CPSC 448A: Directed Studies in Computer Science: Graph Theory, Graph Drawing, and Graph Algorithms

Ziyang Jin

Department of Computer Science
The University of British Columbia

September 2018 — December 2018

1 Information


```
Student    :  Ziyang Jin (f4a0b@ugrad.cs.ubc.ca)
Supervisor :  William S. Evans (will@cs.ubc.ca)
```

```
Time       :  2018 - 2019 Winter Term 1 [Sep - Dec]
Topic      :  Graph Theory, Graph Drawing, and Graph Algorithms
Github     :  https://github.com/ziyaj/directed_studies
```

The studies are mainly about graph theory, graph drawing, and graph algorithms. We have also included some part in randomized algorithms and computational geometry. Every week, I meet Will and discuss what I learned for the past week. Will will ask me questions to test if I really understand the algorithms or concepts in the topics. Will will also speaks about interesting facts and theorems he likes.

2 Topics

1. Basic definitions in graph theory
2. Ramsey's theory and Ramsey number
3. Planar graphs
    - Definition and characterization of planar graphs
    - The left-right planarity test
    - Algorithms in planar graphs
4. Graph drawing
    - Draw a planar graph on a grid
    - Schnyder's algorithm
    - Tutte's algorithm
    - Force-directed Methods
5. Graph Algorithms
    - Karger's algorithm
    - Dijkstra's algorithm using Fibonacci heaps
    - Kosaraju's algorithm
6. Spectral graph theory
    - Eigenvalues and optimization
    - Drawing graph using eigenvalues and eigenvectors
    - Detecting graph isomorphism via eigenvalues
7. Computational Geometry
    - Voronoi diagrams
    - Fortune's algorithm
    - Delaunay triangulations
    - Convex Hull
    - Chan's algorithm and output sensitivity
    - Algebraic Decision Tree
8. Markov chain and Random walk

3 Reading Materials

Papers

1. *The Left-Right Planarity Test* by Ulrik Brandes
2. *How to Draw a Planar Graph on a Grid* by Hubert de Fraysseix, János Pach, and Richard Pollack
3. *Embedding Planar Graphs on the Grid* by Walter Schnyder
4. *How to Draw a Graph* by William T. Tutte
5. *Maximum Flows and Parametric Shortest Paths in Planar Graphs* by Jeff Erickson

Textbooks

1. *Algorithm Design* by Jon Kleinberg and Éva Tardos
2. *Introduction to Algorithms* by Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein
3. *Graph Theory* by John A. Bondy and U. S. R. Murty
4. *Graph Drawing - Algorithms for the Visualization of Graphs* by Giuseppe Di Battista, Peter Eades, Roberto Tamassia, and Ioannis G. Tollis
5. *Randomized Algorithms* by Rajeev Motwani and Prabhakar Raghavan

Lecture notes

1. *Graph Drawing* lecture notes by William S. Evans, The University of British Columbia.
   `https://www.cs.ubc.ca/~will/536E/`
2. *Spectral Graph Theory* lecture notes by Daniel A. Spielman, Yale University.
   `http://www.cs.yale.edu/homes/spielman/561/syllabus.html`
3. *Algorithms for Planar Graphs and Beyond* lecture notes by Erik D. Demaine, Massachusetts Institute of Technology.
   `http://courses.csail.mit.edu/6.889/fall11/lectures/`
4. *Computational Geometry* lecture notes by David M. Mount, University of Maryland, College Park.
   `http://www.cs.umd.edu/class/fall2016/cmsc754/lectures.shtml`
5. *Randomized Algorithms and Probabilistic Analysis* lecture notes by Anna R. Karlin, University of Washington.
   `https://courses.cs.washington.edu/courses/cse525/13sp/`

4 Meeting Notes

Week 00 - The First Meeting

`* Meeting:  2018-09-11, Tue, 11:30 - 12:40, in Will's office`

What we did

We started to figure out the topics for the direct study and search out for materials like websites and textbooks related to these topics. We also discussed Ramsey number and proofs for $R(3,3) = 6$. Unfortunately the argument seems cannot be applied to bigger Ramsey numbers. We also discussed some examples of random walk.

What I learned from discussion

- Matrix representation of graph is quite useful. It is normally used in computing paths. Also, one question we want to ask about a graph is how connected the graph is. That has something to do with the eigenvalues of the matrix representation of graph. Google's page rank is an example of applying the use of eigenvalues in graph representation. Also, matrix representation of graphs can also be used in graph drawing.
- A clique in graph is a complete graph with n nodes.
- Ramsey number $R(x, y)$ is defined as the smallest complete graph s.t. for all choices of red/blue colorings, there (unavoidably) exists a red x-clique or a blue y-clique. $R(3,3) = 6$, and $R(4,4) = 18$, and $R(5,5)$ has not be confirmed yet, but it is ≥ 43 and ≤ 48. Based on the computing power human beings currently have, brute force will not for finding Ramsey numbers.

To-do before next meeting

- Borrow "Bondy and Murty" and "Di Battista et al." from UBC CS reading room.
- Read "Bondy and Murty" about graph theory and planar graphs.
- Find 4 things I am interested in graph drawing, from "Di Battista et al."

Week 01

```
* Meeting:  2018-09-18, Tue, 11:30 - 12:40, in Will's office
```

What we did

We proved that $K_{3,3}$ is non-planar using the Jordan curve theorem. I explained the idea of stereographic projection, and its interesting corollaries. I explained the idea of dual, and we also did a proof that the dual of a plane graph is also a planar graph by countering each other's arguments. In the end, as a practice, we found a $K_{3,3}$ subdivision from the Petersen graph.

What I learned from discussion

- A nonseparable graph is a graph that has no cutvertex.
- When asked why in a proof, never say why not. Proof techniques of making the statement more and more precise by looking at more and more complicated examples.
- The circular ordering of edges at a vertex (clockwise and counter clockwise).

To-do before next meeting

- Read "The Left-Right Planarity Test" by Ulrik Brandes.
- Download "P.I.G.A.L.E", play with it, and read the algorithm implementations.
- Find papers to read about how to test whether a graph is planar and how to draw a planar graph.

Week 02

`* Meeting:  2018-09-25, Tue, 11:30 - 13:00, in Will's office`

What we did

I explained what a LR partition is and how the left-right planarity test works in general. Then Will asked me how to do the LR partition in linear time. It is equivalent to a 2SAT problem: for each pair of back edges, it is either the same-side or different-side. Then Will showed me the implication graph of 2SAT, and explained to me the concept of strongly connected components. I was unable to come up with a linear-time algorithm quickly. Then we discussed how to solve 2SAT in linear time using Kosaraju's algorithm. In the end, we had a brief discussion in zero-knowledge proofs.

What I learned from discussion

- Never give an exponential algorithm to solve 2SAT outside of Will's office. In general, do not get satisfied with a non-polynomial algorithm unless it can be proven NP-hard.
- In order to solve a 2SAT, we can use implications from graph. To decide whether the Boolean formula can be satisfied, we just need to find whether a variable a and its complement $\overline{a}$ are strongly connected in the equivalence graph. Kosaraju's algorithm performs 2 depth-first searches. The second search is on the graph G^T with reversed edges, and we follow the reverse order of the time leaving the vertex in the first depth-first search.
- Zero-knowledge proofs: person A can make person B believe that A knows something in a way such that B cannot reproduce what person A did to convince others that B knows the same thing. For example, Will knows a Hamiltonian cycle in a complicated graph. He convince me by letting me have a choice of showing the Hamiltonian cycle with the whole graph blocked, or the whole graph (in order to verify that it is the original graph - Will isn't cheating). By doing this, it make me have a $1-2^k$ (where k can be efficiently large) chance of believing that Will knows the Hamiltonian cycle, but I cannot reproduce it to show others. Zero-knowledge proofs were first conceived in 1985 by Shafi Goldwasser, Silvio Micali, and Charles Rackoff.

To-do before next meeting

- Read *How to draw a planar graph on a grid* by H. de Fraysseix, J. Pach, R. Pollack, 1990
- Read *Embedding planar graphs on the grid* by Walter Schnyder, 1990
- Read *How to draw a graph* by W.T. Tutte, 1963

Week 03

`* Meeting:  2018-10-02, Tue, 11:30 - 12:20, in Will's office`

What we did

We discussed the the paper by de Fraysseix, Pach, and Pollach, and drew how the algorithm works on board for cases with 3, 4, 5 vertices. Then we discussed the paper by Schnyder, and I explained how the normal labeling of a maximal planar graph works and how to derive a realizer, and how we can build the barycentric coordinates of the graph given its realizer.

What I learned from discussion

- LEMMA 2.1. Let $v \in V(G) \to (v_1, v_2, v_3)$ be a barycentric representation of a graph G. Then given any three noncolinaer points α, β, γ. The mapping $f : v \in G \to v_1\alpha + v_2\beta + v_3\gamma$ is a straight line embedding of G in the plane spanned by α, β, γ. The proof is a slick.
- When reading an algorithm on graph drawing, it is better to simulate it by drawing it with a hand to see how it really works. Otherwise, I may miss some important details in the algorithm.
- What is interesting in the paper by de Fraysseix, Pach, Pollack is that how they establish the normal ordering of vertices, and use that to gradually build the graph step by step.
- What is interesting in the paper by Schnyder is that how he came up with the labeling of the graph, which is from an implication of something else Schnyder is researching. Will liked the paper so much when he was a graduate student that he put the paper in the bag until it was broken.

To-do before next meeting

- Read *How to draw a graph* by W.T. Tutte, 1963
- Explore materials on MIT 6.889 course website:
  `http://courses.csail.mit.edu/6.889/fall11/`

Week 04

```
* Meeting:  2018-10-09, Tue, 11:30 - 12:30, in Will's office
```

What we did

We discussed the the paper by W.T.Tutte. I drew an example graph and explained how to establish the linear system. The coordinates of an "inner" vertex is the average of the coordinates of all its neighbours. Then we discussed how can it make sure that by performing Tutte's algorithm there is no edge crossing. Tutte made a complicated argument on proving the correctness of his algorithm, i.e., ensure no edges crossing. In the end, we looked at the website of MIT 6.889 to figure out what to do next week.

What I learned from discussion

- There are two major things Tutte is trying to prove. First, by averaging the coordinates of vertices, there cannot be a concave inner face; every face has to be convex. Second, if we draw a line between shared edges of two faces, it must be true that one of the face is on one side of the line and the other face has to be on the other side of the line.
- Why we cannot write the paper more intuitively? Because your intuition might be different from others. So other people may not understand your intuition. Therefore, you have to establish your theorem by formal proofs so that everybody can understand.
- A $K_{3,3}$ can be drawn in different ways. Some embedding is not easy to be recognized as $K_{3,3}$. If the polygon graph is even-cycle, we can recognize a bipartite graph from it.

To-do before next meeting

- Read chapter for randomized algorithms in *Algorithm Design* by Jon Kleinberg and Éva Tardos.

Week 05

`* Meeting: 2018-10-16, Tue, 11:30 - 12:25, in Will's office`

What we did

I explained what I studied for randomized algorithms, and showed some basic results on randomized algorithms. I described a simple version of Karger's Algorithm (Contraction Algorithm), which runs in $\mathcal{O}(n^4)$ to get a constant bound $(1/e)$ in failing to find the global minimum cut. Then Will searched for a more efficient version (the Karger-Stein algorithm), which runs in $\mathcal{O}(n^2 \log n)$. It uses recursion on smaller subsets. Then we showed another randomized algorithm for Max 3-SAT, which if we run $8k$ times, where k is the number of clauses.

What I learned from discussion

- Uniform circuit set is another computational model other than Turing machine. There is a Turing machine that given some problem it can output the uniform circuit set to solve the problem.
- Empirical algorithms are algorithms that are valued by their practice performance, like SAT-solvers. SAT is NP-Complete, but many SAT solvers have good performance on most cases.
- The intuition is that when the graph is huge, it is usually non-planar.
- It is not the case that most real world problems are planar problems. However, if it is known to be planar, take advantage of the planarity is good.
- No NP-hard problem has a probabilistic algorithm that has a success probability $\geq 1/2 + \epsilon$ for a positive constant ϵ. Such algorithm would imply NP is contained in BPP, which is not believed to be true.

To-do before next meeting

- Read chapter 19: Fibonacci Heaps in *Introduction to Algorithms* by Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein.
- Read *Maximum Flows and Parametric Shortest Paths in Planar Graphs* by Jeff Erickson.

Week 06

`* Meeting: 2018-10-23, Tue, 11:32 - 12:33, in Will's office`

What we did

I explained what I studied in Fibonacci heaps, and we looked at other data structure that uses disjoint heaps. We looked at binary heaps, binomial heaps, and strict Fibonacci heaps, and we compared their performance on different operations. In particular, strict Fibonacci heaps get us interested because the performance of strict Fibonacci heaps is $\Theta(1)$ (not amortized!) except for the *delete-min* operation.[1] Then we discussed on Jeff Erickson's paper and his $\mathcal{O}(n \log n)$ algorithm on maximum flow in planar graphs. In the end, Will showed me his implementations of Tutte's algorithm and (incomplete) Forced-directed drawing algorithms in Icon.

What I learned from discussion

- The intuition of amortized analysis is to balance the work of each operation and the change in data structure. The change in data structure can potentially "save" some work in other operations, which leads to the idea of potential function.
- It is incredibly hard to find the potential function to do amortized analysis. You have to understand the change in data structure very well in order to come up with the correct potential function.
- Icon is a high-level, general-purpose programming language with novel features including string scanning and goal-directed evaluation.[2]
- Other than graph theory and graph drawing, Will used to do researches in complexity theory, compression, and programming languages.

To-do before next meeting

- Read chapter 10: Force-Directed Methods in *Graph Drawing: Algorithms for the visualization of graphs* by Giuseppe Di Battista, Peter Eades, Roberto Tamassia, and Ioannis G. Tollis.

[1] *Strict Fibonacci Heaps* by Gerth S. Brodal, George Lagogiannis, Robert E. Tarjan in STOC 2012 `http://tildeweb.au.dk/au121/papers/stoc12.pdf`

[2] The Icon Programming Language: `https://www2.cs.arizona.edu/icon/index.htm`

Week 07

`* Meeting:  2018-10-30, Tue, 11:30 - 12:45, in Will's office`

What we did

We discussed about force-directed layouts in graph drawing, including string algorithms, Tutte's algorithm, Kamada and Kawai's algorithm, and Davidson and Harel's algorithm (general energy functions). Will showed me a webpage `http://www.yasiv.com/graphs` that draws amazing graphs using iterative methods. In the end, I happened to meet Prof. David Kirkpatrick[3]. I am glad that in his 70s, David is still healthy, energetic, and he keeps working on theoretical computer science papers.

What I learned from discussion

- If we only nail down 2 vertices when doing Tutte's algorithm, the result will be all the vertices lie in the line segment formed by the 2 vertices.
- In Kamada and Kawai's algorithm on graph theoretic distances, we choose the unit distance be the longest graph theoretic distance between any 2 vertecies in the graph.
- "Equations describing the minimal energy states are stiff for some graphs of low connectivity". Here 'stiff' means the parameters are fixed, not likely to change.
- Genetic algorithms are algorithms that simulates the nature of gene. Let's say we have solutions A and B and we want to combine them in a way that it gets the benefit of A and B. The difficult part is how you replicate the 'gene' and how to combine them. Genetic algorithms are usually heuristic, so you need to see a lot of evidence to see how it really works.
- "Resolution" can be defined as the smallest length of the edges in the drawing over the maximum length of the edges. Tutte's algorithm has a exponentially decreasing resolution, which is bad.
- To minimize the number of edge crossings in the general energy function, we can try to move the vertex to one side, such that the total number of crossing for that vertex is smaller.

To-do before next meeting

- Read Daniel A. Spielman's lecture notes on *Spectral Graph Theory*. Websites: `http://www.cs.yale.edu/homes/spielman/561/syllabus.html` and `http://www.cs.yale.edu/homes/spielman/561/561schedule.html`

[3]David Galer Kirkpatrick is a Professor Emeritus of computer science at UBC. He is known for the Kirkpatrick–Seidel algorithm and his work on polygon triangulation, etc. `https://www.cs.ubc.ca/people/david-kirkpatrick`

Week 08

`* Meeting:  2018-11-06, Tue, 11:25 - 12:45, in Will's office`

What we did

We discussed topics on spectral graph theory, particularly lecture 1 and lecture 2 of Spielman's notes. We went through drawing with Laplacian eigenvalues, and isoperimetry number. We carefully investigated the proof on the lower bound of θ_G, the isoperimetric number to understand its proof techniques and intuitions behind the steps. Then we looked at a result of the path graph, where the eigenvectors of small to big eigenvalues have increasing number of waves. We found some analogy in physics and mathematics (i.e. Taylor expansion), but did not totally find the proof. Then we talked about using λ_2 to draw graphs (it does not always work). In the end, I explained the algorithm for detecting graph isomorphism.

What I learned from discussion

- The diffusion operator is about the probability distribution of a random walk. Applying the diffusion operator will re-distribute the probability at each node.
- Delaunay triangulations are dual of Voronoi diagram. In practice, people draw Delaunay triangulations by building cones in 3D and project it on a plane.
- There is a bacteria analogy of the Delaunay triangulation. The points in a triangle is closer to its center node than in any other nodes. This is a like we put some bacteria at different points in a plane and let it grow, in the end they will intersect and form the lines.
- There is a linear time algorithm for polygon triangulation but probably no one has ever implemented it. In practice, people use triangulation algorithm that runs in $O(n \log n)$ or so.

To-do before next meeting

- Read David M. Mount's lectures notes on *Computational Geometry*, particularly "Lecture 11: Voronoi Diagrams and Fortune's Algorithm", "Lecture 12: Delaunay Triangulations: General Properties", and "Lecture 13: Delaunay Triangulations: Incremental Construction".

  `http://www.cs.umd.edu/class/fall2016/cmsc754/lectures.shtml`

Week 09

`* Meeting: 2018-11-13, Tue, 11:25 - 12:45, in Will's office`

What we did

We discussed Voronoi diagrams and Delaunay triangulation. We analyzed the intuition of Fortune's line sweeping algorithm comes from looking at a plane intersecting cones in 3D. We found a counter example of 4 vertices that disapproves the Delaunay triangulation minimizes the total edge length. We briefly talked about the idea of backwards analysis, which is common in analysing computational geometry algorithms.

What I learned from discussion

- Ipe is a drawing editor for creating figures in PDF or (encapsulated) Postscript format. It supports making small figures for inclusion into LaTeX-documents as well as making multi-page PDF presentations that can be shown on-line with Acrobat Reader.[4]
- We can do a reduction from the sorting problem to constructing Voronoi diagrams, which gives us a bound that constructing Voronoi diagrams takes $\Omega(n \log n)$. The construction put the points on a parabola, with each point corresponding to a number x, and has position (x, x^2). The solution to Voronoi diagram gives us an ordering of the points, which gives us a sorting of the numbers.
- The *Kirkpatrick–Seidel algorithm*[5] called by its authors "the ultimate planar convex hull algorithm", is an algorithm for computing the convex hull of a set of points in the plane, with $O(n \log k)$ time complexity, where k is the number of points in the hull. Thus, the algorithm is output-sensitive: its running time depends on both the input size and the output size.
- *Chan's algorithm*[6] simplified the Kirkpatrick-Seidel algorithm and generalized it to 3 dimensions. Timothy Chan is a great and brilliant man.

To-do before next meeting

- Read David M. Mount's lectures notes on *Computational Geometry*, particularly "Lecture 3: Convex Hull's in the plane" and "Lecture 4: Convex Hulls: Lower Bounds and Output Sensitivity".

  `http://www.cs.umd.edu/class/fall2016/cmsc754/lectures.shtml`
- Figure out what to do for the last week of our studies.

[4] `http://ipe.otfried.org/`

[5] David G. Kirkpatrick and Raimund Seidel. *The Ultimate Planar Convex Hull Algorithm?*. SIAM Journal on Computing. 1983.

[6] Timothy M. Chan. *Optimal Output-Sensitive Convex Hull Algorithms in Two and Three Dimensions*. Discrete Comput Geom. 1996.

Week 10

`* Meeting:  2018-11-20, Tue, 11:26 - 12:30, in Will's office`

What we did

We discussed the basic algorithms for finding convex hulls, namely *Graham's scan* $O(n \log n)$, convex hull with divide and conquer $O(n \log n)$, *Jarvis's march* $O(nh)$, and *Chan's algorithm* $O(n \log h)$. We would like to understand how the determinant of a matrix is related to its corresponding graph's left-right orientation. We then looked at the runtime analysis of Chan's algorithm, with the insight that the time taken for all failing cases is asymptotically the same as the time needed for the last success case. we also discussed how to do binary search in Chan's algorithm of finding the tangent lines of mini-hulls. In the end, we discussed the proof of lower bound, any convex hull algorithm under the model of "algebraic decision tree" takes at least $\Omega(n \log h)$.

What I learned from discussion

- When we do reduction, we need to be careful that the two problems and the reduction steps should be under the same computational model. So the simple reduction from sorting to convex hull is problematic because the computational model for sorting is using only comparisons while the reduction needs to multiply numbers.
- Graham's scan plus convex hull with divide and conquer is the original ultimate algorithm by Kirkpatrick and Seidel, and Chan's first algorithm for convex hull. Graham's scan plus Jarvis's march is the second algorithm Chan found for convex hull, which he used to generalize it to 3 dimensional space.

To-do before next meeting

- Read Anna R. Karlin's notes on *Randomized Algorithms and Probabilistic Analysis*, particularly "Lecture 17 : Coupling, martingales".
- Read Chapter 7 of *Probability and Computing: Randomized Algorithms and Probabilistic Analysis* by Michael Mitzenmacher and Eli Upfal.
- Read Chapter 6 of *Randomized Algorithms* by Rajeev Motwani and Prabhakar Raghavan.

Week 11 - The Last Meeting

`* Meeting: 2018-11-27, Tue, 12:09 - 13:46, in Will's office`

What we did

We discussed the topic in Markov chain and random walks following the book by Motwani and Raghavan. We spent quite some on Exercise 6.1, to prove the expected number of steps in a simple random walk that begins at u ends upon first reaching v is $n-1$ and the expected number of steps to visit all the vertices in G starting from u is $(n-1)H_{n-1}$, where $H_{n-1} = \Sigma_{j=1}^{n-1} 1/j$ is the Harmonic number. We then proved that the random walk on K_n is exactly the same as coupon collection with $n-1$ coupons. After this, we spent time proving that the expected number of steps for the randomized $2-SAT$ algorithm to find a satisfying assignment is $O(n^2)$ by starting from a base case of looking at patterns of $h_{01} = 1, h_{02} = 4, h_{03} = 9, ...$ Will then explained to me what Markov's inequality is $Pr[X \geq aE(X)] \leq 1/a$ and did a simple proof from the definition of $E(X)$. Then I asked Will about periodicity, and Will explained to me that the basic idea of periodicity by giving examples of bipartite graph, then Will asked about tripartite graph, and I proved the random walk is *aperiodic*. In the end, we looked at one application of *coupling*. If we shuffle one deck of cards by picking one card out and put it in the bottom, how many cards do we expected to pick until we get to the stationary state.

What I learned from discussion

- Periodicity is that there is a set of probabilities on vertices in the random walk that are positive periodically (they are not always > 0). For example, in a bipartite graph, in each step, there is only one side of the vertices that have positive probabilities, and the other side of vertices all have 0 probability. In the next step, the situation switches.
- The *cover time* for a random walk in a complete graph of n vertices is $\Theta(n \log n)$. This is due to the result we proved in Exercise 6.1 that the expected number of steps to visit all the vertices is $(n-1)H_{n-1}$. The harmonic number can be taken integral to $\ln n$. So the total number of steps $\leq (n-1)\ln n = \Theta(n \log n)$
- *Coupling* can be applied to shuffling a deck of cards. You want to figure out how many single card picks you need to approach the stationary state, where each ordering of cards have equal probability. You have your messy deck of cards, and you also have a deck of sorted cards (this is like the stationary distribution). Then we pick one card from the sorted deck and put it on to the bottom and pick the same card on the messy deck. When the messy deck has exactly the same ordering of card as the transformed sorted deck, then we are done. This is similar to coupon collection, so the expected number of card picks is $\Theta(n \log n)$, so about $52 \cdot \ln 52 = 206$ times.