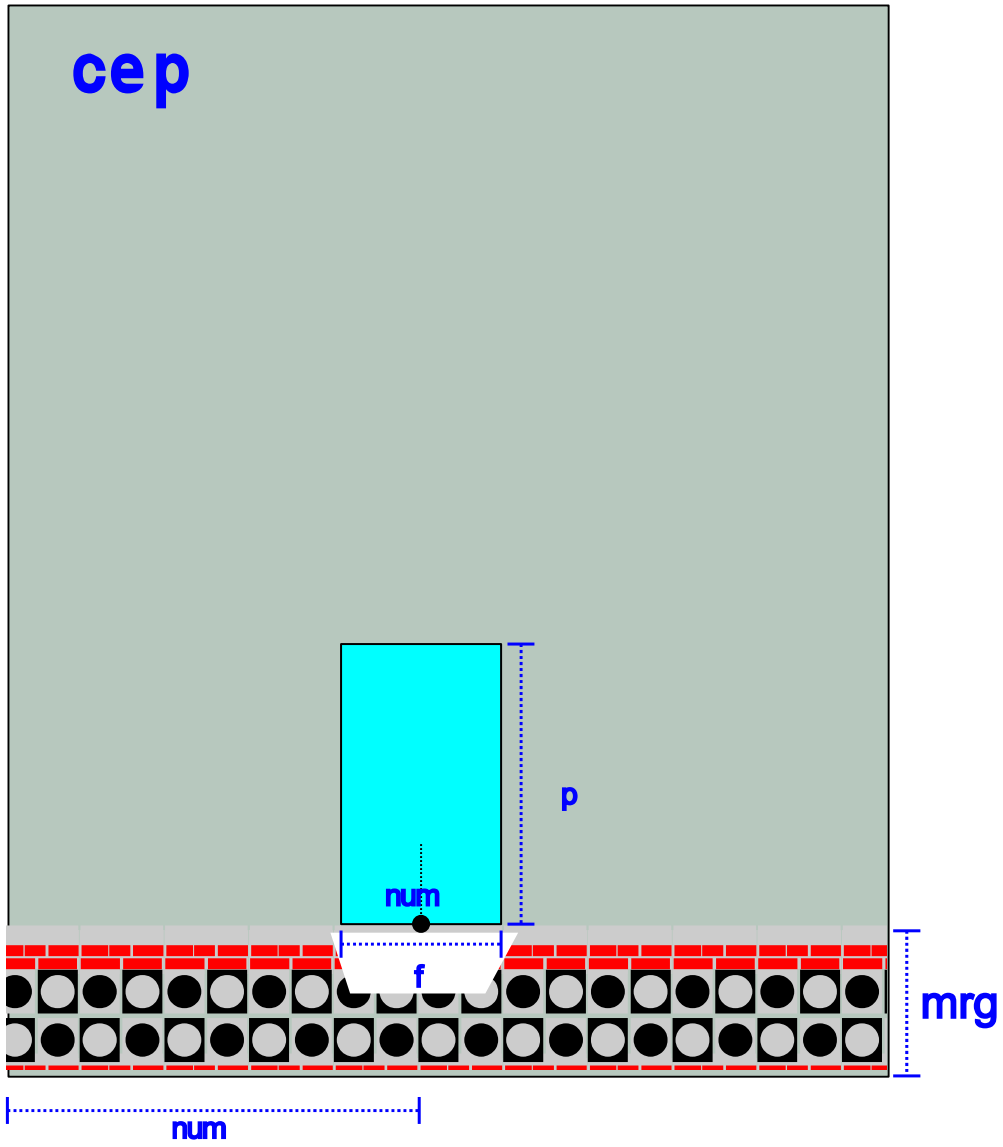


Illustration 3: Comando prd (uma quadra com calçada e um prédio)

Exemplo dos novos comandos num arquivo .geo:

```
...outros comandos ...
prd cep12 N 50 15.0 25.33 6.50
mur 10.5 5.5 15.0 20.0
mur 15.0 20.0 15.0 30.0
mur 150.0 150.0 200.5 210.5
...outros comandos ...
```

A seguir, são apresentadas as novas consultas que são adicionadas nesta fase.

comando	parâmetros	
brl	`x y`	*Detonada bomba de radiação luminosa na coordenada (x,y)* ***SVG:*** *Região afetada (point visibility region)*
fi	`x y ns r`	*Foco de incêndio identificado na coordenada (x,y). Encontrar os ns semáforos mais próximos do foco e os hidrantes que estejam a uma distância de até r do foco.* ***TXT:*** *identificação dos semáforos que devem ter programação alterada e dos hidrantes que devem ser ativados* ***SVG:*** *hidrantes e semafóros circulados, linha ligando foco ao hidrante/semáforo.*
fh	`[+|-]k cep face num`	*Determinar os k hidrantes mais próximos (se -k) ou mais distantes (se +k) do endereço.* ***TXT:*** *Listar identificadores* ***SVG:*** *Circular o hidrantes. Colocar segmentos do endereço aos hidrantes.*
fs	`k cep face num`	*Determinar os k semáforos mais próximos do endereço cep,face,num.* ***TXT:*** *Listar identificadores* ***SVG:*** *Circular semáforos. Colocar linhas do endereço para o semáforo.*
Novos comandos do arquivo .qry		

A figura abaixo apresenta um exemplo de uma região de visibilidade a partir de uma coordenada específica. Este ponto está representado por uma bomba. Note que a cidade foi cercada por 4 segmentos azuis.(formando um retângulo). Note que a cidade possui 7 prédios e 6 muros.

Exemplo de um arquivo .qry:

```
brl 150.0 215.80
fi 300.0 320.0 4 50.5
fh -10 cep5 N 40
fh +30 cep14 S 75
fs 8 cep8 L 18
```

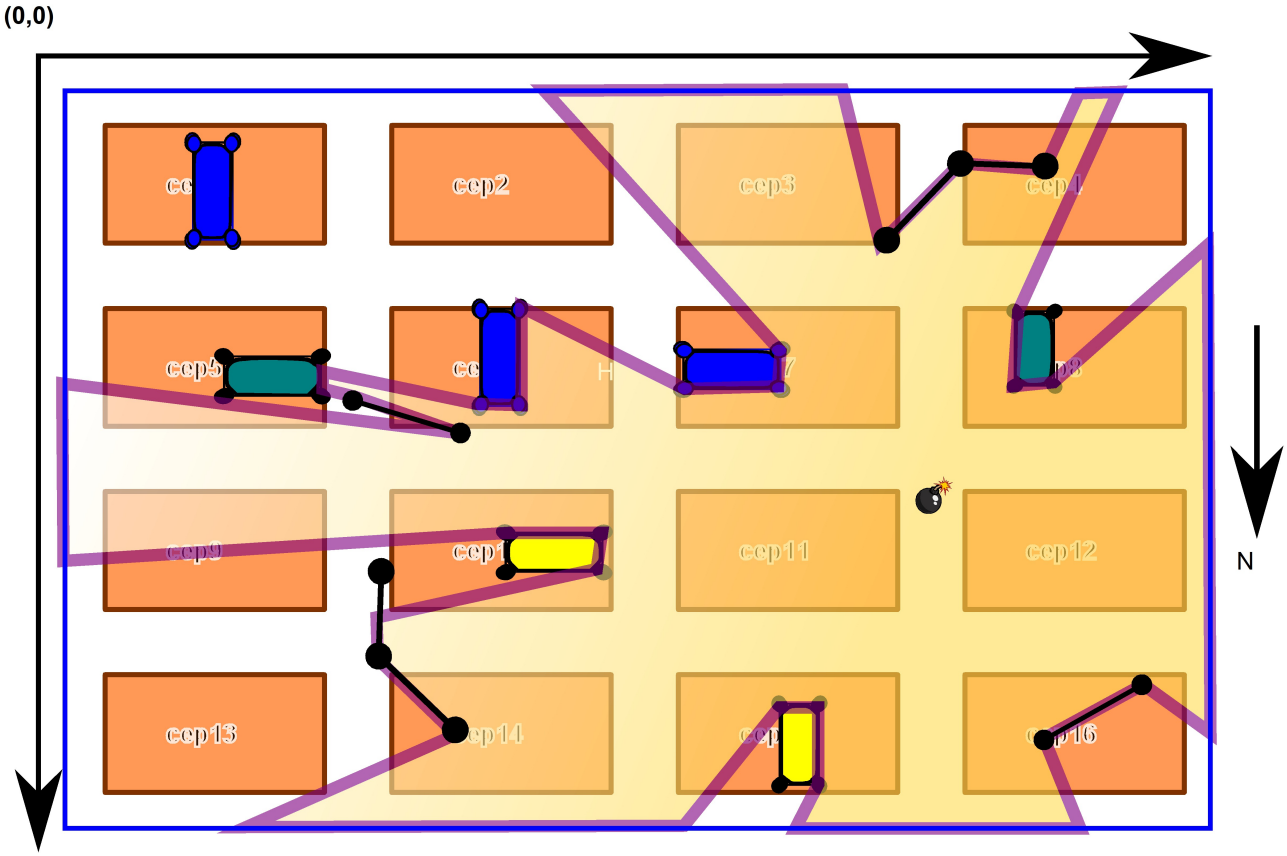


Illustration 4: Exemplo de uma região de visibilidade

A SAIDA

A arquivo `.geo` (com os novos comandos) deve continuar produzindo as mesmas saídas da fase anterior (com as modificações aprensentadas nesta fase).

A IMPLEMENTAÇÃO

Modificar o algoritmo Heapsort para implementar o algoritmo findKNearest/Farest (k mais próximos e k mais distantes).

O cálculo da região de visibilidade demanda a ordenação das barreiras (segmentos). Esta

ordenação deve usar o procedimento qsort provido na biblioteca padrão do C.

Os elementos da cidade (quadras, equipamentos urbanos, prédios e muros) devem ser armazenados na lista estática duplamente encadeada

Outros TADs **devem** ser projetados.

Todos os TADs **devem** ser implementados como módulos, conforme descrito em sala de aula (arquivos .h, .c)

Continua (até o final do projeto) a ser **expressamente proibida** a declaração de structs em arquivos .h.

FASE IV

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

Infelizmente, os alienígenas desenvolveram uma nova bomba – a bomba de radiação nuclear. Esta bomba é letal aos seres humanos.

Felizmente, os terráqueos desenvolveram uma tinta especial que bloqueia os efeitos da radiação. Como esta tinta é muito cara, ela foi usada para pintar os muros da cidade.

CONSULTAS CUJA REGIÃO DE INTERESSE SÃO POLIGONOS

Em muitas situações, quer-se determinar, por exemplo, os equipamentos urbanos que estão dentro de uma região de interesse. Esta região de interesse é definida por um polígono.

Um polígono pode ser representado por uma sequência de pontos que determinam suas arestas. Por exemplo, o polígono pode ser representado pela sequência

(x1,y1) (x2,y2) ... (x7,y7)

p1 p2 p7

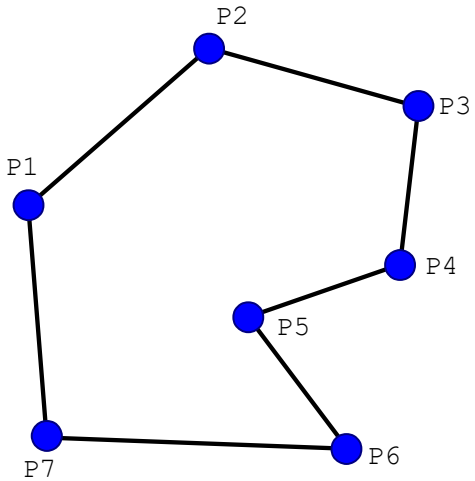
Esta sequência significa que o polígo é formado pelas arestas:

p1 – p2 p2 – p3 p3 – p4 p4 – p5 p5 – p6 p6 – p7 p7 – p1

Uma região de interesse é especificada em um arquivo com o formato abaixo. Cada linha corresponde a um ponto da sequência que determina o polígono. Note que a ordem dos pontos nesta sequência é importante

```
x1 y1
x2 y2
x3 y3
...
xn yn
```

poligono1.pol



ARQUIVOS DE ENTRADA

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

Estabelecimentos Comerciais

comando	parâmetros	
t	`codt descricao`	*Define tipo de estabelecimento comercial*
e	`cnpj cpf codt cep face num nome`	*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.
Comandos do arquivo .ec (-ec)		

Pessoas e moradores

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

comando	parâmetros	
brn	`x y arq-polig`	*Detonada bomba de radiação nuclear na coordenada (x,y). Salvar o poligono de visibilidade no arquivo* `arq-polig` *na diretório de saída. Atenção:* `arq-polig` *pode conter path relativo.* ***TXT:** medida da área afetada, cpf e nome dos moradores da região afetada.* ***SVG:** Região afetada (point visibility region)*
m?	`cep`	*Moradores da quadra cujo cep é* `cep`*. Mostra mensagem de erro se quadra não existir.* ***TXT:** listar todos os dados do morador (nome, endereço,...)*

mplg?	`arq-polig`	*Moradores dos prédios inteiramente contidos na região delimitada pelo polígono e as quadras que estão ao menos parcialmente dentro da delimitada pelo polígono.* ***TXT:*** *para cada predio da região, produzir uma saida similar a da consulta m?.* ***SVG:*** *deixar a borda das quadras afetadas mais espessa. Pintar os prédio afetados e que possuam moradores. Escrever sobre a quadra o número total de moradores da quadra.*
dm?	`cpf`	*Imprime todos os dados do morador identificado pelo cpf.* ***TXT:*** *dados pessoais, seu endereço*
de?	`cnpj`	*Imprime todos os dados do estabelecimento comercial identificado por cnpj.* ***TXT:*** *dados do estabelecimento (nome, descrição do tipo, etc) e dados de seu proprietário.*
mud	`cpf cep face num compl`	*A pessoa identificada por cpf muda-se para o endereço determinado pelos parâmetros.* ***TXT:*** *Mostrar os dados da pessoa (nome, etc), o endereço antigo e o novo endereço.* *arquivo.*
eplg?	`arq-polig [tp | * ]`	*Estabelecimentos comerciais do tipo* tp *(ou de qualquer tipo, caso *) que estão inteiramente dentro da região delimitada pelo polígono.* ***TXT****: dados sobre os estabelecimentos comerciais, incluindo o nome de seu proprietário.* ***SVG****: mudar a cor das quadras que tenham pelo menos um estabelecimento comercial selecionado e do tipo especificado. Destacar o estabelecimento comercial selecionado.*
catac	`arq-polig`	*Considere a região delimitada pelo polígono. Remover as quadras que estejam inteiramente contidas no poligono. Remover prédios (inteiramente contidos no poligono) e respectivos moradores, hidrantes, semáforos, rádios-bases.* ***TXT:*** *imprimir identificadores de tudo o que foi removido.* ***SVG:*** *elementos removidos não devem aparecer. Traçar um X (diagonais) no local dos prédios removidos. Colocar próximo ao X o número de pessoas que moravam naquele prédio.*

dmprbt	`t arq`	*Imprime o estado atual de uma árvore no arquivo* arq.*svg*. t *informa qual das árvores:* **q***: quadras;* **h***: hidrantes;* **s***: semáforos;* **t***: torres;* **p***: prédios;* **m***: muros* *Cada nó da árvore deve estar pintado de vermelho ou preto, deve mostrar as coordenadas relativas do elemento armazenado e alguma irformação relevante sobre tal elemento (p.ex, o cep da quadra, o identificador do hidrante, etc)*
Novos comandos do arquivo .qry		

IMPLEMENTAÇÃO

Deve ser implementado um módulo para o tipo abstrato Tabela de Espalhamento. Deve ser implementado um módulo para para árvore Rubro-Negra.

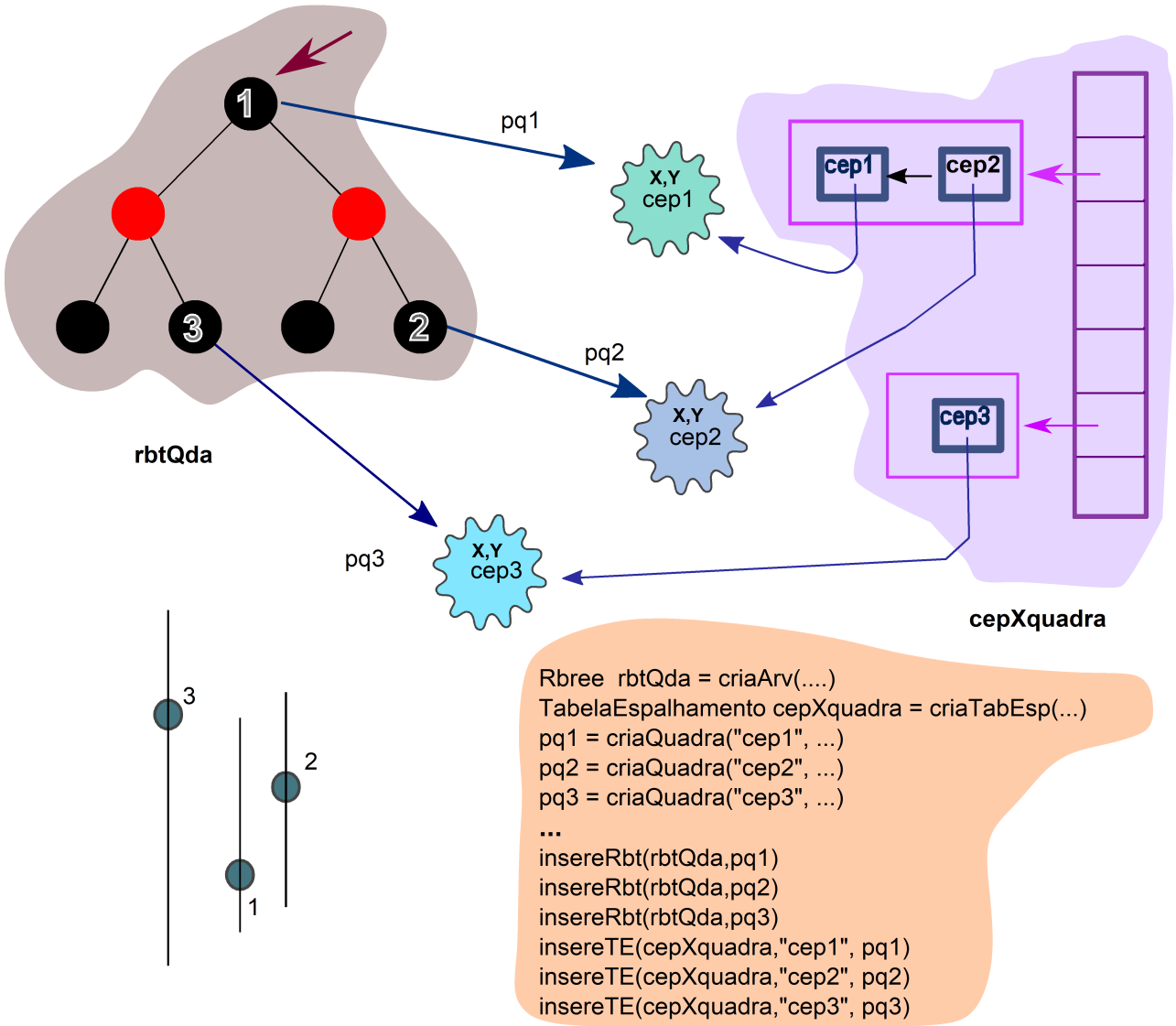
A árvore Rubro-Negra deve organizar os dados espacialmente, i.e., deve levar em consideração as coordenadas do objeto. Ela será usada para consultas espaciais, tais como: “quais elementos estão dentro de uma dada região?”. Assim, as quadras, os equipamentos urbanos, os prédios e os muros devem a ser armazenados em árvores Rubro-Negras .

Tabelas de espalhamento devem ser usadas como dicionários. Um dicionário, por exemplo, pode (deve) ser usado para manter a tabela de tipos de estabelecimentos comerciais (tipo X descrição). Vários dicionários deverão ser usados para facilitar as consultas.

Dicionários que devem ser mantidos e usados em consultas (pelo menos):

- 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 figura abaixo esquematiza a relação entre a árvore e a tabela de espalhamento.



ATENÇÃO: Todos os comandos das fases anteriores devem ser adaptados para usar as novas estruturas de dados. As listas estáticas duplamente encadeadas devem ser completamente substituídas por árvores.

ATENÇÃO: Cada equipamento urbano deve ser armazenada em sua respectiva e exclusiva árvore.

ATENÇÃO: Espera-se que partes comuns, principalmente, para a solução das consultas estejam fatorados (i.e., divididos em procedimentos curtos que executem uma única tarefa) e que sejam reutilizados largamente.

Execução Interativa

A partir desta fase, o programa siguel passará a poder ser executado também interativamente. Se o parâmetro `-i` estiver presente, o programa executa normalmente, mas não termina. Mantém os dados armazenados e abre um prompt para inserção de comandos do usuário.

Os comandos são:

- **`q arq.qry`**: lê e executa o arquivo de consultas `arq.qry`
- **`dmprbt t arq`**: igual comando de mesmo nome
- **`sai`**: sai do modo interativo e encerra a execução do programa
- **`nav t`**: faz navegação interativa na árvore t. Imprime dados relativos ao nó corrente da árvore (inicia com a raiz da árvore). Os dados mostrados devem ser a cor do nó e todos os dados do elemento armazenado. Os comandos de navegação são os seguintes:
  - **`e`**: desce para sub-árvore esquerda. Imprime dados do nó.
  - **`d`**: desce para sub-árvore direita.
  - **`p`**: volta para o pai.
  - **`x`**: sai da navegação

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 podem ser textuais ou podem ser 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.

Entrada de Dados

A entrada de dados será feita por meio dos arquivos-texto do trabalho anterior, 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 ou 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.

A resposta esperada é a descrição textual do percurso e a representação pictórica do percurso do caminho mais curto e do caminho mais rápido. No caso da representação pictórica, na consulta é informado o sufixo do arquivo de saída (**nome-arq**-**sufixo**.svg).

A sintaxe destas consultas são da forma:

`comando sufixo` ***outros-parâmetros***

Atenção: a região afetada por uma brn é intransitável, assim, o nenhum trajeto deve passar por tal região.

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. Os parâmetros básicos de percurso estão indicados por (¶).

comando	parâmetros	
@m?	`r cpf`	*Armazena no registrador r a posição geográfica da residência do morador de cpf* `cpf`

@e?	`r cep face num`	*Armazena no registrador r a posição geográfica do endereço cep/face/num*
@g?	`r id`	*Armazena no registrador r a posição geográfica do equipamento urbano cujo identificador é* id
@xy	`r x y`	*Armazena no registrador* `r` *a posição geográfica (x,y)*
p?	¶ `r1 r2 cormc cormr`	*Qual o melhor trajeto entre a origem está que no registrador* `r1` *e o destino que está no registrador* `r2`. `O` *percurso na representação pictórica deve indicar o trajeto mais curto na cor* `cormc` *e o trajeto mais rápido com a cor cormr* *ATENÇÃO: o cálculo do percurso deve levar em consideração as áreas interditadas por bombas*
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.

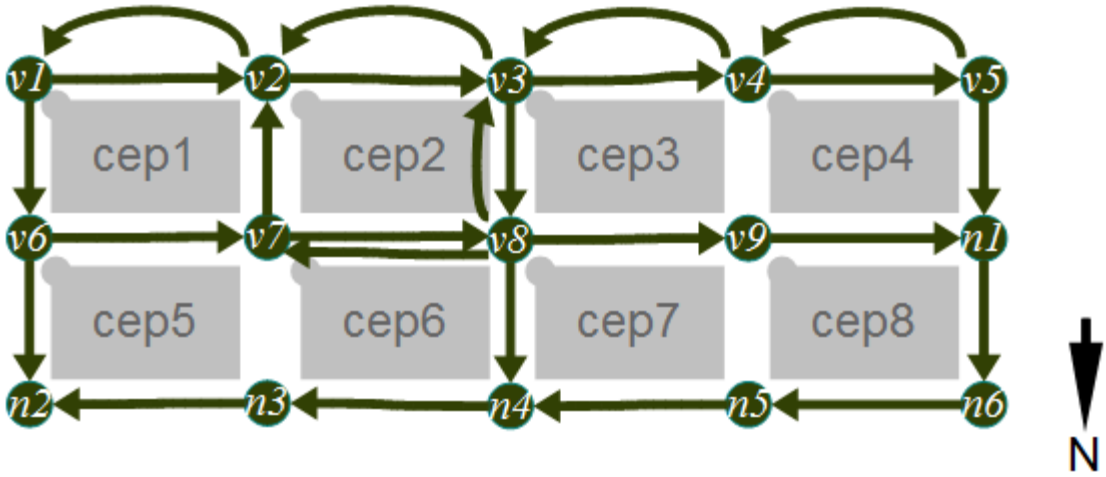


Figura 5:

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`	*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:

```
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.

Vale relembrar e enfatizar as as diversas estruturas de dados (árvore, grafo, tabela de espalhamento, etc) devem ser implementadas em módulos separados e bem documentados.

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.

Navegação Interativa

A navegação interativa deve ser incrementada com os seguintes comandos:

- **(qr | qc) arq.qry**: arquivo arq.qry é executudo. Espera-se que este arquivo tenha uma consulta p?. Quando tal consulta for executada, é feita a simulação da navegação pelo caminho mais rápido (qr) ou pelo caminho mais curto (qc). Em cada passo, deve ser indicado por texto na saida padrão (para os valentes, pode ser voz) a direção a seguir. O usuário não é obrigado a seguir a indicação. Porém, caso o usuário se afaste mais de 200 unidades de medida do último ponto correto, uma nova rota deve ser calculada. Os comandos de navegação são:
  - **n | s | l | o** : siga um passo na direção norte/sul/leste/oeste.
  - `rr` : recalcule o trajeto mais rápido a partir da posição atual
  - `rc` : recalcule o trajeto mais curto a partir da posição atual
  - `x` : encerre a navegação

O Programa

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

siguel [-i] [-e *path*] -f *arq*.geo [-q consulta.qry] -o *dir*

O primeiro parâmetro (`-i`) indica o modo interativo. O segundo 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 terceiro parâmetro (**-f**) especifica o nome do arquivo de entrada que deve ser encontrado sob o diretório informado pelo primeiro parâmetro . O quarto 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/al1/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`

A Avaliação

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

A avaliação se baseará em dois critérios: **(a)** inspeção do código-fonte (especial atenção na modularização); **(b)** compilação e testes do executável; (c) video (10 minutos) colocado no youtube

O Que Entregar

Submeter a sala Moodle o arquivo .zip com os fontes , conforme descrito anteriormente. Colocar um arquivo com o link do vídeo.

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**.
`-ec` ***`arq`*`.ec`**	S	Arquivo de estabelecimentos comerciais. Este arquivo deve estar sob o diretório **BED**.
`-pm` ***`arq`*`.pm`**	S	Arquivo de pessoas. Este arquivo deve estar sob o diretório **BED**.
`-i`	S	Processa os arquivos de entrada normalmente, mas não encerra o programa. Dados são mantidos em memória. Inicia modo interativo.
`-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][8]

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

8 Podem ser produzidos o respectivo arquivos .svg e/ou .txt, dependendo da especificação do comando.