Q 5.2: Consider the rough guide to worst-case time complexity of algorithms (Table 5.2, pp. 171). Classify the complexity of following algorithms into classes of constant, logarithmic, linear, n*log(n), polynomial and exponential: Assume relevant line-strings have N segments, polygons have N vertices and graphs have N vertices and E edges.
1. Douglas-Pucker algorithm to discretize arcs (sec. 5.2.3, pp. 176-177) Linear
2. Compute area of a simple polygon (sec. 5.5.1, pp. 195-197) Linear
3. Compute centroid of a polygon (sec. 5.5.1, pp. 195-197) Linear
4. Point in polygon (sec. 5.5.2, pp. 197-199) Binary
5. Intersection or a polygon-pair, each with N vertices (sec. 5.5.3, pp. 201-202) Linear
6. Depth-first or breadth-first graph traversal (sec. 5.7.2, pp. 213-217) Sub Linear
7. Single-pair shortest path in a graph (sec. 5.7.2, pp. 213-217) Polynomial
8. All-pair shortest paths in a graph (sec. 5.7.2, pp. 213-217) Polynomial
9. Hamiltonian circuit (or Traveling salesman problem) (sec. 5.7.2, pp. 213-217) Exponential

Q 5.4: Consider alternative data models for spatial object domain (section 5.3, pp. 177-187) including spagetti, node-arc-area, DCEL, etc. Which data model is closest to the following popular formats:
1. GML Simple Features, NAA, DCEL
2. ESRI Shapefile: wikipedia article. Spagetti

Briefly justify your answer assuming Euclidean space.

The defining factor between these two popular formats is the ability to store topological information about the features they represent. An ESRI shapefile cannot store topological information where as GML Simple Features may store that information.

Chapter 6: Structures and Access Methods

Q 6.2: Compare and contrast R-trees and R+-trees. Consider a minimum orthogonal bounding rectangle for rectangle T and L. Add it to the set of rectangles in Fig. 6.29 (pp. 253) and Fig. 6.30 (pp. 254). Redraw the R-tree and R+-tree in Fig. 6.29 and 6.30 if this new rectangle is inserted. Briefly justify your solutions by recalling how R-tree and R+-tree deal with large objects based on narrative of section 6.6.2 (pp. 252-254)?
Regular R trees break areas into rectangles however they have trouble with rectangles that are large relative to the total space of the data and any overlapping of rectangles as records are only stored once regardless of overlap. R+ trees store the data of leaf nodes in any rectangle that they overlap thereby making the data somewhat redundant but a more efficient mode of storing such data.

This question was a little confusing since if a bounding rectangle for the T and L rectangles was inserted the whole structure of bounding rectangles would benefit from dividing and adjusting the extents of all of the other rectangles to make the index more efficient. I don’t think the extents would remain the same in this case but it is an interesting exercise to try and see how you would handle the tree structures without changing the bounding rectangles.

- **Q 6.4**: Revisit the first problem under chapter 5. Name efficient data-structure and access methods for problems, which can be solved algorithmically.

  4) Region Quadtree
  5) Quaternary Triangular Mash
  6) Quaternary Triangular Mash
  7) R Tree
  8) R Tree
  9) R+ Tree
**Lab**

SQL> select a.id as Activity_ID, r.osm_id as Road_ID from activities a, roads_major r
2   where mdsys.SDO_NN(r.geom, a.geom, 'sdo_num_res=1') = 'TRUE';

SQL> select r.osm_id as Road_ID, r.name as Name, n.osm_id as Water_ID from roads_major r, natural n
2   where n.type = 'water' and mdsys.SDO_OVERLAPS(r.geom, n.geom) = 'TRUE';

SQL> select distinct n1.osm_ID as Riverbank_ID1, n2.osm_ID as Riverbank_ID2 from natural n1, natural n2
2   where n1.osm_ID <> n2.osm_ID and n1.type = 'riverbank' and n2.type = 'riverbank' and mdsys.SDO_OVERLAPS(n1.geom, n2.geom) = 'TRUE';

SQL> select n1.osm_id, n1.type, n2.osm_id, n2.type from natural n1, natural n2
2   where n1.type = 'riverbank' and n2.type = 'forest' and mdsys.SDO_TOUCH(n1.geom, n2.geom) = 'TRUE';

**Trends**

There were little to no revisions based on the reviewer comments. The comments left by team 6 were related mostly to our final two slides being difficult to read. This was mostly due to our slides not being presented using Microsoft's Powerpoint application. The laptop we loaded our slides on was using an open source application for presentations, and the owner warned us that the fonts and layout of the slides would be affected. So, next time we present we will ensure that we a device that is loaded with Microsoft Powerpoint.

Team 6 also mentioned that we didn't ask any questions of the audience. This was mostly due to the general overview of the subject itself, but there are some questions that could be thrown into the presentation. What are the types of databases? What guarantees about data do databases have? (Then show the list after audience answers.) What are the advantages of relational databases and object-orientation databases? Has there been any recent developments in GIS databases that anyone has seen.

Unfortunately, Team 4 does not have their review of our mid-term presentation online. When I click on the link to the review (Group 2 Critique) found at this address http://www-users.cs.umn.edu/~hendawi/CSci5980WebPage/, it links me to the review for Group 8.

**Project**

Project: Implementation of advanced GIS features for E-911 services.
Motivation
- Rise of location aware smartphones and E-911 has given emergency responders new tools to accurately determine where to direct resources.
- Potential for application of additional GIS features to serve as advanced options to assist E-911 dispatchers and responders.

Goal:
- Identify value added GIS features for E-911 and implement features using web based client/server architecture.
- Demonstrate one or more of the implemented features in the classroom.

Assumptions:
- Information from call is collected via automated/manual methods and input into system outside of our implementation
- Server/Client service, where client requests relevant information from server
  - E.g. Robberies will automatically forward to Police, fires forward to Fire Department, car accidents forward to police and medical…etc.

Potential Advanced GIS features
- Graph search using Dijkstra’s algorithm to determine search area extent
- Geocoding coordinates to return meaningful street locations
- Voronoi diagrams to forward incidents to closest responder.
  - Dynamic refresh for police, do to patrols
  - Static for fire department, EMRs

Project Features
In this section, we will discuss features that could be implemented to provide additional functionality. It is unknown if these features are implemented in current enhanced 911 applications, but it would make sense to include them.

**Voronoi Polygon processing for optimal response time**
This feature would maintain a Voronoi polygon set which is based on current location of emergency responders out in the field. The application would maintain 3 sets of polygon layers, which correspond to Police, Fire, and Emergency Medical Teams. Since Fire and EMT locations typically are static until tasked to respond to an appropriate call, the Voronoi diagrams for these two layers would be static and would only require updating upon the addition or deletion of a fire station or hospital.

The Voronoi polygon set for the Police is far more interesting because Police units are typically deployed and patrolling roads within their jurisdiction. This polygon set would need to be continuously updated to guarantee querying the set for the police unit which can respond the soonest to an incoming emergency call. To maintain this set, we are
faced with the problem of quickly generating the polygon set from moving objects. This would require a fast operating algorithm which constructs Voronoi polygons. One such algorithm would be Fortune's plane sweep algorithm, which runs in $O(n \log n)$ time.

This algorithm runs a sweep line either vertically or horizontally across the space to be considered. It is assumed that the points used to build the Voronoi polygons are in a sorted structure. After a sweep line has passed a point, the trailing beach line begins a parabola from the point to create a parabola around the point. When the parabolas from three separate points intersect, this intersection forms one vertex of a polygon within the set. This sweep continues until the line reaches the end of the area to be checked.

1. Fortune's sweep algorithm, just prior to intersection of 3 parabolas
2. Fortune's sweep algorithm, just after the intersections of three parabolas.

3. Voronoi polygons following Fortune's sweep algorithm.
The Fortune algorithm from its Wikipedia entry
(http://en.wikipedia.org/wiki/Fortune’s_algorithm)

let \( \ast(z) \) be the transformation \( \ast(z) = (z_x, z_y + d(z)) \), where \( d(z) \) is a parabola with minimum at \( z \)

let \( T \) be the "beach line"

let \( R_p \) be the region covered by site \( p \).

let \( C_{pq} \) be the boundary ray between sites \( p \) and \( q \).

let \( p_1, p_2, \ldots, p_m \) be the sites with minimal \( y \)-coordinate, ordered by \( x \)-coordinate

\[ Q' \leftarrow S - p_1; p_2, \ldots, p_m \]

create initial vertical boundary rays \( C_{p_1,p_2}^0, C_{p_2,p_3}^0, \ldots, C_{p_{m-1},p_m}^0 \)

\[ T \leftarrow \ast(R_{p_1}), C_{p_1,p_2}^0, \ast(R_{p_2}), C_{p_2,p_3}^0, \ldots, \ast(R_{p_{m-1}}), C_{p_{m-1},p_m}^0, \ast(R_{p_m}) \]

while not IsEmpty(\( Q' \)) do

\[ p \leftarrow \text{DeleteMin}(Q') \]

**case** \( p \) of

\( p \) is a site in \( \ast(V) \):

- find the occurrence of a region \( \ast(R_q) \) in \( T \) containing \( p \), bracketed by \( C_{rq}^- \) on the left and \( C_{qs}^+ \) on the right
- create new boundary rays \( C_{pq}^- \) and \( C_{pq}^+ \) with bases \( p \)
- replace \( \ast(R_q) \) with \( \ast(R_q), C_{pq}, \ast(R_p), C_{pq}^+, \ast(R_q) \) in \( T \)
- delete from \( Q' \) any intersection between \( C_{rq}^- \) and \( C_{qs}^+ \)
- insert into \( Q' \) any intersection between \( C_{rq}^- \) and \( C_{pq}^+ \)
- insert into \( Q' \) any intersection between \( C_{pq}^- \) and \( C_{qs}^+ \)

\( p \) is a Voronoi vertex in \( \ast(V) \):

- let \( P \) be the intersection of \( C_{qr}^- \) on the left and \( C_{rs}^+ \) on the right
- let \( C_{uq} \) be the left neighbor of \( C_{qr}^- \) and
- let \( C_{sv} \) be the right neighbor of \( C_{rs}^+ \) in \( T \)
- create a new boundary ray \( C_{qs}^0 \) if \( q_y = s_y \),
- or create \( C_{qs}^+ \) if \( p \) is right of the higher of \( q \) and \( s \),
- otherwise create \( C_{qs}^- \)
- replace \( C_{qr}, \ast(R_r), C_{rs} \) with newly created \( C_{qs} \) in \( T \)
- delete from \( Q' \) any intersection between \( C_{uq} \) and \( C_{qr}^- \)
- delete from \( Q' \) any intersection between \( C_{rs}^+ \) and \( C_{sv}^- \)
- insert into \( Q' \) any intersection between \( C_{uq} \) and \( C_{qs}^+ \)
- insert into \( Q' \) any intersection between \( C_{qs}^+ \) and \( C_{sv}^- \)
record \( \mathbf{p} \) as the summit of \( C_{qr} \) and \( C_{rs} \) and the base of \( C_{qs} \)
output the boundary segments \( C_{qr} \) and \( C_{rs} \)
endcase
endwhile
output the remaining boundary rays in \( T \)

Other sites with information about Fortune's algorithm

  - Contains implementations in ActionScript 3, JavaScript, and C++
  - Implementation of Fortune's algorithm in C#
- [http://www.skynet.ie/~sos/mapviewer/docs/Voronoi_Diagram_Notes_1.pdf](http://www.skynet.ie/~sos/mapviewer/docs/Voronoi_Diagram_Notes_1.pdf)
  - More reading on Fortune's algorithm
Dijkstra's Algorithm for finding all paths for a given time interval

Dijkstra's Algorithm was primarily designed to find the shortest path between two points. However, it can be modified to allow the search to run until all paths are discovered for a given time interval. It works by specifying a start position and a given time interval for travel time. By operating on a road network represented as a weighted graph where each graph segment uses the following weight: road speeds/segment distance = time, we can determine how far a vehicle can travel in any direction given the time interval.

When Dijkstra's algorithm finishes its search, it will have a list of nodes stored in memory which will have a time value assigned. This time value represents the lowest total amount of time required to reach each node from the start point, which is important to note given there might be multiple paths to each node depending on the direction of the search. When providing visualization of the search results, there are several options to consider, but most visualizations involve a green to yellow to red color scheme. Green represents nodes easily reached given the time frame and red represents nodes that are the furthest nodes reachable given the time constraint. This color scheme can be applied to point features representing the nodes, or using contour lines to represent time intervals.

4. Travel time search represented with point features
5. Travel time search represented with contour lines.
Dijkstra's Search algorithm in psuedocode from

```
function Dijkstra(Graph, source):
    for each vertex v in Graph:
        dist[v] := infinity ;
        previous[v] := undefined ;
    end for
    dist[source] := 0 ;
    Q := the set of all nodes in Graph ;
    while Q is not empty:
        u := vertex in Q with smallest distance in dist[] ;
        remove u from Q ;
        if dist[u] = infinity:
            break ;
        end if
        for each neighbor v of u:
            alt := dist[u] + dist_between(u, v) ;
            if alt < dist[v]:
                dist[v] := alt ;
                previous[v] := u ;
                decrease-key v in Q;
            end if
        end for
    end while
    return dist;
```

Project Sources

Tacit Information

System for Mobile GPS positioning and tracking:
Most likely an app that already exists and that can be modified.
Available across a variety of platforms. (iPhone, Android, etc.)
Private, secure monitoring system with specific subscribed users.
Interface for displaying spatial data and positions:
Maximize the map area within the interface.
Available across a variety of platforms. Will probably have to use javascript for compatibility.
Employ some attribute data access.

**Storage and software:**
We will likely need some server storage to record data from the participants and push data out to users.
A database system may be needed for archiving location records for future retrieval.
Use of ArcMap or another GIS for processing functions if needed.
Geoprocessing functions will need to be scripted using some time of programming language.
(python?)

**Codified Information**

**CrimeReports:** [https://www.crimereports.com/](https://www.crimereports.com/)
CrimeReports claims to be the largest and most comprehensive crime mapping network in the world. It is a web mapping interface that employs the Google Maps API and maps a variety of crime types across the United States, Canada, and the United Kingdom. CrimeReports is a near real-time source of official crime data from a large number of law enforcement agencies. There is also a free downloadable mobile version of CrimeReports for the iPhone, but none for the Android user market. The user can also sign up for crime alerts and view crime data in different ways. A temporal aspect has been added to the mapping interface to allow the user to view different types of crimes over time. However, analysis tools are not included in the interface to analyze the crime data and its spatial distributions. Users are also able to report crime incidences within the mapping interface with the click of a button. The intermingling of official crime data and crowd-sourced crime data does present the possibility for misuse and abuse of their service.

**Real Time Crime Center:**
The New York City Police Foundation has developed a Real Time Crime Center. This innovative center was developed to incorporate advanced technological concepts and methods to fight crime and assist the New York Police Department in the Twenty-First Century. It was the first Real Time Crime Center in the United States and is equipped with the hardware and software to not only store, retrieve, and analyze billions of records but also provide up to date real-time information for officers and detectives when crimes take place. Using satellite imagery and GPS, the center is able to map and record the NYPD’s resources throughout the city of New York. The Real Time Crime Center has also developed a 911 call location mapping service which records the location of the emergency calls within the city to analyze how personnel and resources may be used to efficiently deal with each situation. This comprehensive system has been used to solve a variety of crimes within New York City.

CrimeStat is a free stand-alone statistics package that is used to analyze crime data distributions and can be implemented into most GIS packages. The advantage of CrimeStat is that it performs a multitude of spatial analysis functions for crime data such as distribution or centrographic measures, spatial autocorrelation, hot spot detection and analysis, interpolation, space-time analysis, journey to crime analysis, and demand modeling. CrimeStat is not only a useful for law enforcement; it can be applied to traffic analysis, urban planning, sociology, and even forestry. In the future (post 2009), CrimeStat will have a GUI interface and also be consistent with .NET framework. Additional spatial analysis tools will also be added to CrimeStat for enhanced crime data analysis techniques.

Geographic Dynamics, Visualization and Modeling, M. Yuan, page 358-362

Geographic Dynamics is a highly complex concept that is multiscalar and multidimensional with a variety of processes that operate at multiple spatiotemporal scales in multiple thematic domains. With so many processes acting in geographic dynamics, modeling such integrated processes becomes a very daunting task. However, geographic dynamics can be a powerful way to gain useful insights into geographic processes. Animated cartography, geovisualization and geocomputational models have all been used to visualize and display geographic phenomena and processes that happen on temporal scales. A large amount of varying algorithms, models, and relationships contribute to each attempt to express geographic dynamics however, the applications of geographic dynamics are extremely useful. Many of these applications are present in our everyday lives whether we realize it or not. Weather and climate modeling is a great example of geographic dynamics that is used around the world every day. Other applications such as land cover change, human activity (ex. Crime events), and other biological, geological, and tectonic models are key for the monitoring and interpretation of the world around us.
Web

Outside (under tree canopy):

www.tc.umn.edu/~kell1026/o3_gps.html

CSci 5980: From GPS & Google Earth to Spatial Computing

Current position: lat=44.948089, long=-93.3840727 (accuracy 26m)
last reading taken at: 21:31:54

In a moving car:

www.tc.umn.edu/~kell1026/o3_gps.html

CSci 5980: From GPS & Google Earth to Spatial Computing

Accuracy not sufficient (1302m vs 150m) - last reading taken at: 07:25:04
Current position: lat=44.94839031666667, long=-93.3828828333333 (accuracy 24m)
last reading taken at: 07:25:04
Current position: lat=44.9484114666666, long=-93.3356167666666 (accuracy 6m)
last reading taken at: 07:25:04
Current position: lat=44.9484471666666, long=-93.3385042393333 (accuracy 6m)
last reading taken at: 07:25:04
Current position: lat=44.9484671666666, long=-93.3360018 (accuracy 4m)
last reading taken at: 07:25:07
Current position: lat=44.948480063, long=-93.336209006 (accuracy 4m)
last reading taken at: 07:25:06
Current position: lat=44.9484812833333, long=-93.3369073833333 (accuracy 4m)
last reading taken at: 07:25:00
Current position: lat=44.94847691, long=-93.3365770833333 (accuracy 4m)

Inside a building:
CSci 5980: From GPS & Google Earth to Spatial Computing

Current position: lat=44.9706563, long=-93.2841302 (accuracy 34m)
Last reading taken at: 11:59:37
Stop