Assignment 1

Divide and Conquer Say a website maintains a database (data-structure) in which several person's ranking on a set of n movies (same set is considered for all) are stored already. Now, you have given your preference of ranking on the same set of n movies. The website try to match your preferences with the preferences of the others. Once the website has identified people with "almost similar" tastes based on a comparison of how you and they rate the movies, it can recommend that these are the others who have also liked the movies. Define a suitable measure for "almost similar" rankings. Design a brute force algorithm (which may run in $O(n^2)$) for the above problem for a single pair of rankings. Now improve your running time by designing the algorithm with the divide and conquer approach.

Hint: Refer *Algorithm Design* book by Jon Kleinberg and Eva Tardos (KT book) in divide and conquer chapter (Follow the term Collaborative filtering).

Assignment 2

Medians and Order Statistics Professor Olay is consulting for an oil company, which is planning a large pipeline running east to west through an oil field of n wells. The company wants to connect a spur pipeline from each well directly to the main pipeline along a shortest route (either north or south), as shown in figure. Given the x- and y- co-ordinates of the wells, how should the professor pick the optimal location of the main pipeline, which would be the one that minimizes the total length of the spurs? show how to determine the optimal location in linear time.

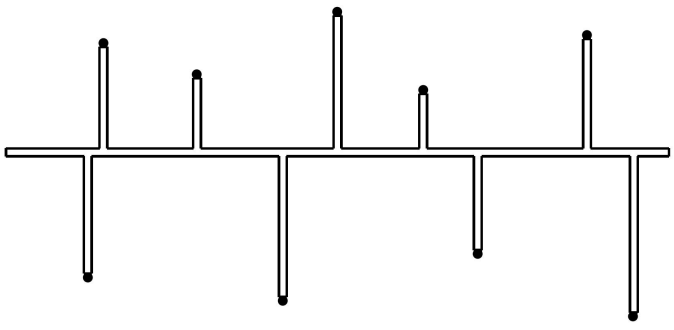


Figure 1: Professor Olay needs to determine the position of the east-west oil pipeline that minimizes the total length of the north-south spurs.

Hint: Refer *Introduction to Algorithms* book by T.H Cormen et. al (CLRS); Chapter: Medians and order statistics; Exercise 9.3-9; For answer you can refer Introduction to Algorithm solution manual of CLRS.

Assignment 3

Coin changing (Greedy Algorithm). Consider the Problem of making change for n cents using the fewest number of coins. Assume that each coin's value is an integer. Implement a greedy algorithm to make change consisting of quarters ($1), dimes ($5), nickels ($25), and pennies ($50).

Hint: Refer *CLRS* and *KT* book and Princeton lecture notes on Greedy Algorithms from KT book (*https://www.cs.princeton.edu/ wayne/kleinberg-tardos/ pearson/.*

Assignment 4

Interval Scheduling (Greedy Algorithm). Say you have n number of classes and each class has a duration with a start time s_i and a finish time f_i. You have only a single class room available. Design an algorithm to schedule maximum number of classes in that class room so that they do not overlap. Now say we have associated a value v_i to each class i.e if we schedule a class we can earn Rs. v_i. Design an algorithm to find a non-overlapping subset of classes of maximum value. Refer *KT* book (chapter number 1, and 6) and Princeton lecture notes on Greedy Algorithms from KT book (*https://www.cs.princeton.edu/ wayne/kleinberg-tardos/pearson/.*

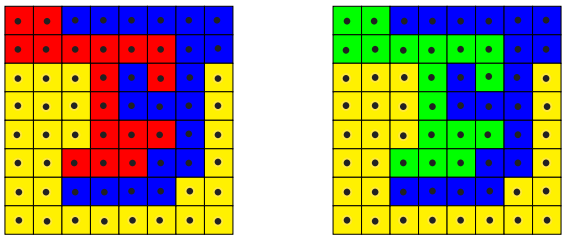
Mini-project (Assignment 5):

Say there are n students and n number of companies. Companies are coming to hire the students as interns and students have the liberty to give their preferences for all of the companies. Say their are three students (s_1, s_2, s_3) and three companies (A, B, C) and for example student s_1 will give his or her preference in some strict sense i.e. s_1 will convey that he prefers A over B over C. Likewise s_2 and s_3 will give their preferences. Now you have to design an algorithm so that each one will get one company. Mind it that if your algorithm is not good then there is a chance that the students can manipulate their true preferences and can get a better company. Say, for example, if your algorithm is: whenever s_1's turn will come, he will be given a job that was his last preference. In this case s_1 may be tempted to give his first preference to the last and can get his actual first preference during allocation. So during the designing of your algorithm, you have to design it in such a manner that every student will not hesitate to

provide his true preference list and also it will be efficient i.e. the students will get his first preference as much as possible.
Say in this case I have designed an algorithm like this: First I randomly order the students and one-by-one take one student in that order. Each student when I will take, I randomly chose a company from his remaining preference list and assign that company to him.
Your job will be to compare your algorithm with my algorithm in terms of the number of students getting his first preference. Plot a graph, by running the algorithms on 50 students-50 companies, 100 students-100 companies, 150 students-150 companies. In the X-axis you plot the number of stdents and in Y-axis you plot how many stdents are getting their first preferences based on your algorithm and my algorithm.
Refer: CS269I: Incentives in Computer Science (Lecture 1) by Tim Roughgarden.

Assignment 6

Say there is a grid of pixels. Some portions are coloured red. Say you are working in TuxPaint and you have to make the colour green. You can do that by just clicking on the region with green pallete selected. Design an algorithm to do that.
Your input will be a grid (say 11 X 11) as shown in the Figure 2 and randomly partition it into subgraphs. Each subgraph will represent different coloured regios. Now select a node randomly and identify the region where it belongs.



(a) Initial grid. (b) After recoloring.

Figure 2: Recoloring red region into green.

Hint: Think all pixels which are connected with different colours as different connected components of a graph and run BFS on the graph with the region selected from a node. Refer *https://www.cs.princeton.edu/ wayne/kleinberg-tardos/pdf/03Graphs.pdf* (Slides: 17-24)

Assignment 7

A Bipartite graph (Figure 3) is a graph whose vertices can be divided into two independent sets, U and V such that every edge $\langle u, v \rangle$ either u belongs to U and v belongs to V or u belongs to V and v belongs to U. We can also say that there is no edge that connects vertices of same set. A bipartite graph can be colored by using only *two* colors. Note that a graph having cycle of odd length (Figure 4b, 4c) can not be bipartite, hence can not be colored with only *two* colors. With the help of BFS algorithm, design an algorithm to verify whether a graph is bipartite or not.

Hint: Refer *https://www.cs.princeton.edu/ wayne/kleinberg-tardos/pdf/ 03Graphs.pdf* (Slides: 26-32)

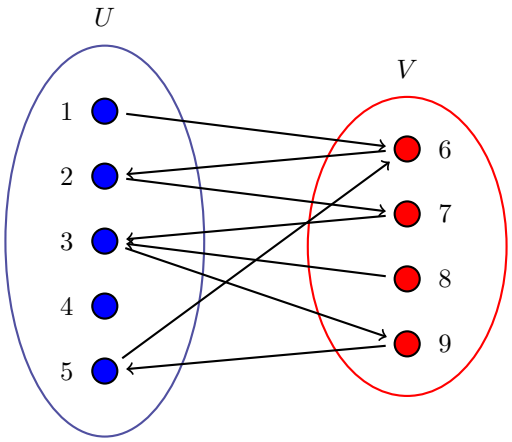
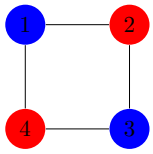
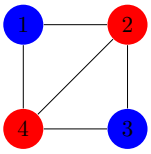


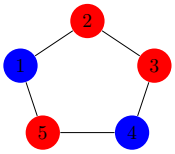
Figure 3: A bipartite graph colored with only two colors



(a) Even length cycle.



(b) Odd length cycle.



(c) Odd length cycle.

Figure 4: Different scenarioes of odd and even length cycles

Assignment 8

Application of Ford-Fulkerson Figure 5 contains a flow network $G = (V, E)$ for the Lucky Puck Company's trucking problem. The Vancouver factory is the source s, and the Winnipeg warehouse is the sink t. The company ships pucks through intermediate cities, but only $c(u, v)$ crates per day can go from city u to city v. Each edge is labeled with its capacity. We want to compute the maximum flow in the given flow network.

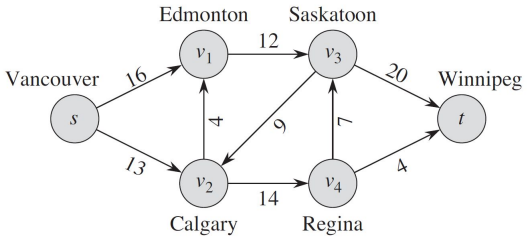
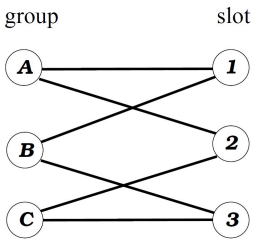


Figure 5: Flow network for Lucky Puck company

We can improve the bound by finding augmenting path with BFS algorithm. Then we call the $Ford - Fulkerson$ method so implemented the $Edmonds - Karp$ algorithm.

Assignment 9

Application of bipartite graph Say we wanted to be more sophisticated about assigning groups to homework presentation slots. We could ask each group to list the slots acceptable to them, and then write this as a bipartite graph by drawing an edge between a group and a slot if that slot is acceptable to that group. For example: This is an example of a bipartite graph: a graph



with two sides L and R such that all edges go between L and R. A matching is a set of edges with no endpoints in common. What we want here in assigning

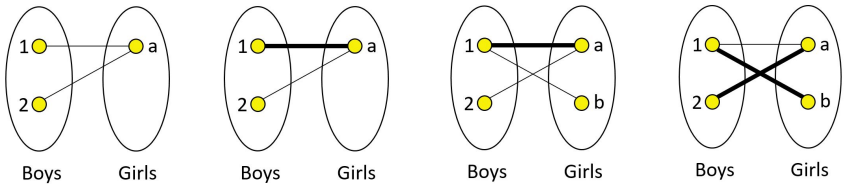
groups to time slots is a perfect matching: a matching that connects every point in L with a point in R. For example, what is a perfect matching in the bipartite graph above?

More generally (say there is no perfect matching) we want a maximum matching: a matching with the maximum possible number of edges. We can solve this using *Bipartite Matching*.

- Set up a fake start node s connected to all vertices in L. Connect all vertices in R to a fake sink node T. Orient all edges left-to-right and give each a capacity of 1.
- Find a max flow from s to t using Ford-Fulkerson.
- Output the edges between L and R containing nonzero flow as the desired matching.

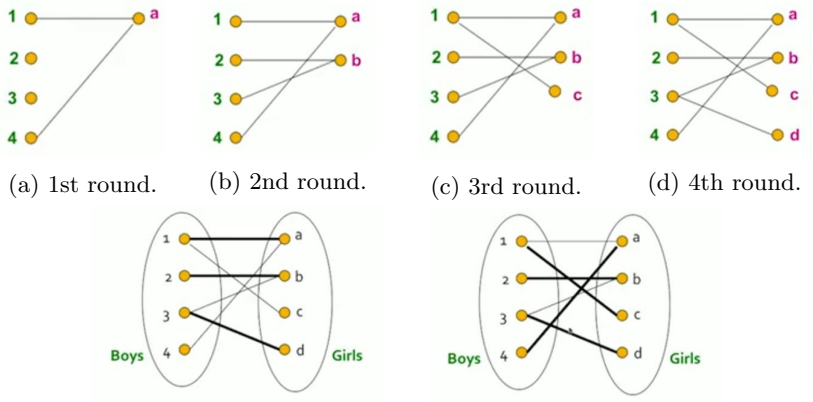
Assignment 10

Application of Online Bipartite Matching Say in a bipartite graph, left(L) and right(R) set represents set of boys and girls respectively. The edges between them denotes compatible pairs. Our goal is to match as many compatible pairs as possible. This problem becomes critical when matching is to be done online. Initially we are given the set of boys and girls. In each round, one girl's choices (edges) are revealed. At that time, we have to decide either to pair that girl with a boy or do not pair with any boy. In Figure 6a one girl's choice is revealed and matching is done (Figure 6b). But we can not match the 2nd girl appeared in Figure 6c. However, optimal matching has cardinality 2 (Figure 6d). So, here competitive ratio is $\frac{1}{2}$. Figure 7 contains another example where each set contains 4 nodes. Possible greedy solution and optimal solution are given in Figure 7e & 7f.



(a) "a" arrived. (b) 1st matching. (c) "b" arrived. (d) Optimal.

Figure 6: Online Bipartite Matching with two nodes in each set. L={1,2},R={a,b}.



(a) 1st round. (b) 2nd round. (c) 3rd round. (d) 4th round.

(e) Greedy solution. (f) Optimal solution.

Figure 7: Online Bipartite Matching.L={1,2,3,4},R={a,b,c,d}.

- Initially we are given the sets L & R but the edges are not given.
- In each round, adjacent edges are revealed for one element of set R. At that time we have to decide either to match or not to match.
- To find out maximum matching, greedy algorithm can be applied with competitive ratio $\frac{1}{2}$.

The killer application is Web advertising. The vertices of L, which are known up front, represent advertisers who have purchased a contract for having their ad shown to users that meet specifed demographic criteria. For example, an advertiser might pay (in advance) to have their ad shown to women between the ages of 25 and 35 who live within 100 miles of New York City. If an advertiser purchased 5000 views, then there will be 5000 corresponding vertices on the left-hand side. The right-hand side vertices, which arrive online, correspond to "eyeballs". When someone types in a search query or accesses a content page (a new opportunity to show ads), it corresponds to the arrival of a vertex $w \in R$. The edges incident to w correspond to the advertisers for whom w meets their targeting criteria. Adding an edge to the matching then corresponds to showing a given ad to the newly arriving eyeball.

Hint: Refer CS261: A Second Course in Algorithms Lecture 14: Online Bipartite Matching by Tim Roughgarden. *http://timroughgarden.org/w16/l/l14.pdf.* Video Lecture 63 - Computational Advertising Bipartite Graph Matching, Stanford. *http://infolab.stanford.edu/∼ullman/mining/2009/advertising.pdf*

Assignment 11

Application of Load Balancing Say there are m identical machines and n jobs. Job j has processing time t_j. Any job j must run contiguously on one machine. Each machine can process at most one job at a time, but we can assign multiple jobs in one machine for future processing. The *Load* of a machine is the total processing time of jobs assigned to that machine. The *makespan* is the maximum load on any machine. Design a 2- approximation algorithm to assign jobs to the machines with an aim to minimize the *makespan*.

Hint: Assign job j to machine i whose load is the smallest so far. Keep the complexity under $O(n * logm)$. Refer *https://www.cs.princeton.edu/∼wayne/kleinberg-tardos/pdf/11ApproximationAlgorithms.pdf*

Assignment 12

Knapsack Problem A thief robbing a store finds n items. The i-th item is worth v_i bucks and weighs w_i pounds, where v_i and w_i are integers. The thief wants to take as valuable a load as possible, but he can carry at most W pounds in his knapsack, for some integer W. This is called $Fractional\ Knapsack\ problem$ as the thief can take fractions of items, rather than having to make a binary (0-1) choice for each item. Design a greedy algorithm to maximize the value without violating any constraint.

Assignment 13

Application of closest pair finding Say there are n number of ships in a specific region of sea. An observer can track location of these ships from outside. The observer is supposed to alert the pair of ships who are the closest. Design an algorithm for that observer to find out the closest ships in the region.

This problem can be mapped into a famous computational geometry problem named "*Closest Pair Finding*". If we imagine the sea as a 2-D plane, locations of those ships can be represented by coordinate points. Now, the problem can be stated as follows: Given a set of points $\{p_1, p_2, p_3, ...p_n\}$, find the pair of points $\{p_i, p_j\}$ that are the closest.

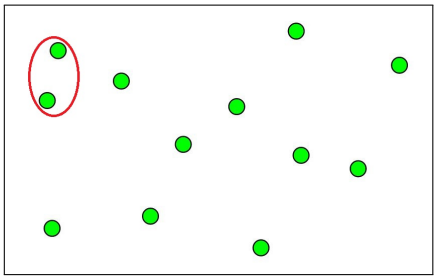


Figure 8: The closest points are marked.

Hint: Refer *Introduction to Algorithms* book by T.H Cormen et. al (CLRS); Chapter: Computational Geometry.

Refer *Algorithm Design* book by Jon Kleinberg and Eva Tardos (KT book) in *Divide and Conquer*, *Randomized Algorithms* chapters (Follow the term Finding the Closest Pair of Points).