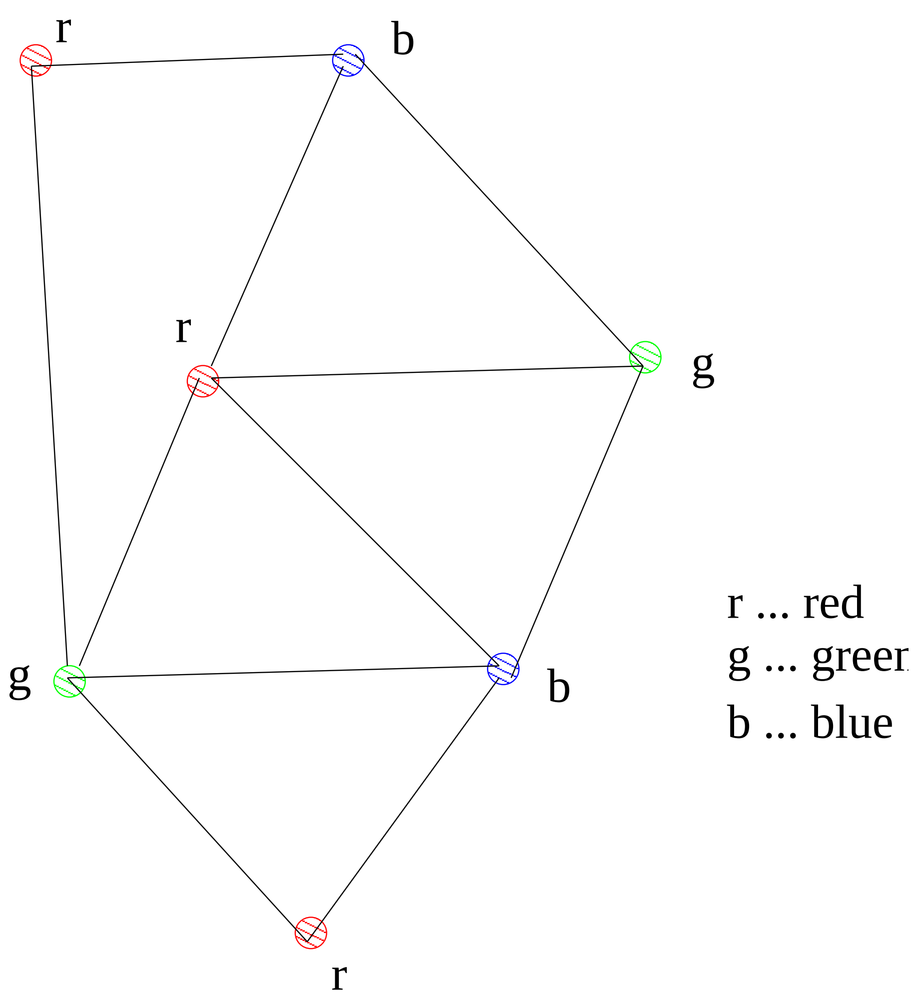
Graph Coloring Problem

Given a graph $G(V, E)$ and an integer k, is there a coloring of the graph with at most k colors so that no two vertices with the same color are adjacent?



The graph is 3−colorable,
cannot be colored with only 2 colors

Graph Coloring Problem is $\mathcal{NP}-complete$.

There are no good general approximation algorithms for the Graph Coloring.

For example, the best known approximation algorithm for 3-colorable graphs give a coloring that uses up to $n^{1/4}$ colors.

So there is no good approximation algorithm available.

From this point of view, a 2 approximation algorithm, or $\log n$ approximation algorithm look great.

$\mathcal{NP}-complete$ problems are not all the same!

The rest of the lecture is a review of the course.

Lecture 1

Measures of efficiency:
<u>time</u>
<u>space</u>

Analytical Methods:
analyze the structure of an algorithm and derive the time and space needed.

Empirical and analytical results.

Asymptotic notation

Worst case analyses:

Average case analyses (expected):

Best case analyses:

Lecture 2

Divide-and-conquer algorithms

Divide
the problem into subproblems.

Conquer
the subproblem by solving them recursively. (small size subproblems are solved directly).

Combine
the solution to the subproblems into a solution of the original problem

a) binary search
b) Merge-Sort
c) Quick-Sort, Randomized Quicksort

Lecture 3

Divide-and-conquer, continue.

Median and Order Statistics,

Making the partitioning $O(\log n)$ in the worst case.

Given a set of n points Q in the plane,
find a closest pair of points.

Lecture 4

Dynamic programming

1. Characterize the structure of an optimal solution.
2. Recursively define the value of an optimal solution.
3. Compute the value of an optimal solution in bottom-up fashion.
4. Construct an optimal solution from computed information.

(an optimal solution $\neq$ value of an optimal solution)

Assembly-line scheduling
Longest common subsequence
Optimal binary search trees:

Lecture 5.

5. Greedy algorithms

When a choice is to be made, it chooses what looks best at that moment.

Activity Selection Problem:

Scheduling Problem.

General Scheduling Problem.
NP-complete (last part of the course).

Knapsack problems

0-1 knapsack Problem:

Fractional Knapsack Problem

Huffman codes

Dijkstra Shortest Path

Lecture 6.

Minimum spanning tree

Kruskal algorithm,

Prim-Jarnik algorithm

Amortized Analysis

Aggregate Analysis

Accounting Method

Potential Method

Lecture 7.

Graph Algorithms

Breadth first search **BFS**

Depth first search **DFS**

Topological Sort

Strongly Connected Components

Lecture 8.

Bellman-Ford Algorithm

All-Pairs-Shortest-path

Ford-Fulkerson Flow Method:

augmenting paths
residual network,

Edmonds-Karp Algorithm = Ford-Fulkerson with BFS for augmenting paths

Maximum Bipartite Matching problem.

Lecture 9.

Linear Programming

a general method for optimizing a set of linear inequalities.

Simplex algorithm.

String matching

Rabin-Karp Algorithm

Knuth-Morris-Pratt Algorithm

Computational Geometry

Convex Hull: Graham Scan

Lecture 11 and 12.

Class $\mathcal{P}$ (or polynomial)

Class $\mathcal{NP}$ (or nondeterministic polynomial)

3-SAT Problem

The Clique Problem

The Vertex Cover Problem

The Hamiltonian-cycle Problem

The Traveling Salesman Problem

The Subset sum problem

The Knapsack problem

The Program Optimization:

polynomial-time reducibility of B **into** C.

Lecture 11

Some special cases of $\mathcal{NP}-$complete problems have polynomial algorithms.

Approximation algorithms for $\mathcal{NP}-$complete problems.

$\rho(n)$ approximation

Vertex cover: a polynomial 2-approximation algorithm.

Traveling Salesman with Triangular inequality,

Set Covering Problem: $\log n$-approx. algorithm

The Subset-Sum Problem: polynomial scheme.

Graph Coloring Problem