PROJETO 2018

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/**

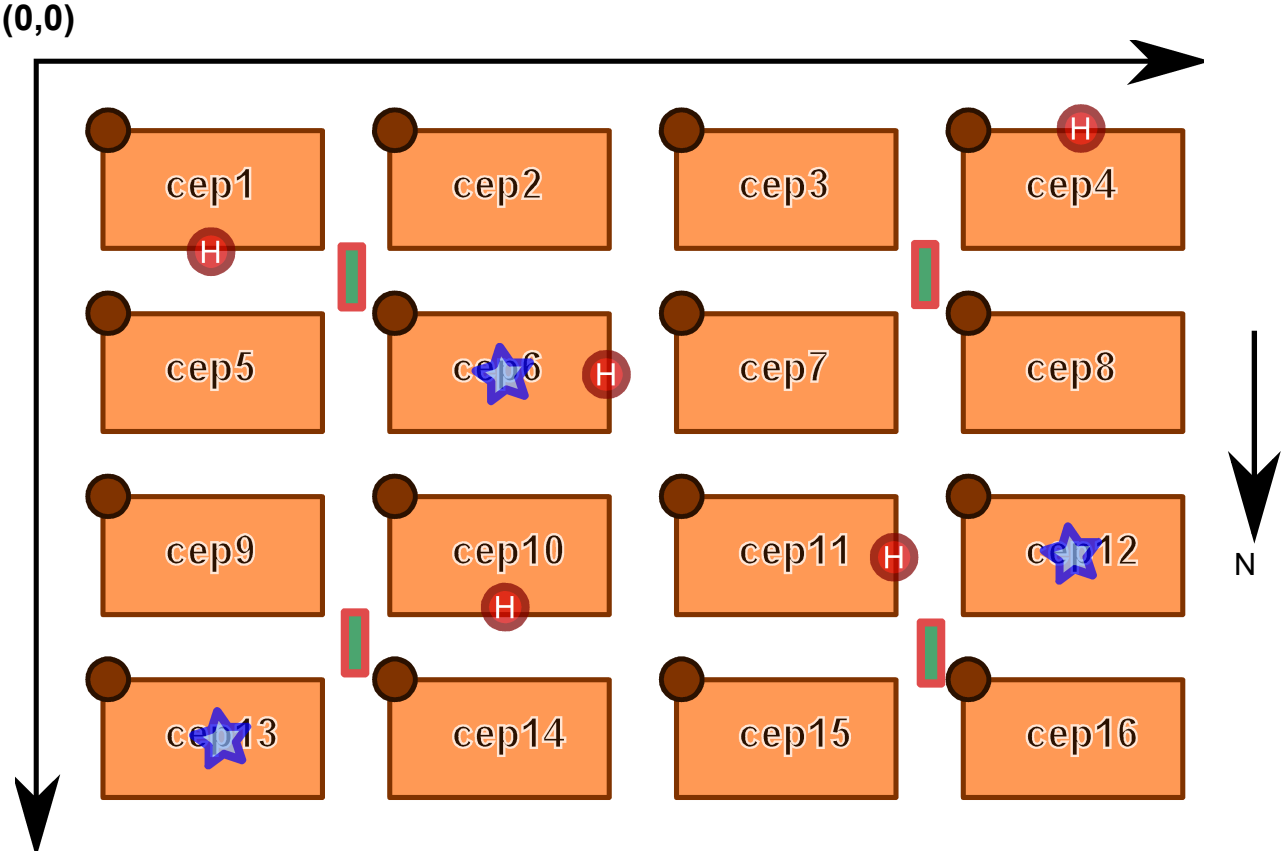
O sistema será implementado incrementalmente e em fases. É importante enfatizar que em cada fase o sistema evolui, isto é, novas funcionalidades são acrescentadas, requisitos de implementação podem ser mudados, porém, as funcionalidades existentes **devem continuar funcionais**.

DESCRIÇÃO

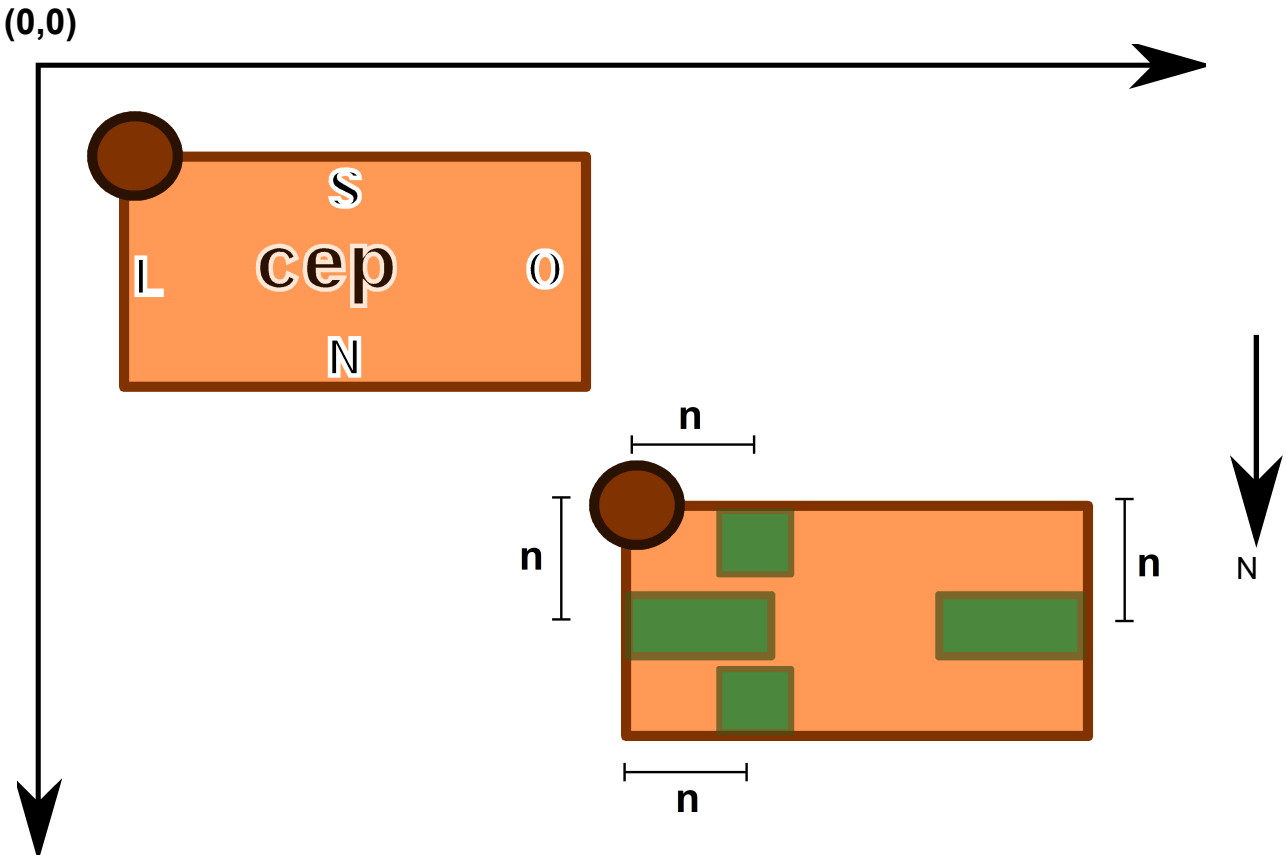
Um Sistema de Informações Geográficas (SIG), para a nossa finalidade, é um sistema que contém (não exclusivamente) dados geo-referenciados, isto é, dados com algum atributo de localização espacial (uma coordenada).

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.

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.

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, 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 as opções* **-pedantic -ansi**.
*.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	*arquivos de teste e, talvez, 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.

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.

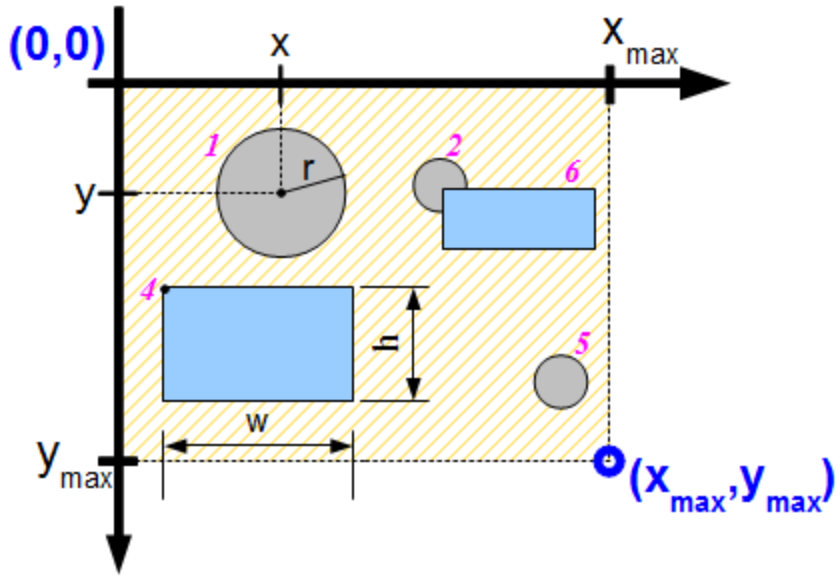


Ilustração 1

A tabela abaixo mostra o formato do arquivo de entrada, composto, basicamente, por conjunto de comandos (uma operação por linha), a saber: **`c`** (desenhe um círculo), **`r`** (desenhe um retâgulo), **`o`** (consulta sobreposição), **`i`** (ponto é interno a figura?), **`d`** (calcule a distância entre duas figuras) e **`a`** (desenhe apenas as âncoras das figuras). A última linha possui a marca de final de arquivo (**`#`**).

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.

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 cor1 cor2 r x y`	*desenhar círculo. cor1 é a cor da borda e cor2 é a cor do preenchimento*
r	`i cor1 cor2 w h x y`	*desenhar retângulo: w é a largura do retângulo e h, a altura. cor1 é a cor da borda e cor2, a do preenchimento*
o	`j k`	*As formas geométricas cujos indentificadores são j e k se sobrepõem?*[6]
i	`j x y`	*O ponto (x,y) é interno à j-ésima forma geométrica?*[7]
d	`j k`	*Qual é a distância entre os centro de massa das formas geométricas i e j?*
a	`i sufixo`	*Cria um arquivo svg contendo os círculos e retângulos referentes aos comandos c e r processados até o momento. Além dos círculos e retângulos, devem ser traçadas linhas a partir do centro de massa da figura de identificador i até o centro de massa de todas as outras figuras.* *O nome do arquivo gerado deve ser* nomebase-*sufixo.svg. As linhas devem ser desenhadas usando a cor da borda da figura* `i`*. Próximo a cada linha deve ser escrito o seu comprimento.*
#		*Marca o final do arquivo*

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.

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

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.

```
c 1 grey magenta 50.00 50.0 30.00
r 6 cyan yellow 121.0 46.0 100.0 30.0
c 2 grey magenta 120.0 45.0 15.0
r 4 cyan yellow 10.0 150.0 90.0 40.0
c 5 grey magenta 230.0 180.0 13.0
o 2 6
o 1 5
a 2 lnhs2
#
```

***a01*.geo**

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.

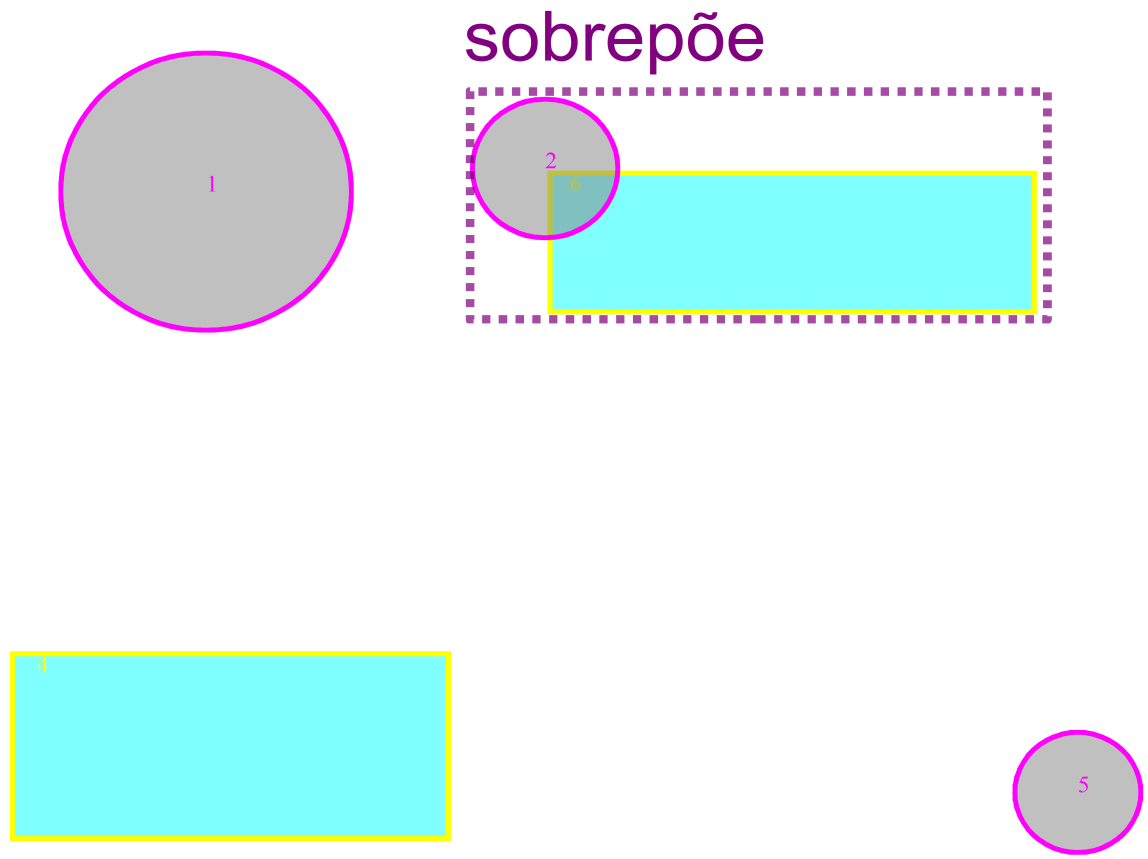


Illustration 2: Arquivo a01.svg

A operação **a** deve produzir um outro arquivo .svg, com nome *nomebase*`-sufixo.svg`, (no exemplo, `a01-lnhs2.svg`) , como exemplificado na figura abaixo.

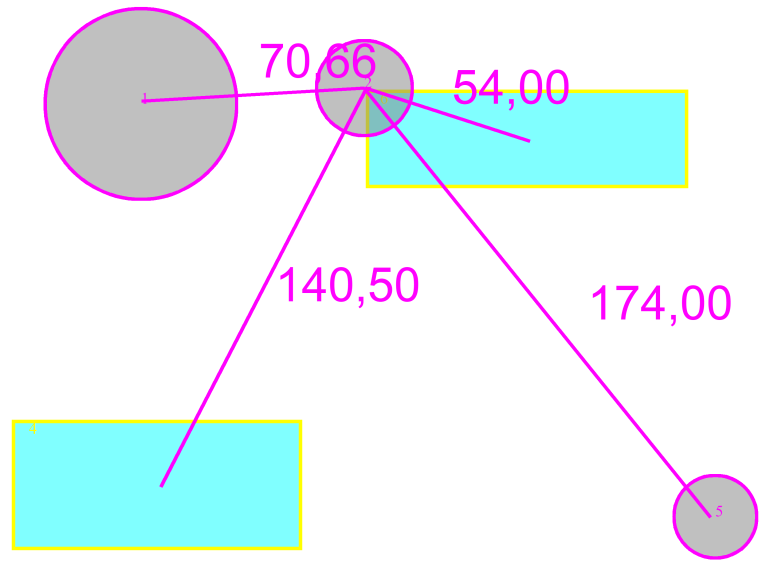


Illustration 3: Arquivo a01-lnhs2.svg

O processamento de um arquivo `.geo` deve também produzir um arquivo-texto contendo o resultado das consultas `i, d, o`. Este arquivo-texto deve ser nomeado `nome-base.txt`.

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
```

Arquivo a01.txt

O Programa

O nome do programa deve ser `siguel` e aceitar três parâmetros:

siguel [-e *path*] -f *arq*.geo -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 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`

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.

Recentemente, em Bitlândia (o país onde está Bitnópolis) foi editada uma lei que determina a distância mínima entre as rádio-bases de uma cidade. A distância mínima é a distância entre as rádio-bases mais próximas no momento da edição da lei. Assim, uma de suas tarefas é encontrar quais são as duas torres mais próximas na cidade qual a distância entre elas.

Novos comandos ao arquivo **`.geo`**:

comando	parâmetros	
q	`cep x y larg alt`	*Insere uma quadra (retângulo e cep)*
h	`id x y`	*Insere um hidrante*
s	`id x y`	*Insere um semáforo*
t	`id x y`	*Insere uma rádio-base (torre de celular)*
cq	`cstrk cfill`	*Cores do preenchimento e da borda das quadras (a partir deste comando)*
ch	`cstrk cfill`	*Cores do preenchimento e da borda dos hidrantes (a partir deste comando)*
ct	`cstrk cfill`	*Cores do preenchimento e da borda das torres de celular (a partir deste comando)*
cs	`cstrk cfill`	*Cores do preenchimento e da borda dos semáforos (a partir deste comando)*

Novos comandos do arquivo .geo

A partir desta fase, um novo arquivo será utilizado. Arquivos `.qry`:[8] podem conter alguns comandos de atualização e consulta.

comando	parâmetros	
q?	`x y larg alt`	*Reporta as quadras e equipamentos urbanos que estejam inteiramente dentro da retângulo determinado pelos parâmetros do comando.* *Saída: arquivo.txt: todos os dados sobre as quadras e equipamentos urbanos selecionados.* *arquivo.svg: traçar o retângulo da região de busca com linhas pontilhadas.*
Q?	`raio x y`	*Similar a q?. A região de busca é dada pelo círculo com centro* `x,y` *e raio* `raio`.

8 O arquivo `.qry` será informado pelo parâmetro `-q`.

comando	parâmetros	
dq	`x y larg alt`	*remove todas quadras que estiverem inteiramente dentro do retângulo determinado pelos parâmetros do comando. Obs. Este comando, em particular, deve remover uma quadra inserida pelo comando* **q** *com idênticos parâmetros* `x`, `y`, `larg` *e* `alt`. *Saída. arquivo .txt: deve apresentar os ceps das quadras removidas. arquivo .svg: deve apresentar o retângulo correspondente à região da consulta; as quadras removidas não devem aparecer*
dle	`t x y larg alt`	*Semelhante ao dq. Remove equipamentos urbanos do tipo* `t` *dentro da região. (reporta o id e não mostra no .svg) t pode ser qualquer combinação:* `h` *(hidrante),* `s` *(semaforo),* `r` *(rádio-base)*
Dq	`raio x y`	*Remove todas as quadras que estiverem inteiramente contidas dentro do círculo de centro em (x,y) e de raio* `raio`. *Reporta no arquivo .txt o cep das quadras. Quadras removidas não devem ser mostradas no .svg.*
Dle	`t x y raio`	*Semelhante a dle. Remove os equipamentos urbamos do tipo t dentro da região. (Reporta o* `id`*, não mostra no .svg)*
cc	`( cep | id ) cstrk cfill`	*Muda as cores do contorno e do preenchimento da quadra identificada por* `cep` *ou do equipamento urbano identificado por* `id`. *Saída: arquivo.svg: quadra ou equipamento pintados com as novas cores.*
crd?	`( cep | id )`	*Imprime no arquivo .txt as coordenadas e a espécie do equipamento urbano de um determinado cep ou com uma determinada identificação. Atenção: quadras e equipamentos que foram removidos por comandos d* e D* devem efetivamente ter sido removidos.*
crb?		*Determina quais são as duas rádio-bases mais próximas. Saida: arquivo.svg: circular a rádio-base destacando-a; arquivo.txt: reportar id das torres e a distância.*
Comandos do arquivo .qry		

EXEMPLOS

```
cq blue black
ch red yellow
ct 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 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 40.00 60.00
r 6 cyan yellow 121.0 46.0 100.0 30.0
c 2 grey magenta 120.0 45.0 15.0
r 4 cyan yellow 10.0 150.0 90.0 40.0
c 5 grey magenta 230.0 180.0 13.0
```

a1.geo

```
Dq 100.50 99.9 100.0
crd? cep_001-10
crd? h-12
crb?
dle hr 20.0 30.0 50.0 60.0
crb?
Dle s 100.50 99.9 100.0
Q? 100.50 99.9 100.0
```

q1.qry

A SAIDA

A arquivo `.geo` (com os novos comandos) deve continuar produzindo as mesmas saídas da fase anterior. As quadras devem ser renderizados como retângulos e pintadas como determinado pelo comando `cq`. Os outros equipamentos urbanos devem ser similarmente pintados com as cores de contorno e de preenchimento informadas pelos respectivos comandos.

Caso o arquivo .geo seja processado com um arquivo .qry, outro arquivo .svg deve ser produzido. O nome deste outro arquivo deve ser a concatenação dos nomes-base de cada um dos arquivos de entrada (no exemplo, a1-q1.svg) O arquivo `.svg` deve ser produzido após o arquivo `.qry` tiver sido processado.

A IMPLEMENTAÇÃO

O TAD Listas mostrado em sala de aula deve ser completamente implementado. As quadras e equipamentos urbanos devem ser armazenados em listas diferentes.

Resolver a consulta crb? equivale a resolver o **Problema do Par de Pontos Mais Próximos**. Este é um problema clássico em Geometria Computacional.[9] A implementação desta consulta deve,

9 Consulte Capitulo 33 (Geometria Computacional) do livro do Cormen.

portanto, usar o algoritmo adequado. Tal algoritmo depende da ordenação dos pontos. O algoritmo a ser usado deve ser o heapsort ou o mergesort.[10]

10 Você deve implementar e testar os dois. Escolher um deles e justificar sua escolha.

FASE III (Rascunho)

Nossa cidade já possui quadras e equipamentos urbanos. Agora começaremos a povoá-la com moradores e estabelecimentos comerciais. O corpo de bombeiros da cidade ficou entusiasmado com o `siguel` e pediu para que o sistema pudesse encontrar rapidamente o hidrante mais próximo de uma determinada localização.

KD-TREE

> *Uma **árvore k-d** (abreviação para a árvore k-dimensional) é uma estrutura de dados de particionamento do espaço para a organização de pontos em um k-dimensional espaço. Árvores k-d são estruturas úteis para uma série de aplicações, tais como pesquisas envolvendo pesquisa multidimensional de chaves (e.g. busca de abrangência e busca do vizinho mais próximo). Árvores k-d são um caso especial de árvores de particionamento binário de espaço.*
>
> *Uma árvore k-d é uma árvore binária em que cada nó é um ponto k-dimensional. Cada nó não-folha pode ser considerado implicitamente como um gerador de um hiperplano que divide o espaço em duas partes, conhecido como semiespaço. Os pontos à esquerda do hiperplano são representados pela subárvore esquerda desse nó e pontos à direita do hiperplano são representados pela subárvore direita. A direção do hiperplano é escolhida da seguinte maneira: cada nó na árvore é associado a uma das k-dimensões, com o hiperplano perpendicular a esse eixo dimensional. Assim, por exemplo, se para uma determinada operação de split o eixo "x" é escolhido, todos os pontos da subárvore com um valor "x" menor que o nó irão aparecer na subárvore esquerda e todos os pontos com um valor "x" maior vão estar na subárvore direita. Nesse caso, o hiperplano seria definido pelo valor de x do ponto, e o seu normal seria a unidade do eixo x.*
>
> **Copiado da Wikipedia: https://pt.wikipedia.org/wiki/%C3%81rvore_k-d**

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.

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

	FONE	POS
**	**32147530**	**5**
	32315623	2
	32569872	20
	43670987	4
	56783456	3
	58769987	7
	75640967	16
	76900098	17
	76989876	9
	78654320	8
	80453587	11
	83434365	1
	84563589	10
	89099788	14
	89449922	19
	89766677	18
	89998742	15
	91676300	13
	91997204	12
	98795432	6

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 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 e tem um dado nome.*
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	
m?	`cep`	*Moradores da quadra cujo cep é cep. Mostra mensagem de erro se quadra não existir.* *SAIDA: arquivo .txt => listar o nome, endereço, celular e operadora dos moradores da quadra.*
mr?	`x y larg alt`	*Moradores das quadras que estão inteiramente dentro da região determinada.* *SAIDA: arquivo .txt => para cada quadra da região, produzir uma saida similar a da consulta m?.*
dm?	`cpf`	*Imprime todos os dados do morador identificado pelo cpf.* *SAIDA: arquivo .txt => dados pessoais, seu endereço, o número de seu celular (se houver) e respectiva operadora, e coordenada geográfica de onde mora.* *arquivo .svg => colocar um circulo (ou retângulo, ou outra forma geométrica) no lugar onde mora com seu cpf*
de?	`cnpj`	*Imprime todos os dados do estabelecimento comercial identificado por cnpj.* *SAIDA: arquivo .txt => dados do estabelecimento (nome, descrição do tipo, etc), endereço e coordenada geográfica.* *arquivo .svg => colocar um circulo (ou retângulo, ou outra forma geométrica) na coordenada geográfica do estabelimento.*

rip	cpf	*Infelizmente pessoa identificada por cpf morreu. Todos os dados armazenados sobre ela devem ser apagados (dados pessoais, dados sobre moradia, etc).* *SAIDA: arquivo .txt => imprimir os dados removidos de cada uma das estruturas. Por exemplo:* *RIP: José Silva, portador CPF, do sexo Masculino, nascido à 01/01/1920, residia no endereço CEP/NUM/FACE.....* *arquivo .svg: colocar um losango preto, com um cruz branca no interior na residência do defunto.c*
ecq?	cep	*Lista os estabelecimentos de uma determinada quadra.* *SAIDA: arquivo .txt =>mostrar cnpj, nome, tipo e endereço de cada estabelecimento comercial.*
ecr?	tp [x y larg alt]	*Lista os estabelecimentos comerciais de um da tipo de, se presente, uma determinada região.*
tecq?	cep	*Lista os tipos de estabelecimentos comerciais de uma dada quadra.* *SAIDA: arquivo .txt =>mostrar, para cada tipo de estabelecimento comercial, listar o nome dos estabelecimentos daquele tipo.*
tecr?	x y larg alt	*Quais são os tipos de estabelecimentos comerciais existentes numa dada região.*
hmpe?	cep face num	*Qual é o hidrante mais próximo do endereço fornecido nos parâmetros?* *SAIDA: arquivo .txt: mostrar todos os dados sobre o hidrante e a distância.* *arquivo .svg:colocar um X na posição do endereço e traçar uma linha tracejada entre o X e o hidrante.*
hmp?	id	*Qual é o hidrante mais próximo da rádio-base identificada por id?* *SAIDA:arquivo .txt: dados dos hidrante, da rádio-base e distancia entre eles* *arquivo .svg: traçar uma linha tracejada entre o hidrante e a rádio-base*
fec	cnpj	*O estabelecimento comercial identificado por cnpj fechou. Os dados a cerca dele devem ser removidos.* *SAIDA: arquivo .txt. Mostrar os dados dos estabelecimento fechado*

mud	`cpf cep face num compl`	*A pessoa identificada por cpf muda-se para o endereço determinado pelos parâmetros. SAIDAS: arquivo .txt. Mostrar os dados da pessoa (nome, etc), o endereço antigo e o novo endereço. arquivo. svg: traça uma seta (linha grossa) da endereço antigo ao endereço novo.*
mudec	`cnpj cep face num`	*O estabelecimento identificado por cnpj muda-se para novo endereço. Similar ao mud. Obs. traçar linha com cor diferente da de mud.*
dpr	`x y larg alt`	*Desapropria região. Apagar quadras, hidrantes, etc dentro da região; etc A pessoa não morre, mas deixa de ser morador. SAIDA: arquivo .txt => listar o que foi removido arquivo .svg => elementos removidos não devem aparecer.*
Novos comandos do arquivo .qry		

Implementação

Deve ser implementado um módulo para o tipo abstrato Tabela de Espalhamento. As quadras e os equipamentos urbanos devem continuar a ser armazenados em kd-trees. 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.

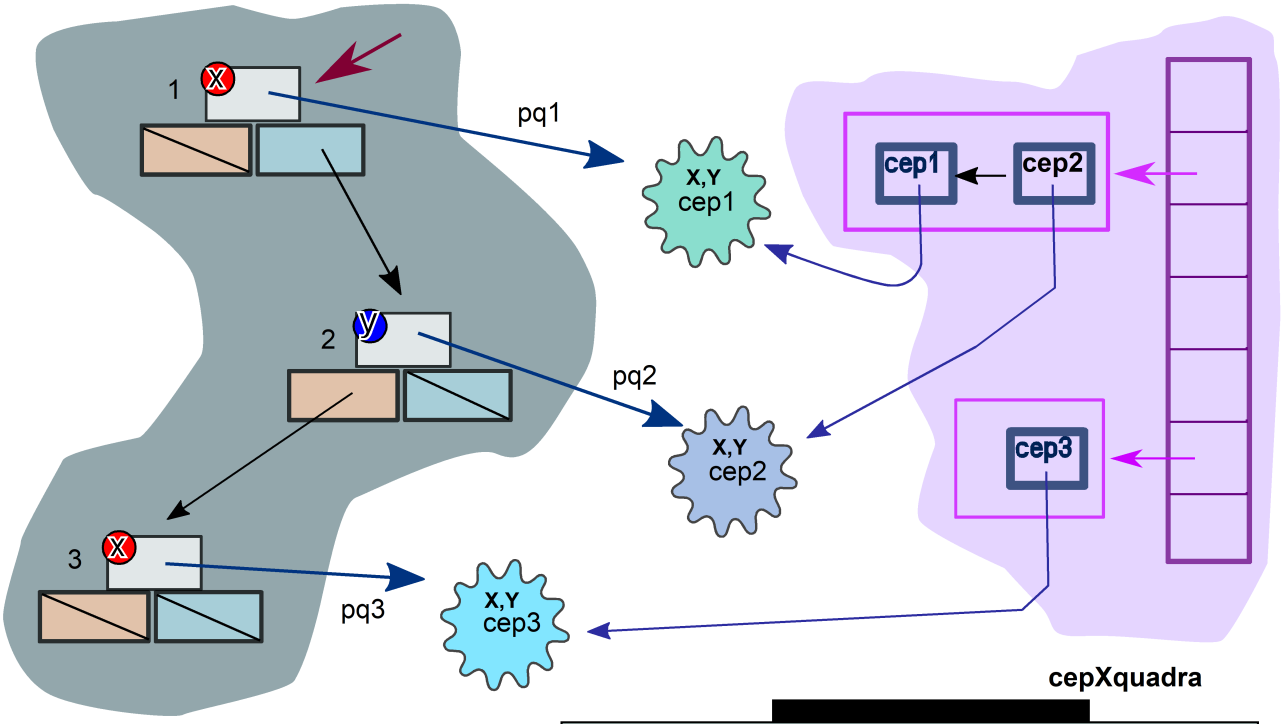
Índices/Dicionários que devem ser mantidos e usados em consultas (pelo menos):

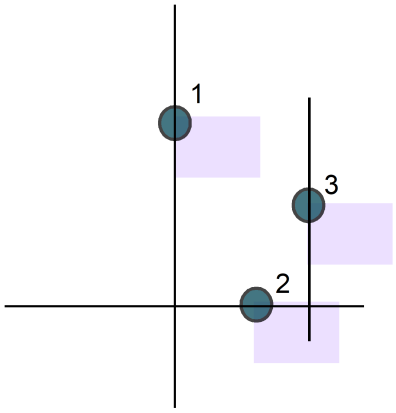
- cpf X cep: dado o cpf, retorna o cep quadra onde a pessoa reside
- tipo estabelecimento comercial X descrição: dado o código do tipo do estabelecimento, retorna a descrição
- dados de uma pessoa: dado um cpf, retorna os dados pessoais daquela pessoa
- cep X quadra: dado um cep, retorna os dados da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

A figura esquematiza a relação entre a kd-tree e a tabela de espalhamento.

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.

OBS.: O comando crb? deve ser adequado para iniciar o processamento a partir de uma árvore cadê.






```
kdree  kdtQda = criaArv(....)
TabelaEspalhamento cepXquadra = criaTabEsp(...)
pq1 = criaQuadra("cep1", ...)
pq2 = criaQuadra("cep2", ...)
pq3 = criaQuadra("cep3", ...)
insereKdt(kdtQda,pq1)
insereKdt(kdtQda,pq2)
insereKdt(kdtQda,pq3)
insereTE(cepXquadra,"cep1", pq1)
insereTE(cepXquadra,"cep2", pq2)
insereTE(cepXquadra,"cep3", pq3)
```

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 três critérios: **(a)** video de 10min demonstrando que você executou completamente e eficazmente o trabalho proposto (subir no Youtube); **(b)** inspeção do código-fonte; **(c)** compilação e testes do executável (o aluno deve entregar os fontes).

O Que Entregar

Submeter a sala Moodle, como de costume, o arquivo .zip com os fontes . O zip deve conter 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` ***arq***`.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**.

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