PROJETO 2017

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/**

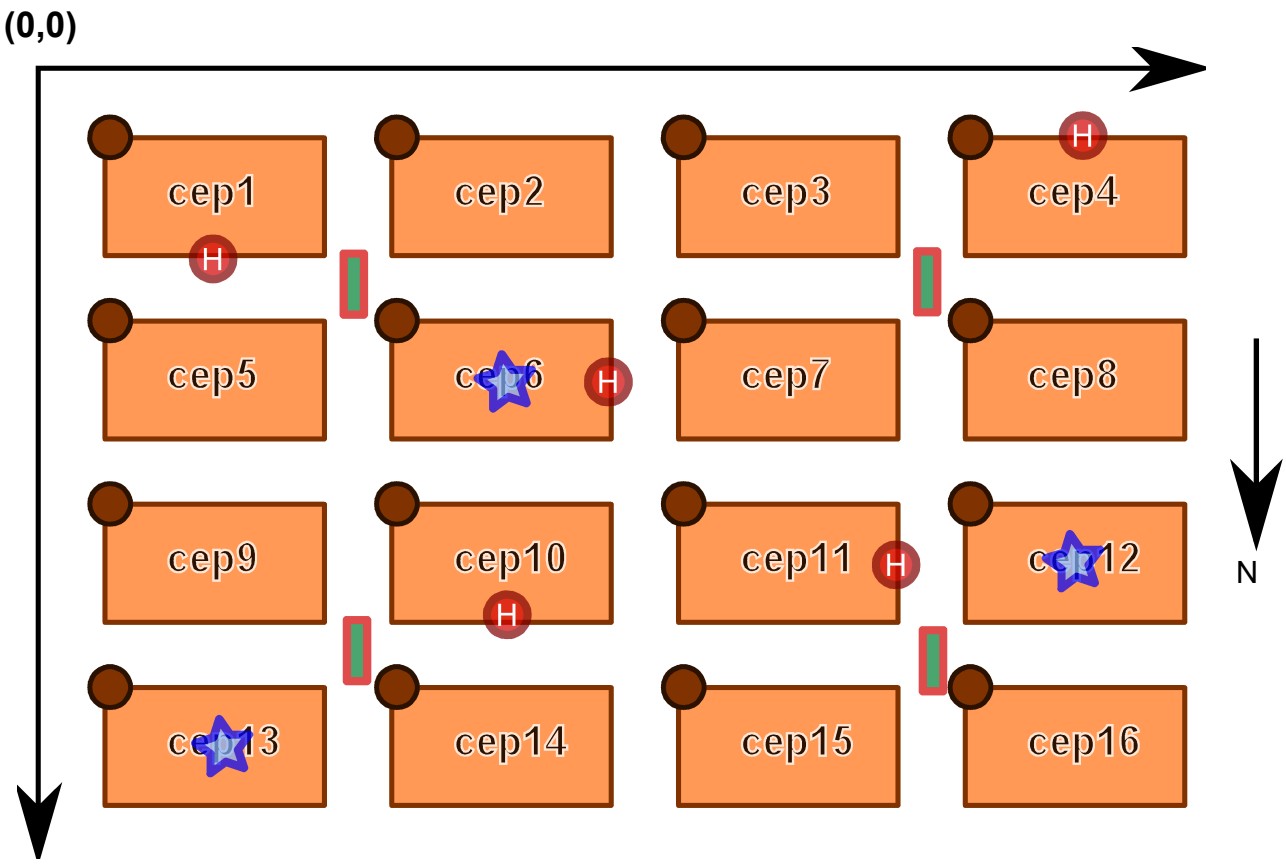
O sistema será implementado incrementalmente e em fases. É importante enfatizar que em cada fase o sistema evolui, isto é, novas funcionalidades são acrescentadas, requisitos de implementação podem ser mudados, porém, as funcionalidades existentes **devem continuar funcionais**.

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).

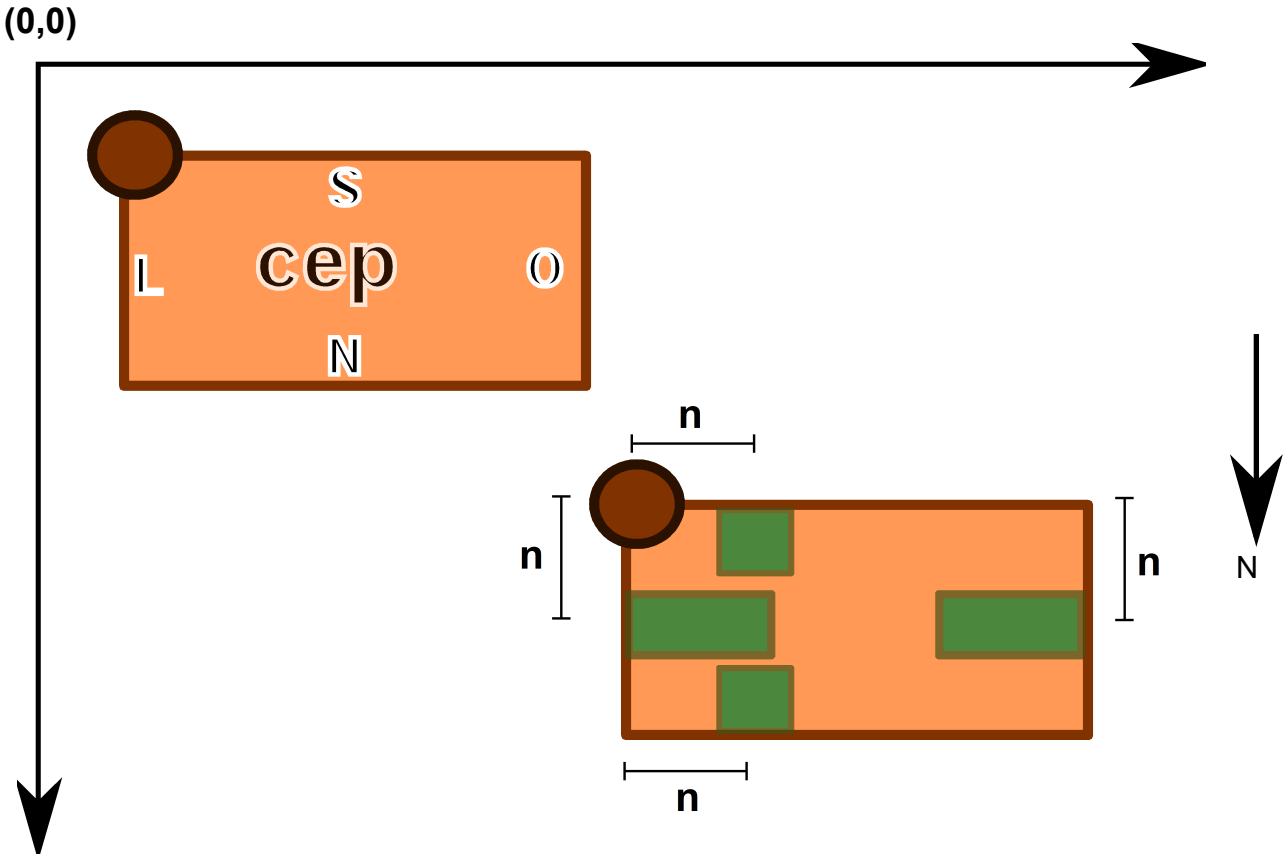
O sistema manipulará o mapa de uma cidade e algumas informações relacionadas.

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.[5]

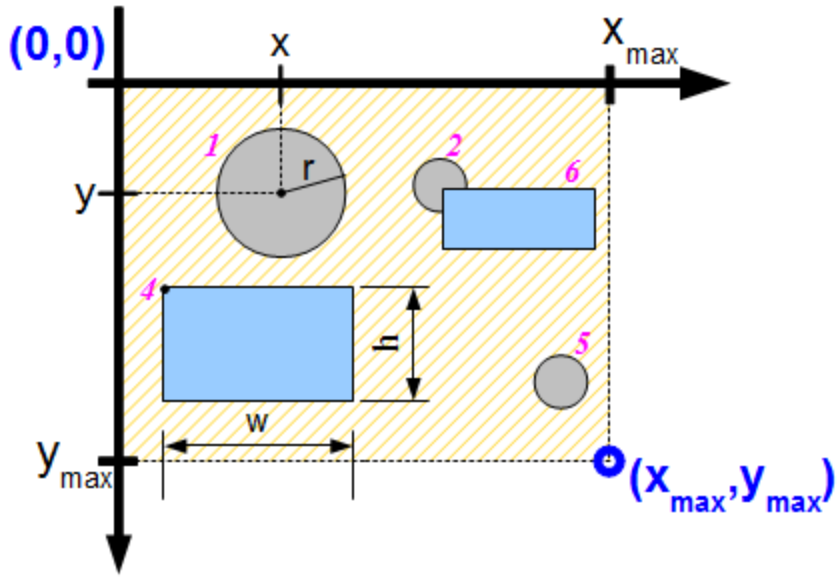


Ilustração 1

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).

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

5

- `cor`: string. Cor válida dentro do padrão SVG.[6]

comando	parâmetros	descrição
c	`i r x y cor`	*desenhar círculo*
r	`i w h x y cor`	*desenhar retângulo: w é a largura do retângulo e h, a altura.*
o	`j k`	*As formas geométricas cujos indentificadores são j e k se sobrepõem?*[7]
i	`j x y`	*O ponto (x,y) é interno à j-ésima forma geométrica?*[8]
d	`j k`	*Qual é a distância entre os centro de massa das formas geométricas i e j?*
a	`sufixo cor`	*Cria um arquivo svg, mostrando apenas os centros dos círculos e os cantos dos retângulos criados pelos comandos c e r. O nome do arquivo gerado deve ser* nomebase-*sufixo.svg. Estes pontos devem ser desenhados usando a cor* `cor`

A figura abaixo mostra um exemplo de um arquivo de entrada (consistente com a Figura 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-p1.svg) mostrando apenas os cantos dos retângulos e os centros dos círculos.

```
c 1 50.00 50.0 30.00 grey
r 6 121.0 46.0 100.0 30.0 cyan
c 2 120.0 45.0 15.0 grey
r 4 10.0 150.0 90.0 40.0  cyan
c 5 230.0 180.0 13.0 grey
o 2 6
o 1 5
a p1 red
#
```

***a01*.geo**

A Saída

O programa deverá produzir um arquivo **`.svg`** e um arquivo **`.txt`** ambos com o mesmo

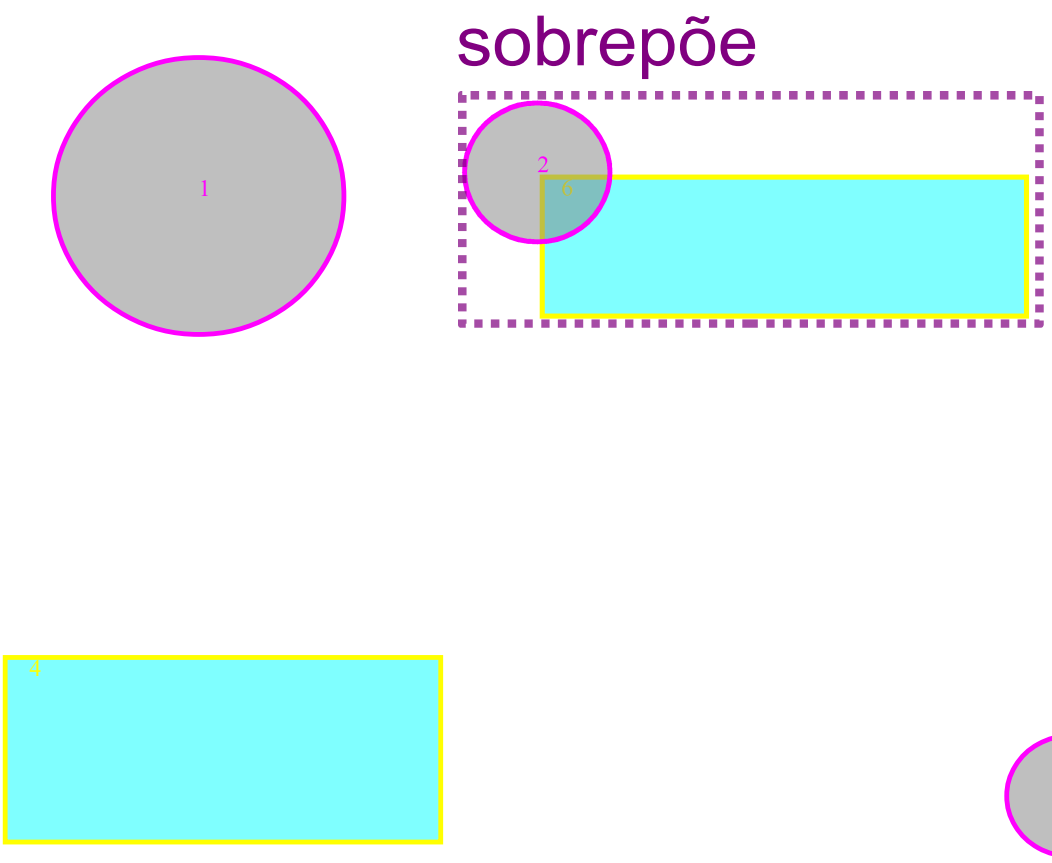
6 http://www.december.com/html/spec/colorsvg.html. https://www.w3.org/Graphics/SVG/IG/resources/svgprimer.html

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

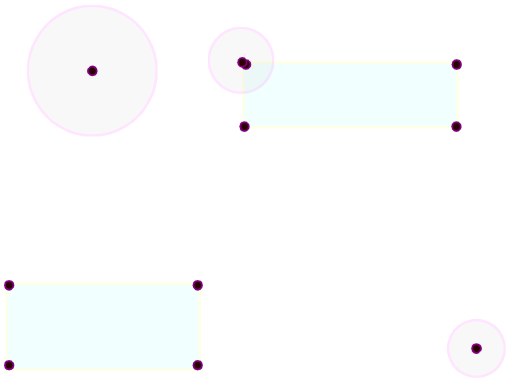
8 Um ponto na borda da figura pertence à figura, **mas não é interno** à figura.

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.



A operação **`a`** deve produzir um outro arquivo .svg, com nome *`nomebase`*`-sufixo.svg`, (no exemplo, `a01-p1.svg`) contendo apenas os cantos dos retângulos e centro dos círculos referentes aos comandos **`c`** e **`r`** processados até o momento, exemplificado na figura abaixo.[9]



O processamento de um arquivo `.geo` deve também produzir um arquivo-texto contendo o

9 A figura do exemplo apresenta as "sombras" dos círculos e retângulos apenas para facilitar o entendimento. No arquivo a01-p1.svg são apenas produzidos os pontos escuros.

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 Programa

O nome do programa deve ser `t2` e aceitar dois parâmetros:

t2 -f *arq*.geo -o *dir* -n *num*

O primeiro parâmetro (**-f**) especifica o nome do arquivo de entrada e o segundo parâmetro (**-o**) indica o diretório onde os arquivos de saída (***arq*.svg e *arq*.txt**) deve ser colocado. Note que o nome do arquivo pode ser precedido por um caminho absoluto ou relativo e ***dir*** é um caminho absoluto ou relativo. O parâmetro **-n** informa o número máximo de operações **r** e **c** contidas no arquivo.

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

- `t2 -f /home/ed/testes/t001.geo -o /home/ed/alunos/aluno1/o/ -n 100`
- `t2 -f /home/ed/testes/t001.geo -n 1000 -o /home/ed/alunos/aluno1/o`
- `t2 -n 10000 -f ./testes/t001.geo -o /home/ed/alunos/aluno1/o/`
- `t2 -f ./testes/t001.geo -n 100000 -o ./alunos/aluno1/o`

FASE II

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

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

Alguns comandos de atualização e consulta podem ser colocados em um arquivo `.qry`:[10]

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`. *No arquivo .txt, deve apresentar os ceps das quadras removidas*
dh	x y larg alt	*Semelhante ao dq. Remove os hidrantes dentro da região. (reporta o id)*
ds	x y larg alt	*Semelhante dq. Remove os semáforos da região.*
dt	x y larg alt	*Semelhante dq. Remove as torres de celular dentro da região.*

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

comando	parâmetros	
Dq	`x y raio`	*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.*
Dh	`x y raio`	*Semelhante a Dq. Remove os hidrantes dentro da região. (Reporta o id)*
Ds	`x y raio`	*Semelhante a Dq. Remove os semáforos dentro da região. (Reporta o id)*
Dt	`x y raio`	*Semelhante a Dq. Remove as torres de celular dentro da região. (Reporta o id)*
crd?	`( cep | id )`	*Imprime no arquivo .txt as coordenadas e a espécie do equipamento urbano de um determinado cep ou com uma determinada identificação.*
Comandos do arquivo .qry		

EXEMPLOS

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

a1.geo

```
Dq 100.50 99.9 100.0
crd? cep_001-10
crd? h-12
```

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. O arquivo `.svg` deve ser produzido após o arquivo `.qry` (se existir) tiver sido processado.

Além disso, deverá ser produzido, no diretório de saída[11], um arquivo denominado `resumo.txt` com um resumo da bateria de testes.[12] Este arquivo deverá conter uma linha para cada um dos testes realizados. Cada linha deve ter os seguintes campos (separados por \t):

```
nome-arq.geo  #ins #cpi #del #cpd
```

resumo.txt

O primeiro campo é o nome do arquivo processado e os outros campos são, respectivamento, os números totais de quadras inseridas, o número total de comparações para inserir as quadras; o número total de quadras removidas e o número total de comparações para removê-las.

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.

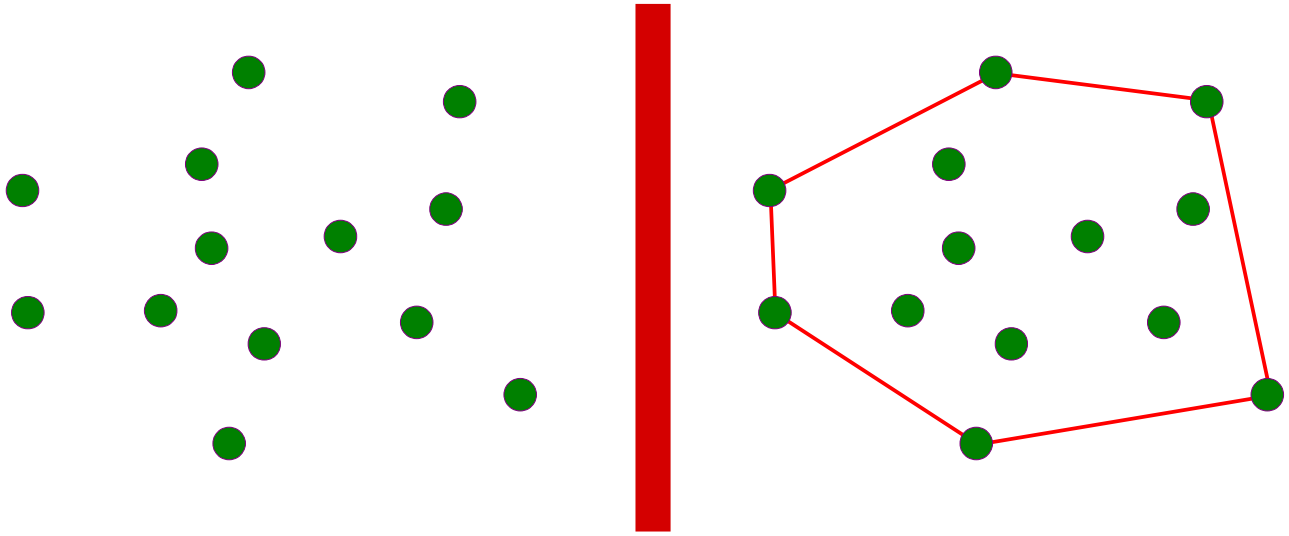
11 Opção `-o`.

12 O início da bateria de testes é indicado pela opção `-acc0 e a continuação indicada por -acc`.

FASE III (RASCUNHO)

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, poderá ser utilizado algum dentre os seguintes algoritmos de ordenação: quicksort, mergesort ou heapsort.

Uma breve explicação sobre um dos algoritmos (Graham Scan) para o cálculo da envoltória convexa pode (deve) ser vista no YouTube:

`https://youtu.be/0HZaRu5IupM`

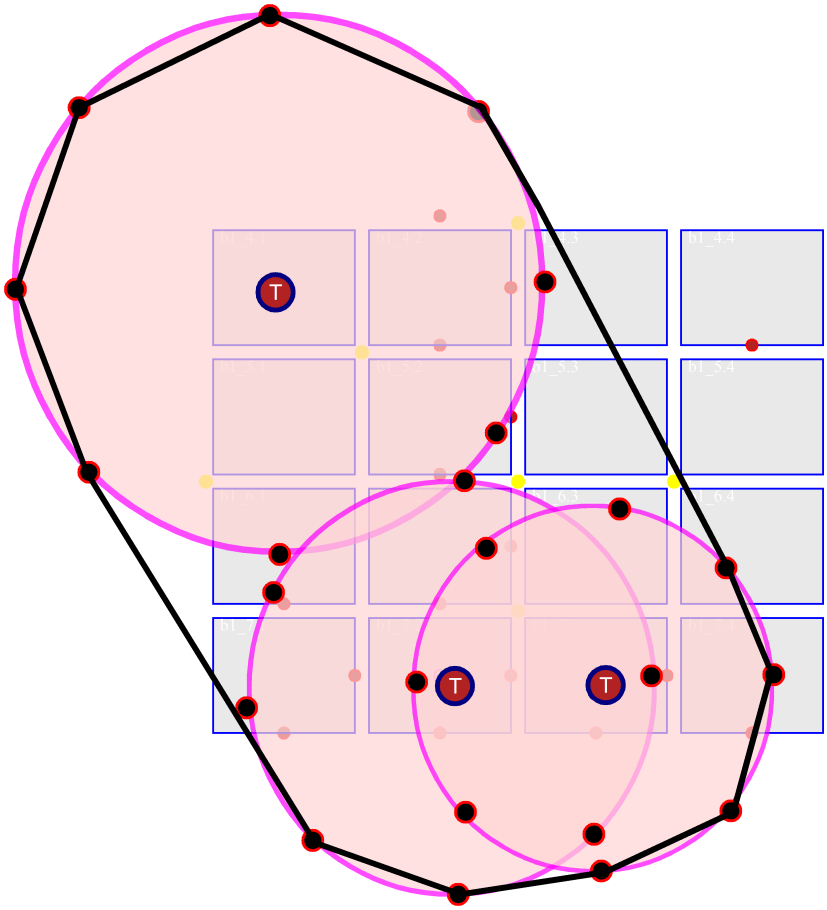
Outros materiais são facilmente encontrados usando as palavras-chave: *convex hull, computational geometry, graham scan*.

POLIGONO DE COBERTURA DE UM CONJUNTO DE RÁDIOS-BASE

É uma aproximação da área real de cobertura de um conjunto de rádio-bases de uma região.

Note que a figura abaixo apresenta um bairro com 3 rádio-bases (círculo rotulado com T). A abrangência de cada rádio-base está indicada por uma circunfência cujo centro é a rádio-base. Se considerarmos alguns pontos nos limites de cada área de abrangência e calcularmos a envoltória convexa deste conjunto de pontos, obtemos o polígono de cobertura.[13] Note que o polígono de cobertura pode excluir áreas que efetivamente estão cobertas e incluir outras que não estão cobertas (por isso, é uma aproximação).

13 P*olígono de cobertura* é apenas um conceito criado para este projeto.



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

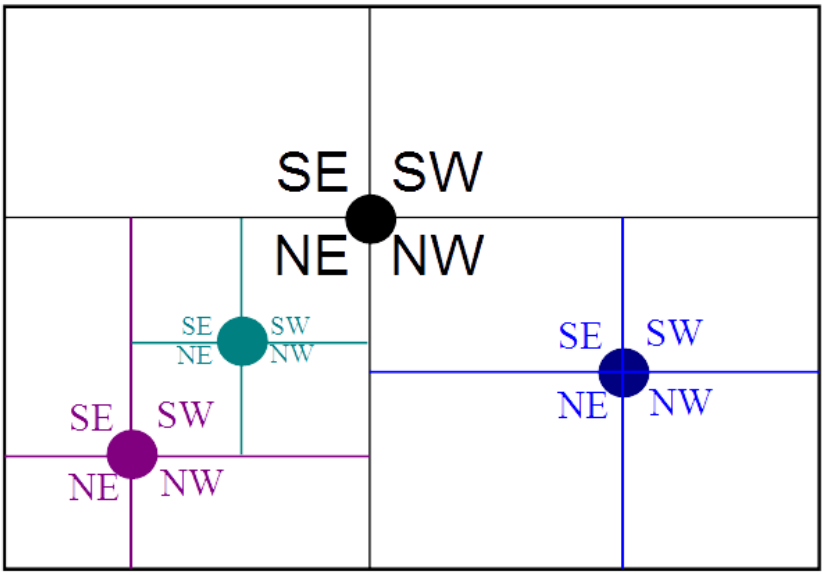
The point quadtree is an adaptation of a binary tree used to represent two-dimensional point data. It shares the features of all quadtrees but is a true tree as the center of a subdivision is always on a point. It is often very efficient in comparing two-dimensional, ordered data points, usually operating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

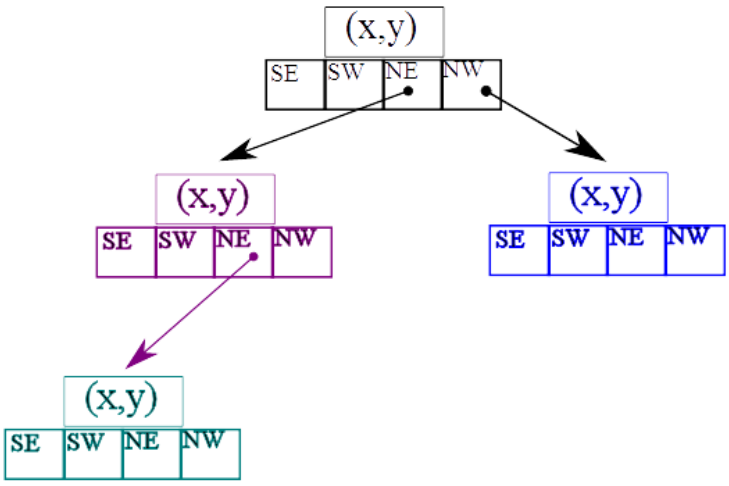
Point quadtrees are constructed as follows. Given the next point to insert, we find the cell in which it lies and add it to the tree. The new point is added such that the cell that contains it is divided into quadrants by the vertical and horizontal lines that run through the point. Consequently, cells 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.





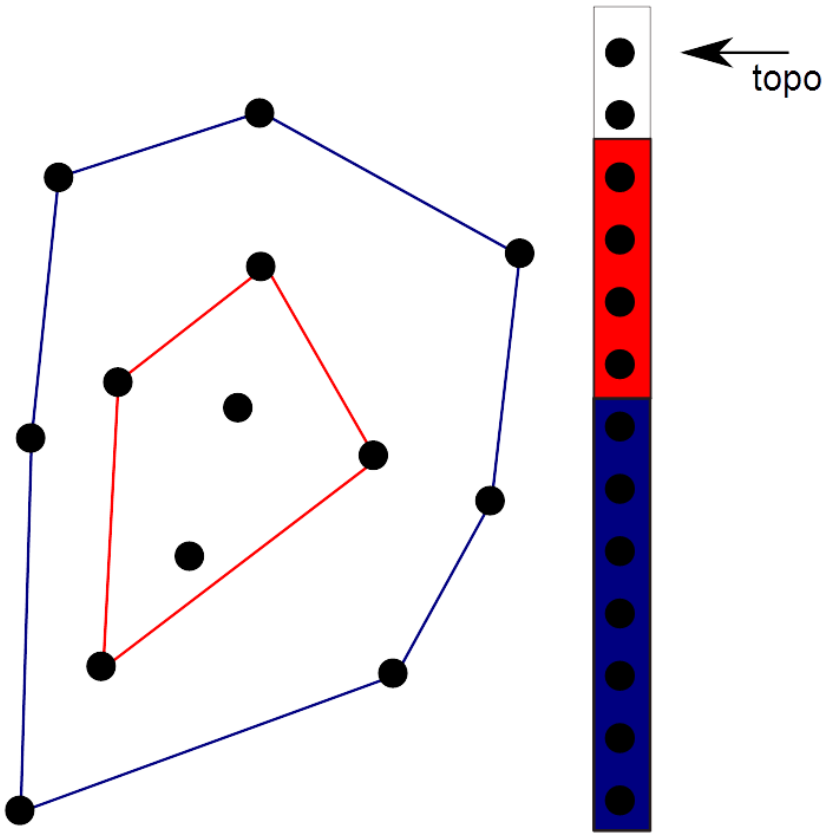
Como mencionado acima, a estrutura de uma quadtree depende da ordem de inserção dos pontos (no exemplo, o ponto preto foi o primeiro a ser inserido). Talvez a estratégia de inserção descrita a seguir possa produzir árvores razoavelmente balanceadas.

Considere a figura abaixo. O poligono 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.[14]

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

14 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 que aparecem no arquivo .geo?

de entrada `.geo`. É necessário apresentar evidências para os argumentos apresentados (por exemplo, mostrar gráficos de testes com e sem estratégia de inserção sugerida)



NOVOS COMANDOS E CONSULTAS

A seguir são mostradas duas tabelas com os novos comandos e novas consultas que são acrescentados aos arquivos `.geo` e `.qry`.

comando	parâmetros	
hI	`id vazao`	*Informa a vazão (l/s) do hidrante* **id**. *O hidrante de código* **id** *deve ter sido inserido anteriormente na cidade por um comando h.*
tI	`id r`	*Informa o raio (m) de alcance da radio-base* **id**. *A rádio-base* **id** *deve ter sido inserido anteriormente por um comando t.*
sI	`id t`	*Informa o tempo (s) do ciclo (verde, amarelo, vermelho) do semaforo* **id**. *O semáforo deve ter sido inserido anteriormente por um comando s.*
Novos comandos do arquivo .geo		

comando	parâmetros	
pc?	`sufixo [ x y larg alt ]`	*calcula o polígono de cobertura das rádio-bases da cidade (ou daquelas contidas na região retangular definida pelos parâmetros). Produz um arquivo svg mostrando a cidade e o polígono de cobertura. A área de cobertura de cada rádio-base deve estar evidenciada na saída svg. O nome do arquivo de saída deve ser uma composição do nome do arquivo .geo com o nome do arquivo .qry e do sufixo informado no comando:* **nomegeo-nomeqry-sufixo**.*svg*
ac?	`[ x y larg alt ]`	*calcula e imprime no arquivo .txt a área (m^2) do polígono de cobertura das rádio-bases da cidade (ou daquelas contidas na região retangular definida pelos parâmetros).*
Comandos do arquivo .qry		

A IMPLEMENTAÇÃO

Nesta fase, as quadras e os equipamentos urbanos devem ser armazenados em diferentes quadtrees. Também, como comentado anteriormente, deve-se implementar um dos algoritmos de ordenação mencionados.

Como anteriormente, o módulos deve ser bem projetados e bem documentados. Escrever o .h antes de escrever o .c!!! Isto é muito importante!!

Lembrar que o arquivo `resumo.txt` deve continuar a ser produzido!

Como mencionado anteriormente, o aluno deverá justificar se a estratégia de inserção proposta é melhor do que inserir as quadras e os equipamentos urbanos na ordem em que aparecem no arquivo .geo. Para tal, provavelmente será necessário instrumentar o código para: ora executar a inserção com a estratégia de inserção proposta, ora executar os mesmos testes sem tal estratégia; contabilizar as operações sigficativas dos algoritmos de busca, inserção e delição dos dados geo-referenciados; entre outros. Uma vez coletados os dados, o aluno poderá fazer gráficos, etc, para tirar suas conclusões.

O QUE ENTREGAR

Nesta fase, o aluno deve produzir um breve vídeo (aproximadamente 10 minutos), explicando os principais detalhes referente a implementação do projeto. O vídeo deve mostrar quais são os módulos do sistema e como se relacionam; qual o método de ordenação escolhido e como ele foi ajustado para ordenanar elementos geométricos; deve explicar os principais detalhes sobre a implementação da quadtree e responder à pergunta “A quadtree pode ser usada para armazenar dados não espaciais? Se sim, como?“. O vídeo, também, deve mostrar algumas (poucas)

execuções do programa. O vídeo deve mostrar as conclusões sobre a estratégia de inserção proposta e sustentá-las com evidências. Espera-se que o vídeo mostre que o aluno fez um trabalho de qualidade.

Como de costume, submeter a sala Moodle o arquivo ***nome*.zip,** contendo os fontes e um arquivo `.html` com o link para o vídeo no youtube.

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 três critérios: **(a)** compilação e testes do executável; **(b)** inspecção do código-fonte; **(c)** análise do video.[15]

A nota será proporcional ao número de testes executados com sucesso. Portanto, caso o programa não compile ou não execute corretamente, a nota poderá ser muito baixa!

15 Tenha especial cuidado com a apresentação de suas conclusões sobre a estratégia de inserção na quadtree.

RESUMO DOS PARÂMETROS DO PROGRAMA SIGUEL

Parâmetro / argumento	Opcional	Descrição
`-acc0`	S	Início da bateria de testes
`-acc`	S	Continuação da bateria de testes
`-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**.
`-id`	S	Se presente, deve imprimir na saída padrão o nome e a matrícula do aluno
`-n` ***num***	X	**Descontinuado**
`-o` ***path***	N	Diretória-base de saída (**BSD**)
`-q` ***arq.qry***	S	Arquivo com consultas

ATENÇÃO: novo parâmetro do programa acrescentado.