Name: Nikhil Sathyanarayana Shastry
USC_ID: 8464475401
Email: nsathyan@usc.edu

CSCI570 - Analysis of Algorithms (HW5)

Q1. In the network below (See Fig. 1), the demand values are shown on vertices (supply values are negative). Lower bounds on flow and edge capacities are shown as (lower bound, capacity) for each edge. Determine if there is a feasible circulation in this graph. Please complete the following steps.

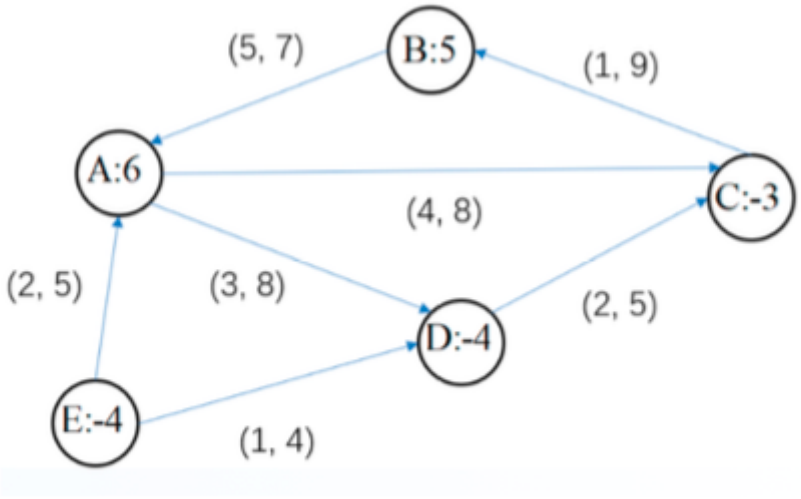


Figure 1

(a) Remove the lower bounds on each edge. Write down the new demands on each vertex A, B, C, D, E, in this order.

(b) Solve the circulation problem without lower bounds. Write down the max-flow value.

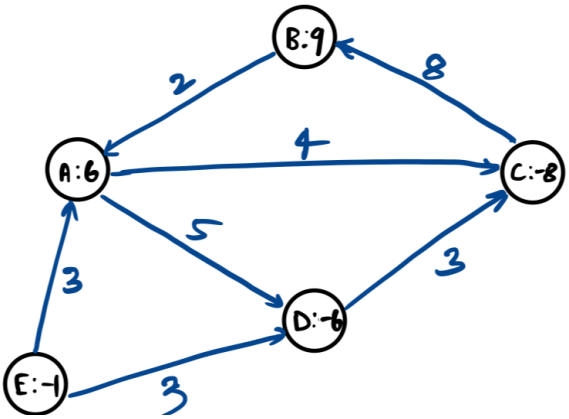
(c) Is there a feasible circulation in the original graph? Explain your answer.

Solution:

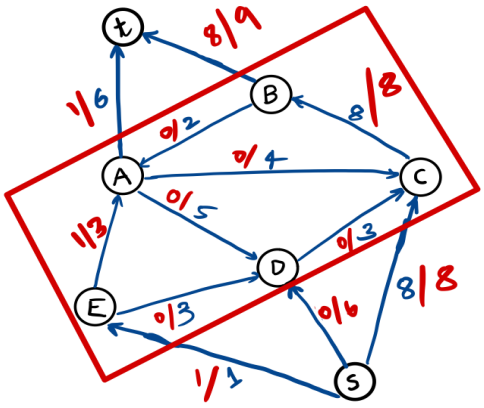
a. The updated demands on each vertex A, B, C, D and E after removing the lower bounds on each edge is as follows:

$$d'(A) = 6 - 5 - 2 + 4 + 3 = 6$$
$$d'(B) = 5 - 1 + 5 = 9$$
$$d'(C) = 1 - 3 - 2 - 4 = -8$$
$$d'(D) = 2 - 4 - 3 - 1 = -6$$
$$d'(E) = 2 + 1 - 4 = -1$$

This is the network after updating the demands and removing the lower bounds:



b. To solve the above circulation and find the max flow we reduce the circulation into a network flow problem by adding the source and the sink vertices. The source ***s*** is connected to all the vertices with negative demands and the edges from the source to these vertices will contain these demands without the negative sign. Similarly, the sink vertex ***t*** is connected to all the vertices with positive demands and the edges from these vertices to the sink will contain these demands. The following network represents the flow/capacity diagram after augmenting flow along the paths to find the max flow:



The max-flow value for the above graph is 9.

c. There is no feasible circulation that exists in the given original graph. This is because not all the edges going out of the source are saturated. A circulation would have been feasible if the max-flow was equal to 15.

Q2. The organizers of the 2022 US Open Tennis Championships are working on making the draws for the first round and scheduling the matches. They are faced with several constraints in doing so.

There are 2N players, seeded (ranked) from 1 to 2N. The players are divided into two halves - the top half containing players seeded 1 to N, and the bottom half containing seeds N + 1 to 2N. The first round consists of N matches among the 2N players, each being one where a top-half player plays a bottom-half player.

Additionally, to keep them a bit more competitive, each match must be between players whose seeds differ by at most N′ (a given constant > N). The sports facility has courts C1, ..., Cm. The matches will be scheduled in timeslots T1, ..., Tk. Matches can run in parallel on different courts in a given time slot. Assume all courts are available in all timeslots, however a single court can have at most p matches to preserve surface quality. For broadcasting reasons, in any given timeslot, there should be at least one match being played and at most r of them.

The bottom-half players were given the option to make requests if they preferred to play in certain timeslots, and each of them has specified k′ of the k time slots that they are okay to play in. Seeded players on the other hand were given the option to make requests to avoid playing on certain courts, and each of them has specified m′ of the m courts that they do not want to play on.

Describe a Network Flow/Circulation based algorithm to determine if it is possible to come up with a feasible schedule of matches based on the above constraints.

Solution:

Q2 REMOVED FROM THE ASSIGNMENT QUESTION POOL

Q3. A company is currently trying to fill a large order of steel, brass, and pewter, measured in tons. Manufacturing each ton of material requires a certain amount of time, and a certain amount of special substance (SS), both of which are limited and are given in below table. Note that it is acceptable to manufacture a fraction of a ton (e.g. 0.5t) of material. Specifically, the company currently has 8 hours of time, and 20 units of special substance (SS). Manufacturing each ton of the three products requires:

Product	Time (hours)	SS (units)
Steel	3	3
Brass	1	10
Pewter	2	5

Figure 2: Table describing the amount of time taken and special substance (SS) required for each product.

(a) Write down a linear program that determines the maximum amount of products (in tons) that the company can make.

(b) Due to the potential danger posed by the special substance (SS), the company would like to use up as much of its supply of special substance (SS) as possible, while:
i. spending at most 8 hours
ii. manufacturing a total of at least 2 tons of steel plus pewter.

Solution:

Let S, B and P be the variables that denote the Steel, Brass, Pewter manufactured in tons respectively.

a. ***Objective function:*** $max(S, B, P)$

   Constraints:

   - Time: $3S + B + 2P <= 8$
   - Special Substance: $3S + 10B + 5P <= 20$

   Here, $S >= 0, B >= 0, P >= 0$

b. ***Objective function:*** $max(3S + 10B + 5P)$

   Constraints:

   - Time: $3S + B + 2P <= 8$
   - Special Substance: $3S + 10B + 5P <= 20$

   - $S + P >= 2$

   $This\ can\ be\ written\ as\ follows: -S - P <= -2$

   Here, $S >= 0, B >= 0, P >= 0$

Q4. Suppose that a cement supplier has two warehouses, one located in city A and another in city B. The supplier receives orders from two customers, one in city C and another in city D. The customer in city C needs at least 50 tons of cement, and the customer in city D needs at least 60 tons of cement. The amount of cement at the warehouse in city A is 70 tons, and the number of units at the warehouse in city B is 80 tons. The cost of shipping each ton of cement from A to C is 1, from A to D is 2, from B to C is 3, and from B to D is 4.
Formulate the problem of deciding how many tons of cement from each warehouse should be shipped to each customer to minimize the total shipping cost as a linear programming. You can assume that the values of units to be shipped are real numbers.

Solution:

Let X_{AC} be the total amount of cement shipped from City A to customer in City C.
Let X_{AD} be the total amount of cement shipped from City A to customer in City D.
Let X_{BC} be the total amount of cement shipped from City B to customer in City C.
Let X_{BD} be the total amount of cement shipped from City B to customer in City D.

We have been given the shipping costs to each of these customers from a particular warehouse and our goal is to minimize the total shipping cost. The following system of equations can be extracted from the given information:

Objective function: $min(X_{AC} + 2X_{AD} + 3X_{BC} + 4X_{BD})$

Constraints:

$$X_{AC} + X_{BC} >= 50$$
$$X_{AD} + X_{BD} >= 60$$
$$X_{AC} + X_{AD} <= 70$$
$$X_{BC} + X_{BD} <= 80$$

Here, $X_{AC} >= 0, X_{BC} >= 0,\ X_{BD} >= 0, X_{AD} >= 0$

We will need to write all the constraints in the standard form. Hence the constraints for the linear program are as follows:

$$max(-X_{AC} - 2X_{AD} - 3X_{BC} - 4X_{BD})$$

$$-X_{AC} - X_{BC} <= -50$$
$$-X_{AD} - X_{BD} <= -60$$
$$X_{AC} + X_{AD} <= 70$$
$$X_{BC} + X_{BD} <= 80$$

Here, $X_{AC} >= 0, X_{BC} >= 0,\ X_{BD} >= 0, X_{AD} >= 0$

Q5. Write down the dual program of the following linear program. There is no need to provide intermediate steps.

$$\max(x_1 - 3x_2 + 4x_3 - x_4)$$

subject to

$$\begin{aligned} x_1 - x_2 - 3x_3 &\leq -1 \\ x_2 + 3x_3 &\leq 5 \\ x_3 &\leq 1 \\ x_1, x_2, x_3, x_4 &\geq 0 \end{aligned}$$

Solution:

$$\mathbf{X}=\begin{bmatrix} X_1 \\ X_2 \\ X_3 \\ X_4 \end{bmatrix} \quad \mathbf{C}=\begin{bmatrix} 1 \\ -3 \\ 4 \\ -1 \end{bmatrix} \quad \mathbf{A}=\begin{bmatrix} 1 & -1 & -3 & 0 \\ 0 & 1 & 3 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \quad \mathbf{B}=\begin{bmatrix} -1 \\ 5 \\ 1 \\ \end{bmatrix}$$

Applying the properties of dual nature:

$\max(c^T x)$		$\min(b^T y)$
$Ax \leq b$	=>	$A^T y \geq c$
$x \geq 0$		$y \geq 0$

$$\mathbf{Y}=\begin{bmatrix} Y_1 \\ Y_2 \\ Y_3 \\ Y_4 \end{bmatrix} \qquad A^T = \begin{bmatrix} 1 & 0 & 0 & 0 \\ -1 & 1 & 0 & 0 \\ -3 & 3 & 1 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

Converting the primal form to dual form we get the following:

$\min(-Y_1 + 5Y_2 + Y_3)$

Subject to:

$Y_1 \geq 1$
$-Y_1 + Y_2 \geq -3$
$-3Y_1 + 3Y_2 + Y_3 \geq 4$
$0Y_1 + 0Y_2 + 0Y_3 + 0Y_4 \geq -1$

Here, $Y_1 >= 0, Y_2 >= 0,\ Y_3 >= 0, Y_4 >= 0$

Q6. Given an undirected graph G = (V,E), a vertex cover is a subset of V so that every edge in E has at least one endpoint in the vertex cover. The problem of finding a minimum vertex cover is to find a vertex cover of the smallest possible size. Formulate this problem as an integer linear programming problem.

Solution:

Let x_i be a binary variable with either 1 or 0 as its value. Here i belongs to V. If the vertex i is included in the vertex cover then the value of x_i will be 1 or else it will be 0.

Objective Function:

$min\ (\sum_{i \in V} x_i)$

***Constraints*:**

For each edge (u, v) ∈ E where u and v ∈ V

$x_u + x_v \geq 1$

This is because at least one vertex from the edge should be in the vertex cover.

$x_i \in \{0,1\}, \forall i, where\ i \in V$

Q7. Assume that you are given a polynomial time algorithm that given a 3- SAT instance decides in polynomial time if it has a satisfying assignment. Describe a polynomial time algorithm that finds a satisfying assignment (if it exists) to a given 3-SAT instance.

Solution:

Let $\phi(x_1, x_2, \ldots, x_n) = c_1 \wedge c_2 \wedge \ldots \wedge c_m$, where $c1, c2, \ldots, cm$ are the clauses and $x1, x2, \ldots, xn$ are the literals be the Boolean formula for a 3-SAT instance. Assume there exists a polynomial time algorithm S, which takes CNF-formula and outputs **true** if there is a satisfying assignment, else it outputs **false.**

We can find a satisfying assignment to ϕ by assuming that there is one. This can be confirmed by making a call to 3-SAT one. The assignment is found by iteratively finding a truth assignment b(1) for x_1, then finding an assignment b(2) for x_2, and so on until we have an assignment for all the variables. Here, each literal x_i is substituted by boolean values b(1).... b(i) ∈ {true, false}

To be specific, in S(($\phi(x_1, x_2, \ldots, x_n)$)), substitute the value of $x1$ = true, if the value of S(($\phi(1, x_2, \ldots, x_n)$)) = true, then we would assign the value of $x1$ as true. Else, we would assign the value of $x1$ as false. Once, the value of $x1$ is assigned, we will follow the same process for the next literal $x2$. For example, if the value of S(($\phi(1, x_3, \ldots, x_n)$)) = true, then $x2$ is true, otherwise $x2$ is false. We follow these steps for all the literals up till x_n.

This process would lead to n iterations with one call to S per iteration. For each iteration, finding the new formula to check if a CNF formula is satisfiable or not by plugging the value of a literal takes $O(m)$ time. Hence, we can find the assignment in polynomial time.

Q8. The graph five-coloring problem is stated as follows: Determine if the vertices of G can be colored using 5 colors such that no two adjacent vertices share the same color. Prove that the five-coloring problem is NP-complete.
Hint: You can assume that graph 3-coloring is NP-complete

Solution:

Firstly, to prove that the five-coloring problem is NP-complete we need to prove that the problem belongs to NP and then prove that it for its NP-Hardness

1. Let us prove that the five-coloring problem is NP:

Given a graph G(V, E) such that each node has a color, for each edge e(u, v) ∈ E, and u, v ∈ V, first we need to check if the color of both vertices u and v are different. Further, we would need to check if there exists five different colors in the graph. We will be able to verify the validity of a given graph in polynomial time. We can verify this by running BFS and checking whether two adjacent nodes have the same color. Moreover, since there are at most five colors possible, it can be verified in polynomial time.

2. Prove that the Five-coloring problem is NP-Hard

We will be able to do this by reducing as follows:

$$\mathbf{3-coloring} \leq_p \mathbf{5-coloring}$$

Construction:

Given a graph G, which is in 3-coloring, we need to create a new graph G', by adding two additional nodes x and y to G. Then, connect x and y to all the vertices present in G and also to each other.

Claim: G' has a 5-coloring solution if and only if Graph G has a 3-coloring.

Proof:

=> Given a valid 5 – coloring solution exists for graph G, we need to show that graph G' has a valid solution for 3-coloring. In graph G', as nodes x and y are connected to all other nodes in G and to each other, the 5-coloring solution must assign distinct colors to nodes x and y, such as yellow and black. Subsequently, the remaining three colors red, blue and green are utilized to color the remaining graph G, thereby establishing a valid 3-color solution. Moreover, if we end up removing the nodes x and y from G', G' is transformed back to G, which still gives us a valid solution for the 3-coloring problem.

<= If a valid 3-coloring scheme exists for graph G, with colors such as red, green, and blue, our objective is to establish that graph G' can be suitably colored using a 5-color scheme. To accomplish this, we initially color all nodes in G' except for x and y with the same colors as their counterparts in graph G. Subsequently, we choose two distinct colors, such as yellow and black, to color nodes x and y respectively. This ensures that nodes x and y acquire colors distinct from all other nodes in G'. By combining this method with the 3-coloring solution

applied to G, we can make use of a maximum of 5 colors to color G', ensuring that adjacent nodes do not share the same color.

Hence, with this we are successfully able to show that solving 5-coloring problem is at least as hard as solving 3-coloring problem, i.e. $3 - colouring \leq_p 5 - colouring$

5-coloring problem is NP-Complete.

Q9. Longest Path is the problem of deciding whether a graph G = (V, E) has a simple path of length greater or equal to a given number k. Prove that the Longest path Problem is NP-complete by reduction from the Hamiltonian Path problem.

Solution:

To prove that Longest Path problem is NP-complete we would need to prove that the problem belongs to NP and is NP-Hard

1. Longest Path problem is NP

Given a path of at least length k, for graph G = (V,E), we can traverse through the path and simultaneously keep track of the length and can easily verify that it is a simple path which is of length greater than or equal to k. This verification of validity can be done in polynomial time.

2. Longest Path problem is NP-Hard

This can be shown by reducing from the Hamiltonian Path problem. =>

$$\boldsymbol{Hamiltonian\ path} \leq_p \boldsymbol{Longest\ Path}.$$

Construct a graph G' such that all the vertices present in graph G is present in graph G' and for each edge (u, v) in graph G, there exists an edge between (u', v') of graph G'.

***Claim*:** There exists a simple path of length greater than or equal to k in Graph G' if and only if there exists a Hamiltonian path in graph G of length greater than or equal to k.

Proof:

=> If G' has a Longest Path of length at least k, where k = |V| -1, then all the edges corresponding to the Hamiltonian Path in G are definitely used. Hence, this gives us the Hamiltonian path in graph G.

<= If there exists an Hamiltonian Path in graph G, then the length of the path would be |V| - 1. If we trace this path in graph G', we would surely get the longest path in graph G' of at least size k where k = |V| - 1.

Since we are reducing our longest path problem from the Hamiltonian Path problem, it is acceptable that only a special instance for the Hamiltonian Path problem is getting handled i.e. the special instance in which k = |V| - 1. Hence, with this we are successfully able to show that solving Longest Path problem is as hard as solving Hamiltonian Path problem, i.e. $Hamiltonian\ path \leq_p Longest\ Path.$

Longest Path problem is NP-Complete

Q10. There are a set of courses in USC, each of them requiring a set of disjoint time intervals. For example, a course could require the time from 9am to 11am and 2pm to 3pm and 4pm to 5pm. You want to know, given a number K, if it's possible to take at least K courses. Since you want to study hard and take courses carefully, you can only take one course at any single point in time (i.e. any two courses you choose can't overlap). Show that the problem is NP-complete, which means that choosing courses is indeed a difficult thing in our life. Use a reduction from the Independent set problem.

Solution:

To show that the Class Scheduling problem is NP-complete we first need to prove that the problem belongs to NP and is NP-Hard

1. Class Scheduling problem is NP

Assume we are given a class schedule S. For each class present in S we check the scheduled time of the class. If we find that there is no overlap between the classes and the number of classes are greater than or equal to k in polynomial time then we can say that our class scheduling problem belongs to the NP set of problems.

2. Class Scheduling problem is NP-hard

We will be able to do this by reducing as follows:

$$\textit{Independent Set} \leq_p \textit{Class Scheduling problem}$$

Construction:

Construct a graph G such every class present in the schedule has a vertex of its own denoting the class. If two classes are having overlapping time intervals, we need to join their vertices with an edge and this edge would be marked with this overlapping time interval. In this constructed graph, we cannot choose any two vertices that are connected by an edge.

Claim:

In graph G there exists a possibility of taking K courses such that the course timings don't overlap if and only if the graph G has an yes instance of independent set problem.

Proof:

=> If in graph G, students can take k courses such that none of the course timings overlap, then this means that we have at least k nodes that aren't connected to each other via edges, and these vertices would form an Independent set.

<= If there exists an Independent set solution for the graph G whose size is greater than k, then no two vertices that are connected by an edge will be part of the independent set. Hence, when there are k independent nodes it means that there are K non overlapping courses, so the algorithm for course schedule will output 'Yes'.

Therefore, with this we can successfully show that solving Course Scheduling problem is as hard as solving Independent Set problem, $Independent\ Set\ \leq_p\ Class\ Scheduling\ problem$

Course scheduling problem is NP-Complete.