Graphs and Algorithms 2016

Teachers: Nikhil Bansal and Jesper Nederlof
TA: Shashwat Garg (Office Hours: Thursday: Pick??)

Webpage: www.win.tue.nl/~nikhil/courses/2WO08
(for up to date information, links to reading material)

Goal: Learn discrete mathematics
Beautiful ideas in Graph Theory
Connections to probability, algebra, …
Various Algorithmic Ideas
(polynomial time, approximation, exponential time, …)

Sometimes see results obtained in last 2-3 years!

Most of all: Learn how to think and solve problems

Administrative Stuff

Lectures + exercise sessions
(Friday schedule: ?? Preference)

Final 50%
Mid-term 30%
(can bring 1 sheet of paper with things written on it)

3-4 Homeworks 20% (teams of up to 2 students)

Homework: Fine to discuss with other students, but acknowledge their names (and write proof in your own words)

Homework solutions must be in Latex, and be clear to understand.

Basic Concepts

Graph: Vertices, Edges

Notions: Degree, connectedness, regular, path v.s. walk, cycle v.s. tour

Types of graphs:

Complete graphs

Trees

Bipartite graphs: Two vertex sets X,Y edges from X to Y.

Planar graphs

…

Basic concepts

- Independent set, Clique: max indep set $\alpha(G)$
  (S independent if no edge contained in S)
- Coloring number: $\chi(G)$
  (each color class is an independent set)

- Hamiltonian Cycle: Visits each vertex exactly once
- Eulerian Tour: Visits each edge exactly once

- Matching: Collection of edges, with no common vertex.

- Vertex Cover: Set of vertices S, so that each edge has at least one endpoint in S. min vertex cover v(G)

Basic Results

Q: If maximum degree is d, there is always an independent set of size >= n/(d+1). Is this bound tight?

Q: If max. degree = d, then (d+1)-colorable

Q: Tree has exactly n-1 edges

Q: G is bipartite if and only if it has no odd cycles

Q: Give efficient algorithm to detect bipartiteness?

Q: For any graph G, $v(G) + \alpha(G) = n$.
(S in an independent set iff V\S is a vertex cover)

Finding maximum independent sets

Q: Can you find max. independent set in G.

Of course in 2^n time. What about poly(n) time?

Tree: Yes. How?

Bipartite: Yes.

Planar graphs: NP-Hard, but can do in $2^{O(\sqrt{n})}$ time.

Will see later: For general graphs $2^{o(n)}$ time is unlikely.

Max independent set

How about approximate solution?

Alg has Approximation ratio c if c = max_G OPT(G)/Alg(G)
(i.e. within factor c for every instance)

n approximation is trivial

Thm (1997): A $n^{0.999\ldots}$ ($n^{1-\epsilon}$ for any $\epsilon > 0$) approximation would imply P=NP.
(will see a weaker result later)

Thm (80's): For planar graphs, can find a $1+\epsilon$ approximation for any $\epsilon > 0$ in $2^{O(\frac{1}{\epsilon})}$ poly(n) time.

Basic results

Q: Planar graph has <= 3n-6 edges

Q: Planar graph is 6-colorable

Q: Planar graph is 5-colorable (Hint: Try swapping a,c or b,d)

Thm (76): Every planar graph is 4-colorable

Thm (1920's Kuratowski): Graph is planar iff no K_5 or $K_{3,3}$ are "minors".

(Minor: Edge/vertex deletions + contractions)

Will see an efficient algorithm for planarity testing.

Eulerian graphs

Eulerian: Connected + a closed tour that visits each edge exactly once.

Q: A connected graph G is Eulerian if and only if each degree is even.

Hint: Induction; find cycle; patch up pieces

Later see a polynomial time algorithm for Chinese postman problem.

(related TSP problem is NP Hard)

Basic Algorithms

Shortest Paths: Dynamic Programming (Bellman-Ford)

Will see idea in exercise session

(Floyd-Warshall: also with negative costs, unless negative cycles)

Minimum Spanning Tree: Order edges by increasing cost. Pick greedily.

Kruskal's agorithm (wikipedia proof); Prim, Boruvka, …

Maximum Bipartite Matching: Will see soon

Philosophical Interlude: NP and co-NP

Yes/No versions (can optimize via binary search)

Shortest Path: Is there a path of length <= D?

Is there a perfect (bipartite) matching?

Is the graph planar?

If answer = NO, how can the algorithm know?

There must an underlying polynomial size “certificate”/ “witness”

Bipartite-Matching

Bipartite graph V=(X,Y)

X saturated matching: every vertex in X is matched

Perfect matching if both X,Y saturated

(note $|X|$ must be equal to $|Y|$)

Hall's theorem: X saturated matching if and only if

For all $S \subset X$, $|N(S)| \geq |S|$

Hall's theorem Proof

Hint: Induction

Proof: If $|N(S)| > |S|$ for each S, ok by induction (just match one vertex arbitrarily and apply induction)

Else some subset $|S|$ is tight, with $|N(S)| = |S|$.

Consider subsets T in $X \setminus S$.

Claim: $|(N(T) \cap (Y \setminus N(S))| >= |T|$ (why?)

Apply induction on graph $(X \setminus S, Y \setminus N(S))$

Applications of Hall's theorem

Show that a d-regular graph has a perfect matching.

Show that a d-regular graph can be decomposed into d edge disjoint perfect matchings.

Suppose we arbitrarily divide deck of cards in 13 pieces of 4 cards each.

Show that, one can always pick one card from each piece so that you have one of each A,2,…,10,J,Q,K

Halls theorem for X-saturated matchings

If the vertex sets X and Y are not equal, and say |X|<|Y|, Hall's theorem for X-saturated matching is the following:

An saturated matching exists iff for each subset S of X, |N(S)| >= |S|

[Prove that the previous proof also works here]

Yet another generalization.

Suppose |X| <= |Y|.

There is a matching of size |X|-t if and only if for each subset S of X, |N(S)| >= |S|-t

Hint: Can you modify the graph and reduce this to previous case?

How to algorithmically find max X-saturated matching

At each step try to increase size of matching.
(they are called augmentation steps)

Pick an unmatched vertex on left side, and try to find an “alternating path” ending at an unmatched vertex on right.

Why does such an alternating path exist?
Do you see where Hall’s condition comes in?

More general variants of matching that can be solved efficiently

Suppose you also have a non-negative cost on the edges, and a bound k on the required cardinality of the matching.

1) Min-cost (left) perfect matching

2) Min-cost matching of cardinality k.

3) Maximum cardinality matching in general graphs

4) Min-cost size k matching in general graphs

Max-cardinality non-bipartite matching

There is a clever algorithm (Edmond's algorithm)

Explore augmenting paths, but get in trouble if encounter odd-cycle. Shrink it and proceed

(need to show nothing goes wrong)

What is the certificate for no perfect matching?

Tutte's Theorem: G has a perfect matching iff

For every $S \subset V$, o(G\S) $\leq$ |S|

where o(G\S) is the number of components of G\S with an odd number of vertices.

Necessity is easy to see. Sufficiency is harder

Applications

Several in exercise sheet