091M4041H - Assignment 1

Algorithm Design and Analysis

September 19, 2016

Notice:

1. Please submit your answer in hard copy AND submit a digital version to UCAS website http://sep.ucas.ac.cn.
2. Hard copy should be submitted before 9 am. September 30 and digital version should be submitted before 12 pm. September 30.
3. You can choose **three** from problems 1-7, and choose **two** from problems 8-11.
4. For problems 1-7, you should do at least the following things:
   (a) Describe your algorithm in natural language **AND** pseudo-code;
   (b) Draw a "subproblem reduction graph", where nodes represent subproblems, and edges describe the "reduction relationship" between them for every problem you choose in problems 1-7;
   (c) Prove the correctness of your algorithm;
   (d) Analyse the complexity of your algorithm.
5. For problems 8-11, you should implement your algorithm in C/C++/Java/Python with good comments.

1 Divide and Conquer

You are interested in analyzing some hard-to-obtain data from two separate databases. Each database contains n numerical values, so there are $2n$ values total and you may assume that no two values are the same. You'd like to determine the median of this set of $2n$ values, which we will define here to be the n^{th} smallest value.

However, the only way you can access these values is through *queries* to the databases. In a single query, you can specify a value k to one of the two databases, and the chosen database will return the k^{th} smallest value that it contains. Since queries are expensive, you would like to compute the median using as few queries as possible.

Give an algorithm that finds the median value using at most $O(\log n)$ queries.

2 Divide and Conquer

Find the k^{th} largest element in an unsorted array. Note that it is the kth largest element in the sorted order, not the k^{th} distinct element.

INPUT: An unsorted array A and k.

OUTPUT: The k^{th} largest element in the unsorted array A.

3 Divide and Conquer

Consider an n-node complete binary tree T, where $n = 2^d - 1$ for some d. Each node v of T is labeled with a real number x_v. You may assume that the real numbers labeling the nodes are all distinct. A node v of T is a *local minimum* if the label x_v is less than the label x_w for all nodes w that are joined to v by an edge.

You are given such a complete binary tree T, but the labeling is only specified in the following *implicit* way: for each node v, you can determine the value x_v by *probing* the node v. Show how to find a local minimum of T using only $O(\log n)$ *probes* to the nodes of T.

4 Divide and Conquer

Suppose now that you're given an $n \times n$ grid graph G. (An $n \times n$ grid graph is just the adjacency graph of an $n \times n$ chessboard. To be completely precise, it is a graph whose node set is the set of all ordered pairs of natural numbers (i, j), where $1 \leqslant i \leqslant n$ and $1 \leqslant j \leqslant n$; the nodes (i, j) and (k, l) are joined by an edge if and only if $|i - k| + |j - l| = 1$.)

We use some of the terminology of problem 3. Again, each node v is labeled by a real number x_v; you may assume that all these labels are distinct. Show how to find a local minimum of G using only $O(n)$ probes to the nodes of G. (Note that G has n^2 nodes.)

5 Divide and Conquer

every year the cows hold an event featuring a peculiar version of hopscotch that involves carefully jumping from rock to rock in a river. The excitement takes place on a long, straight river with a rock at the start and another rock at the end, L units away from the start ($1 \leq L \leq 1,000,000,000$). Along the river between the starting and ending rocks, N ($0 \leq N \leq 50,000$) more rocks appear, each at an integral distance D_i from the start ($0 < D_i < L$).

To play the game, each cow in turn starts at the starting rock and tries to reach the finish at the ending rock, jumping only from rock to rock. Of course, less agile cows never make it to the final rock, ending up instead in the river.

Farmer John is proud of his cows and watches this event each year. But as time goes by, he tires of watching the timid cows of the other farmers limp across the short distances between rocks placed too closely together. He plans to remove several rocks in order to increase the shortest distance a cow will have to jump to reach the end. He knows he cannot remove the starting and ending rocks, but he calculates that he has enough resources to remove up to M rocks ($0 \leq M \leq N$).

Farmer John wants to know exactly how much he can increase the shortest distance before he starts removing the rocks. Help Farmer John determine the greatest possible shortest distance a cow has to jump after removing the optimal set of M rocks. In other words, you need help John to find a way to remove M blocks, so that in the rest of the blocks, the distance between two adjacent blocks which have a minimum spacing is the largest.

6 Divide and Conquer

Recall the problem of finding the number of inversions. As in the course, we are given a sequence of n numbers $a_1, \cdots, a_n$, which we assume are all distinct, and we difine an inversion to be a pair $i < j$ such that $a_i > a_j$.

We motivated the problem of counting inversions as a good measure of how different two orderings are. However, one might feel that this measure is too sensitive. Let's call a pair a *significant inversion* if $i < j$ and $a_i > 3a_j$. Given an $O(n \log n)$ algorithm to count the number of significant inversions between two orderings.

7 Divide and Conquer

A group of n ghostbusters is battling n ghosts. Each ghostbuster is armed with a proton pack, which shoots a stream at a ghost, eradicating it. A stream goes in a straight line and terminates when it hits the ghost. The ghostbusters decide upon the following strategy. They will pair off with the ghosts, forming n ghostbuster-ghost pairs, and then simultaneously each ghostbuster will shoot a stream at his chosen ghost. As we all know, it is very dangerous to let streams cross, and so the ghostbusters must choose pairings for which no streams will cross. Assume that the position of each ghostbuster and each ghost is a fixed point in the plane and that no three positions are collinear.

1. Show that there exists a line passing through one ghostbuster and one ghost such the number of ghostbusters on one side of the line equals the number of ghosts on the same side. Describe how to find such a line in $O(n \log n)$ time.

2. Give an $O(n^2 \log n)$-time algorithm to pair ghostbusters with ghosts in such a way that no streams cross.

8 Divide and Conquer

The attached file Q8.txt contains 100,000 integers between 1 and 100,000 (each row has a single integer), the order of these integers is random and no integer is repeated.

1. Write a program to implement the Sort-and-Count algorithms in your favorite language, find the number of inversions in the given file.

2. In the lecture, we count the number of inversions in $O(n \log n)$ time, using the Merge-Sort idea. Is it possible to use the Quick-Sort idea instead ?
   If possible, implement the algorithm in your favourite language, run it over the given file, and compare its running time with the one above. If not, give a explanation.

9 Dvide and Conquer

Implement the algorithm for the closest pair problem in your favourite language.

INPUT: n points in a plane.

OUTPUT: The pair with the least Euclidean distance.

10 Divide and Conquer

Implement the Strassen algorithm algorithm for MatrixMultiplication problem in your favourite language, and compare the performance with grade-school method.

11 Divide and Conquer

Implement the Karatsuba algorithm for Multiplication problem in your favourite language, and compare the performance with quadratic grade-school method.

091M4041H - Assignment 2

Algorithm Design and Analysis

October 7, 2016

Notice:

1. Please submit your answer in hard copy AND submit a digital version to UCAS website http://sep.ucas.ac.cn.
2. Hard copy should be submitted before 9 am. October 21 and digital version should be submitted before 12 pm. October 21.
3. You can choose **three** from problems 1-5, and choose **one** from problems 6-7.
4. For problems 1-5, you should do at least the following things:
   (a) Describe the optimal substructure and DP equation;
   (b) Describe your algorithm in daily language or pseudo-code;
   (c) Prove the correctness of your algorithm;
   (d) Analyse the complexity of your algorithm.
5. For problems 6-7, you should implement your algorithm in C/C++/Java/Python with good comments.

1 Largest Divisible Subset

Given a set of distinct positive integers, find the largest subset such that every pair (S_i, S_j) of elements in this subset satisfies:$S_i \% S_j = 0$ or $S_j \% S_i = 0$.

2 Money robbing

A robber is planning to rob houses along a street. Each house has a certain amount of money stashed, the only constraint stopping you from robbing each of them is that adjacent houses have security system connected and it will automatically contact the police if two adjacent houses were broken into on the same night.

1. Given a list of non-negative integers representing the amount of money of each house, determine the maximum amount of money you can rob tonight without alerting the police.
2. What if all houses are arranged in a circle?

3 Partition

Given a string s, partition s such that every substring of the partition is a palindrome. Return the minimum cuts needed for a palindrome partitioning of s.

For example, given $s =$ "aab", return 1 since the palindrome partitioning ["aa", "b"] could be produced using 1 cut.

4 Decoding

A message containing letters from A-Z is being encoded to numbers using the following mapping:

$$\begin{aligned} &\text{A} : 1 \\ &\text{B} : 2 \\ &\ldots \\ &\text{Z} : 26 \end{aligned}$$

Given an encoded message containing digits, determine the total number of ways to decode it.

For example, given encoded message "12", it could be decoded as "AB" (1 2) or "L" (12). The number of ways decoding "12" is 2.

5 Frog Jump

A frog is crossing a river. The river is divided into x units and at each unit there may or may not exist a stone. The frog can jump on a stone, but it must not jump into the water.

If the frog's last jump was k units, then its next jump must be either $k-1, k$, or $k+1$ units. Note that the frog can only jump in the forward direction.

Given a list of stones' positions (in units) in sorted ascending order, determine if the frog is able to cross the river by landing on the last stone. Initially, the frog is on the first stone and assume the first jump must be 1 unit.

6 Maximum profit of transactions

You have an array for which the i-th element is the price of a given stock on day i.

Design an algorithm and implement it to find the maximum profit. You may complete at most two transactions.

Note: You may not engage in multiple transactions at the same time (ie, you must sell the stock before you buy again).

7 Maximum length

Given a sequence of n real numbers $a_1, \ldots, a_n$, determine a subsequence (not necessarily contiguous) of maximum length in which the values in the subsequence form a strictly increasing sequence.

Assignment 3

Algorithm Design and Analysis

October 14, 2016

Notice:

1. • **Due** 9:00 a.m., Nov. 28, 2016 for graduate students in UCAS;
2. Please submit your answers in hard copy AND submit a digital version to UCAS website https://www2.ucas.ac.cn/.
3. Please choose at least two problems from Problem 1-4, and choose at least one problem from Problem 5-6.
4. When you're asked to give an algorithm, you should do at least the following things:
   - Describe the basic idea of your algorithm in natural language **AND** pseudo-code;
   - Prove the correctness of your algorithm.
   - Analyse the complexity of your algorithm.

1 Greedy Algorithm

Given a list of n natural numbers d_1, d_2,...,d_n, show how to decide in polynomial time whether there exists an undirected graph $G = (V,\ E)$ whose node degrees are precisely the numbers $d_1, d_2, \cdots, d_n$. G should not contain multiple edges between the same pair of nodes, or " loop" edges with both endpoints equal to the same node.

2 Greedy Algorithm

There are n distinct jobs, labeled $J_1, J_2, \cdots, J_n$, which can be performed completely independently of one another. Each jop consists of two stages: first it needs to be *preprocessed* on the supercomputer, and then it needs to be *finished* on one of the PCs. Let's say that job J_i needs p_i seconds of time on the supercomputer, followed by f_i seconds of time on a PC. Since there are at least n PCs available on the premises, the finishing of the jobs can be performed on PCs at the same time. However, the supercomputer can only work on a single job a time without any interruption. For every job, as soon as the preprocessing is done on the supercomputer, it can be handed off to a PC for finishing.

Let's say that a *schedule* is an ordering of the jobs for the supercomputer, and the *completion time* of the schedule is the earlist time at which all jobs have finished processing on the PCs. Give a polynomial-time algorithm that finds a schedule with as small a completion time as possible.

3 Greedy Algorithm

Assume the coasting is an infinite straight line. Land is in one side of coasting, sea in the other. Each small island is a point locating in the sea side. And any radar installation, locating on the coasting, can only cover d distance, so an island in the sea can be covered by a radius installation, if the distance between them is at most d.

We use Cartesian coordinate system, defining the coasting is the x-axis. The sea side is above x-axis, and the land side below. Given the position of each island in the sea, and given the distance of the coverage of the radar installation, your task is to write a program to find the minimal number of radar installations to cover all the islands. Note that the position of an island is represented by its x-y coordinates.

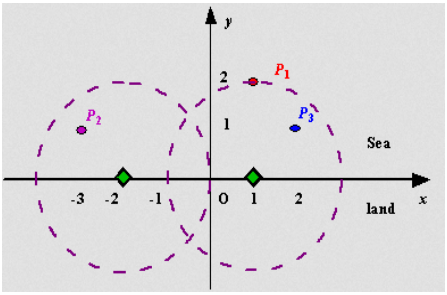


Figure 1

4 Greey Algorithm

Suppose you are given two sets A and B, each containing n positive integers. You can choose to reorder each set however you like. After reordering, let a_i be the ith element of set A, and let b_i be the ith element of set B. You then receive a payoff of $\prod_{i=1}^{n} a_i^{b_i}$. Give an polynomial-time algorithm that will maximize your payoff.

5 Programming

Write a program in your favorate language to compress a file using Huffman code and then decompress it. Code information may be contained in the compressed file if you can. Use your program to compress the two files (*graph.txt* and *Aesop_Fables.txt*) and compare the results (Huffman code and compression ratio).

6 Programming

1. Implement Dijkstra's algorithm (using linked list, binary heap, binomial heap, and Fibonacci heap) to calculate the shortest path from node s to node t of the given graph (*graph.txt*), where s and t are randomly chosen. The comparison of different priority queue is expected.

   Note: you can implement the heaps by yourself or using Boost C++/STL, etc.

2. Figure out how many shortest paths is every node lying on in your program, except starting node s and finishing node t. For example, if there are in total three shortest paths $0 \to 1 \to 2 \to 10$, $0 \to 1 \to 3 \to 4 \to 10$ and $0 \to 1 \to 2 \to 6 \to 7 \to 10$, then 1 lies on 3 shortest paths, 2 lies on 2 shortest paths, and 3 lies on 1 shortest path, etc.

Assignment 4

Algorithm Design and Analysis

Notice:

1. **Due** Nov. 11, 2016 for graduate students at UCAS.
2. Please submit your answers in hard copy, **AND** graduate stuents at UCAS should also submit a digital version to UCAS website https://sep.ucas.ac.cn/.
3. Please choose at least two problems from problems 1-6, and choose at least one problem from problems 7-8.
4. Note that problem 1.2 and problem 6 require knowledge on duality.
5. Integer Linear Programming is different from the classic Linear Programming that some extra constraints such as

   $$x_i \text{ is an integer, } for\ all\ i = 1, 2, \cdots, n$$

   or

   $$x_i \in \{0, 1\},\ for\ all\ i = 1, 2, \cdots, n$$

   are added.
6. When you give the formulation of an LP or ILP, you should explain all mathematical symbols you are using if not appearing in the problem, and interpret the constraints if necessary.

1 Linear-inequality feasibility

Given a set of m linear inequalities on n variables $x_1, x_2, \cdots, x_n$, the **linear-inequality feasibility problem** asks if there is a setting of the variables that simultaneously satisfies each of the inequalities.

1. Show that if we have an algorithm for linear programming, we can use it to solve the linear-inequality feasibility problem. The number of variables and constraints that you use in the linear-programming problem should be polynomial in n and m.
2. Show that if we have an algorithm for the linear-inequality feasibility problem, we can use it to solve a linear-programming problem. The number of variables and linear inequalities that you use in the linear-inequality feasibility problem should be polynomial in n and m, the number of variables and constraints in the linear programming.

2 Airplane Landing Problem

With human lives at stake, an air traffic controller has to schedule the airplanes that are landing at an airport in order to avoid airplane collision. Each airplane i has a time window $[s_i, t_i]$ during which it can safely land. You must compute the exact time of landing for each airplane that respects these time windows. Furthermore, the airplane landings should be stretched out as much as possible so that the minimum time gap between successive landings is as large as possible.

For example, if the time window of landing three airplanes are [10:00-11:00], [11:20-11:40], [12:00-12:20], and they land at 10:00, 11:20, 12:20 respectively, then the smallest gap is 60 minutes, which occurs between the last two airplanes.

Given n time windows, denoted as $[s_1, t_1], [s_2, t_2], \cdots, [s_n, t_n]$ satisfying $s_1 < t_1 < s_2 < t_2 < \cdots < s_n < t_n$, you are required to give the exact landing time of each airplane, in which the smallest gap between successive landings is maximized.

Please formulate this problem as an LP, construct an instance and use GLPK or Gurobi or other similar tools to solve it.

3 Interval Scheduling Problem

A teaching building has m classrooms in total, and n courses are trying to use them. Each course i $(i = 1, 2, \cdots, n)$ only uses one classroom during time interval $[S_i, F_i)$ $(F_i > S_i > 0)$. Considering any two courses can not be carried on in a same classroom at any time, you have to select as many courses as possible and arrange them without any time collision. For simplicity, suppose $2n$ elements in the set $\{S_1, F_1, \cdots, S_n, F_n\}$ are all different.

1. Please use ILP to solve this problem, then construct an instance and use GLPK or Gurobi or other similar tools to solve it.

2. If you relax the integral constraints and change ILP to an LP (e.g. change $x \in \{0, 1\}$ to $0 \leqslant x \leqslant 1$), will solution of the LP contains only integers, regardless of values of all S_i and F_i? If it's true, prove it; if it's false, give a counter example. You can use the following lemma for help.

   Lemma If matrix A has only 0, +1 or -1 entries, and each column of A has at most one +1 entry and at most one -1 entry. In addtion, the vector b has only integral entries. Then the vertex of polytope $\{x | Ax \leqslant b, x \geqslant 0\}$ contains only integral entries.

4 Gas Station Placement

Let's consider a long, quiet country road with towns scattered very sparsely along it. Sinopec, largest oil refiner in China, wants to place gas stations along the road. Each gas station is assigned to a nearby town, and the distance between any two gas stations being as small as possible. Suppose there are n towns with distances from one endpoint of the road being $d_1, d_2, \cdots, d_n$. n gas stations are to be placed along the road, one station for one town. Besides, each station is at most r far away from its correspond town. $d_1, \cdots, d_n$ and r have

been given and satisfied $d_1 < d_2 < \cdots < d_n$, $0 < r < d_1$ and $d_i + r < d_{i+1} - r$ for all i. The objective is to find the optimal placement such that the maximal distance between two successive gas stations is minimized.

Please formulate this problem as an LP.

5 Stable Matching Problem

n men $(m_1, m_2, \cdots, m_n)$ and n women $(w_1, w_2, \cdots, w_n)$, where each person has ranked all members of the opposite gender, have to make pairs. You need to give a stable matching of the men and women such that there is no unstable pair. Please choose one of the two following known conditions, formulate the problem as an ILP (*hint*: Problem 1.1 in this assignment), construct an instance and use GLPK or Gurobi or other similar tools to solve it.

1. You have known that for every two possible pairs (man m_i and woman w_j, man m_k and woman w_l), whether they are stable or not. If they are stable, then $S_{i,j,k,l} = 1$; if not, $S_{i,j,k,l} = 0$. $(i, j, k, l \in \{1, 2, \cdots, n\})$
2. You have known that for every man m_i, whether m_i likes woman w_j more than w_k. If he does, then $p_{i,j,k} = 1$; if not, $p_{i,j,k} = 0$. Similarly, if woman w_i likes man m_j more than m_k, then $q_{i,j,k} = 1$, else $q_{i,j,k} = 0$. $(i, j, k \in \{1, 2, \cdots, n\})$

6 Duality

Please write the dual problem of the MultiCommodityFlow problem in *Lec8.pdf*, and give an explanation of the dual variables.

Please also construct an instance, and try to solve both primal and dual problem using GLPK or Gurobi or other similar tools.

7 Simplex Algorithm

Please implement simplex algorithm or dual simplex algorithm with your favorate language, and make comparison with GLPK or Gurobi or other similar tools.

8 LP And Iterative Algorithm

There are n real numbers $x_1, x_2, \cdots, x_n$, and $x_1 = 0$. We want to choose $n - 1$ values for $x_2, \cdots, x_n$ such that $m = \Theta(n)$ ($\lim_{n\to\infty} \frac{\Theta(n)}{n}$ is a constant) constraints, each looking like $x_i - x_j = d_{ij}$ $(i \neq j)$, can be satisfied. Each x_i and x_j pair $(i, j \in \{1, 2, \cdots, n\}$ and $i \neq j)$ will appear in the same constraint at most only once. However, we cannot satisfy all these m constraints simultaneously with high probability. For example, the constraints

$$x_i - x_j = 1$$
$$x_i - x_k = 1$$
$$x_j - x_k = 1$$

cannot be satisfied simultaneously. As a result, we have to relax the limitations to $|x_i - x_j - d_{ij}| \leqslant \varepsilon_{ij}$ ($\varepsilon_{ij} \geqslant 0$) and try to minimize $\sum \varepsilon_{ij}$.

1. Formulate this problem as an LP.

2. Another algorithm to solve this problem is shown below. Implement this algorithm with your favorate language.

Algorithm 1 Problem 8

Construct an undirected graph $G = (V, E)$ with n isolated vertices $v_1, \cdots, v_n$;
for each constraint $x_i - x_j = d_{ij}$ **do**
    Connect v_i and v_j with an edge e_{ij};
Set $x_1^{(0)} = 0$;
BFS the graph from v_1, and set initial values of other vertices according to constraints on edges (e.g. when you are visiting unvisited vertex v_j from v_i along the edge e_{ij} with constraint $x_i - x_j = d_{ij}$, set $x_j^{(0)} = x_i^{(0)} - d_{ij}$);
Set $t = 0$;
repeat
    for $i = 1 \to n$ **do**
        Clear set S;
        for each vertex x_j which is connected to x_i by an edge with constraint $x_i - x_j = d_{ij}$ **do**
            Put $x_i^{(t+1)} = x_j^{(t)} + d_{ij}$ to set S;
        Sort elements in set S and set $\lfloor \frac{|S|+1}{2} \rfloor$-th smallest element as $x_i^{(t+1)}$;
    $t++$;
until convergence;
Output values of $0, x_2^{(t+1)} - x_1^{(t+1)}, \cdots, x_n^{(t+1)} - x_1^{(t+1)}$ as $x_1, x_2, \cdots, x_n$.

3. Construct an instance, then use GLPK or Gurobi or other similar tools to solve your LP, use the algorithm you implement previously to solve the problem, and compare the results and running time.

Assignment 5

Algorithm Design and Analysis

December 6, 2016

Notice:

1. **Due** 9:00 a.m., Dec. 09 for hard copy and 11:55 p.m., Dec. 09 for digital version;
2. Please submit your answers in hard copy **AND** submit a digital version to UCAS website https://www2.ucas.ac.cn/ .
3. Please choose **at least 4** problems from Problem 1-7 and **at least 2** problems from Problem 8-10.
4. When you're asked to give an algorithm, you should do at least the following things:
   - Describe the basic idea of your algorithm in natural language **AND** pseudo-code;
   - Prove the correctness of your algorithm.
   - Analyse the complexity of your algorithm.

1 Load balance

You have some different computers and jobs. For each job, it can only be done on one of two specified computers. The load of a computer is the number of jobs which have been done on the computer. Give the number of jobs and two computer ID for each job. You task is to minimize the max load.

(hint: binary search)

2 Matrix

For a matrix filled with 0 and 1, you know the sum of every row and column. You are asked to give such a matrix which satisfys the conditions.

3 Unique Cut

Let $G = (V, E)$ be a directed graph, with source $s \in V$, sink $t \in V$, and nonnegative edge capacities c_e. Give a polynomial-time algorithm to decide whether G has a unique minimum st cut.

4 Problem Reduction

There is a matrix with numbers which means the cost when you walk through this point. you are asked to walk through the matrix from the top left point to the right bottom point and then return to the top left point with the minimal cost. Note that when you walk from the top to the bottom you can just walk to the right or bottom point and when you return, you can just walk to the top or left point. And each point CAN NOT be walked through more than once.

5 Network Cost

For a network, there is one source and one sink. Every edge is directed and has two value c and a. c means the maximum flow of the adge. a is a coefficient number which means that if the flow of the edge is x, the cost is ax^2.

Design an algorithm to get the Minimum Cost Maximum Flow.

6 Maximum Cohesiveness

Given an undirected graph, each edge is assigned one weight, find a subset S of nodes to maximize $e(S)/|S|$, where $e(S)$ denotes the sum of edge weights in S and $|S|$ is the number of nodes in S. Give a polynomial-time algorithm that takes a rational number α and de- termines whether there exists a set S with cohesiveness at least α.

7 Maximum flow

Another way to formulate the maximum-flow problem as a linear program is via flow decomposition. Suppose we consider all (exponentially many) s-t paths p in the network G, and let f_p be the amount of flow on path p. Then maximum flow says to find

$$\begin{array}{rrcl} \max & z & = & \sum f_p \\ s.t. & \sum_{e \in p} f_p & \leq & u_e, \quad for \quad all \quad edge \quad e \\ & f_p & \geq & 0 \end{array}$$

(The first constraint says that the total flow on all paths through e must be less than ue.) Take the dual of this linear program and give an English explanation of the objective and constraints.

8 Ford-Fulkerson algorithm

Implement Ford-Fulkerson algorithm to find the maximum flow of the following network, and list your intermediate steps. Use you implementation to solve problem 1 and show your answers.
INPUT: (N, M) means number of jobs and computers. Next N line, each line has two computer ID for a job. see more detial in the file problem1.data.
OUTPUT: the minimum number of the max load.

9 Push-relabel

Implement push-relabel algorithm to find the maximum flow of a network, and list your intermediate steps. Use your implementation to solve problem 2 and write a check problem to see if your answer is right.
INPUT: Numbers of rows and columns. And the sum of them. See more detial in the file problem2.data.
OUTPUT: The matrix you get. Any one satisfy the conditions will be accept.

10 Cycle canceling

Implement Cycle canceling algorithm to find the minimum cost flow of a network, and list your intermediate steps.
INPUT: A directed graph $G =< V, E >$. Each edge e has a capacity c_e and a cost w_e. Two special points: source s and sink t. Please make the input in DIMACS format. For DIMACS format, see DIMACS maximum flow problems.html in this folder.
OUTPUT: For each edge e, to assign a flow f_e such that $\sum_{e \in E} f_e w_e$ is minimized.

091M4041H Assignment 7

Algorithm Design and Analysis

December 12, 2016

Notice:

1. Deadline: Dec. 30, 2016

2. Please submit your answers in hard copy AND submit a digital version to UCAS website https://sep.ucas.ac.cn/.
3. You can arbitrarily choose two from Problems 1-7.
4. In this assignment, only the following problems which have been proved in NP-complete in slides of this course can be used as the well-known NP-complete problem, in order to avoid circular reasoning:
   - SAT
   - 3SAT
   - CLIQUE
   - Independent Set
   - Vertex Cover
   - Set Cover
   - Subset Sum
   - 3 Coloring
   - Hamiltonian Cycle

1 Integer Programming

Given an integer $m \times n$ matrix A and an integer m-vector b, the Integer programming problem asks whether there is an integer n-vector x such that $Ax \geq b$. Prove that Integer-programming is in NP-complete.

2 Mine-sweeper

This problem is inspired by the single-player game Mine-sweeper, generalized to an arbitrary graph. Let G be an undirected graph, where each node

either contains a single, hidden mine or is empty. The player chooses nodes, one by one. If the player chooses a node containing a mine, the player loses. If the player chooses an empty node, the player learns the number of neighboring nodes containing mines. (A neighboring node is one connected to the chosen node by an edge.). The player wins if and when all empty nodes have been so chosen.

In the **mine consistency problem**, you are given a graph G, along with numbers labeling some of G's nodes. You must determine whether a placement of mines on the remaining nodes is possible, so that any node v that is labeled m has exactly m neighboring nodes containing mines. Formulate this problem as a language and show that it is NP-complete.

3 Half-3SAT

In the Half-3SAT problem, we are given a 3SAT formula ϕ with n variables and m clauses, where m is even. We wish to determine whether there exists an assignment to the variables of ϕ such that exactly half the clauses evaluate to false and exactly half the clauses evaluate to true. Prove that Half-3SA problem is in NP-complete.

4 Solitaire Game

In the following solitaire game, you are given an $n \times n$ board. On each of its n^2 positions lies either a blue stone, a red stone, or nothing at all. You play by removing stones from the board so that each column contains only stones of a single color and each row contains at least one stone. You win if you achieve this objective.

Winning may or may not be possible, depending upon the initial configuration. You must determine the given initial configuration is a winnable game configuration. Let **SOLITAIRE** $= \{< G > | G$ is a winnable game configuration$\}$. Prove that **SOLITAIRE** is NP-complete.

5 Directed Disjoint Paths Problem

The **Directed Disjoint Paths Problem** is defined as follows. We are given a directed graph G and k pairs of nodes $(s_1, t_1), (s_2, t_2), \cdots, (s_k, t_k)$. The problem is to decide whether there exist node-disjoint paths $P_1, P_2, \cdots, P_k$ so that P_i goes from s_i to t_i. In details, the node-disjoint paths means that P_i and P_j $(1 \leq i \leq k, 1 \leq j \leq k, i \neq j)$share no nodes.

Show that **Directed Disjoint Paths** is NP-complete.

6 SIP problem

The **set intersection problem (SIP)** is defined as follows: Given finite sets $A_1, A_2, \ldots, A_r$ and $B_1, B_2, \ldots, B_s$, is there a set T such that

$$|T \cap A_i| \geqslant 1 \quad \text{for } i = 1, 2, \ldots, r$$

and

$$|T \cap B_i| \leqslant 1 \quad \text{for } i = 1, 2, \ldots, s$$

Prove that the **SIP** is NP-complete.

7 Your favorite intelligence game

Choose one of your favorite intelligence games such as Air Solitaire, Sokoban, Sudoku, 2048 or other games.

Firstly please formulate the game. Maybe the rules of the game is too complex to formulate. To make it easy to analyze, you can simplify the rules or alter the objective of the game while the essential properties of game are still preserved.

Then please analyze the complexity of the game. Give a strategy with polynomial time complexity or prove that the game is NP complete.

You can refer to the Second problem **Mine-Sweep** in this problem set for an example.

091M4041H: Final Examination

2016

Notice:

1. Please write your name along with student ID;
2. There are 8 sections in the sheet, and for sections 1-6, you can arbitrarily choose one problem. If you answer two problems, say 1.1 and 1.2, the higher mark will be chosen;
3. When you are asked to give an algorithm, you should describe your algorithm in natural language or pseudo-codes, prove the correctness, and analyze time complexity;
4. You can write answers in either Chinese or English.

1 Divide and Conquer

(12 marks)

1.1

1.2

2 Dynamic Programming

(12 marks)

2.1

2.2

3 Greedy Algorithm

(12 marks)

3.1

3.2

4 Linear Programming Formulation

(12 marks)

4.1

4.2

5 Network Flow Formulation

(12 marks)

5.1

5.2

6 NP-completeness Reduction

(12 marks)

6.1

6.2

7 Bonus 1

(14 marks)

8 Bonus 2

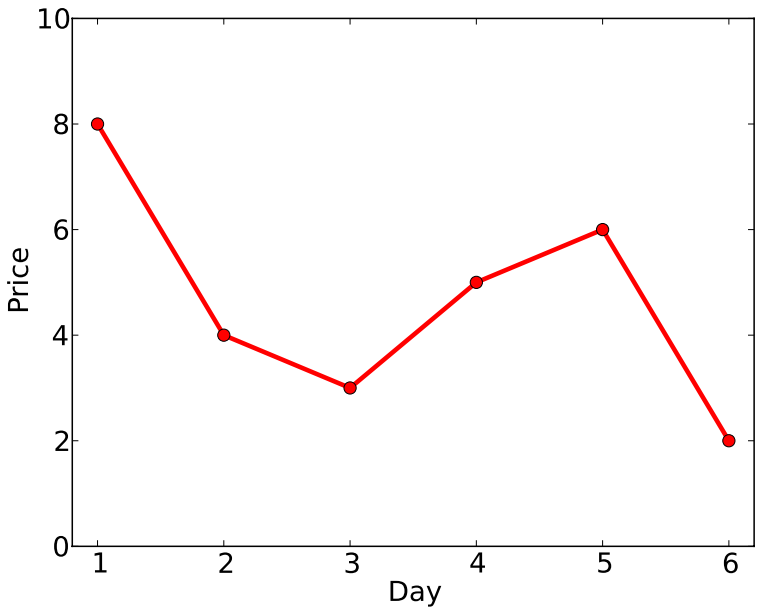
(14 marks)

Final Examination Example

1 Problem in Assignment

In stock market, HH-index (historically highest) of the current price is k means that current price is the highest price in the previous k days, but not the highest one in the previous $k+1$ days.

For example, the price changes as showed in the following figure.



The price in Day 5 is the highest in Days $5, 4, 3, 2$, but not highest one in Days $5, 4, 3, 2, 1$, thus $HH(5) = 4$;

The price in Day 4 is the highest in Days $4, 3, 2$, but not highest one in Days $4, 3, 2, 1$, thus $HH(4) = 3$;

The price in Day 3 is the highest in Day 3, but not highest one in Days $3, 2$, thus $HH(3) = 1$;

The input in this example is $8, 4, 3, 5, 6, 2$, the answer is $1, 1, 1, 3, 4, 1$. Given the prices of n days, please give an algorithm of $O(n)$ time complexity to calculate the HH−index of all days.

2 Problem in Examination

There are n cows standing in a line, and the cows have **distinct** heights. Suppose that each cow can see all cows in front of her till a cow taller than her.

Given the heights of n cows, please design an algorithm of $O(n)$ time complexity to calculate the number of cows that each cow can see.

For example, if the heights of cows are 4, 7, 8, 6, 3, 9, then the cow of height 4 can see **NO** cows (because she cannot see herself), the cow of height 3 can see **NO** cows, the cow of height 8 can see **2** cows. In conclusion, the cows can see 0, 1, 2, 0, 0, 5 cows in the front, respectively.