Final Exam Solutions

1 Chicago Seven / Conspiracy Eight (2×15 pts)

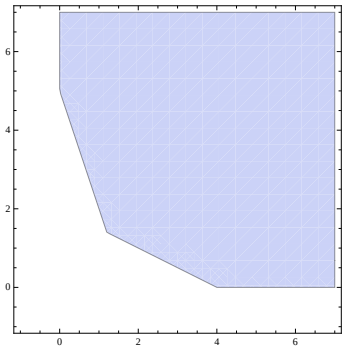
1.1 LP solve (15 pts)

Consider the following linear program:

$$\min x_1 + x_2$$
$$x_1 + 2x_2 \geq 4$$
$$3x_1 + x_2 \geq 5$$
$$x_1 \geq 0, \quad x_2 \geq 0$$

- (10 pts) Solve the linear program and find the:
  - $Value = \frac{13}{5}$
  - $(x_1, x_2) = \frac{1}{5}(6, 7)$

  Solution: Draw the region given by $x_1 \geq 0, x_2 \geq 0, x_1 + 2x_2 \geq 4, 3x_1 + x_2 \geq 5$. We see that the solution is bounded since $x_1 + x_2 \geq 0$ in the feasible region. The vertices of the region are the feasible points where two constraints intersect: $(0, 5), (4, 0)$ and $(6/5, 7/5)$. The values of the objective in these vertices are $5, 4$ and $13/5$ so the minimum is $13/5$.



- (5 pts) Write the dual of the linear program in the following form:

$$\begin{aligned} \min y^T b \\ y^T A \geq c^T \\ y \geq 0 \end{aligned}$$

$A = \begin{pmatrix} -1 & -2 \\ -3 & -1 \end{pmatrix}$
$b = (-4, -5)^T$

$c = (-1, -1)^T$

We first write the original linear program in standard form:

$$\max c^T x, \quad Ax \leq b, x \geq 0$$

So the dual will be

$$\min y^T b, \quad y^T A \geq c^T, \quad y \geq 0.$$

Since the original statement had $\min$ instead of $\max$ and $\geq$ constraints instead of $\leq$ we rewrite it:

$$\begin{aligned} \max \ -x_1 - x_2 \\ -x_1 - 2x_2 \leq -4 \\ -3x_1 - x_2 \leq -5 \\ x_1 \geq 0, \quad x_2 \geq 0 \end{aligned}$$

and see that

$$A = \begin{pmatrix} -1 & -2 \\ -3 & -1 \end{pmatrix}, \quad b = (-4, -5)^T, \quad c = (-1, -1)^T$$

1.2 Time to Play (15 pts)

Given an undirected, unweighted graph, with each node having a certain value, consider the following game.

- All nodes are initially *unmarked* and your score is 0.
- First, you choose an unmarked node u. You look at the neighbors of u and add to your score the sum of the values of the *marked* neighbors v of u.
- Then, mark u.
- You repeat the last two steps for as many turns as you like (you *do not* have to mark all the nodes. Each node can be marked at most once).

For instance, suppose we had the graph $A - B - C$ with A, B, C having values $3, 2, 3$ respectively. Then, the optimal strategy is to mark A then C then B giving you a score of $0 + 0 + 6$. We can check that no other order will give us a better score.

- (5 pts) Suppose all the node values are nonnegative. Give an efficient algorithm to determine the order to mark nodes to maximize your score. Justify your answer.

  Solution: We claim that sorting the nodes by decreasing value (breaking ties arbitrarily), and taking them in that order will lead to an optimal solution. We show this by an exchange argument. For any other ordering π that isn't sorted, we can find an i such that $l(\pi_i) < l(\pi_{i+1})$. Now, suppose we swapped them. We have two cases:

  - Case 1: π_i and π_{i+1} do not have an edge between them: In this case, swapping these two nodes will not decrease our score, since these are unrelated nodes.
  - Case 2: π_i and π_{i+1} do have an edge between them: We can ignore all neighboring nodes, since we will mark these two nodes eventually. However, notice that it is optimal to mark the higher valued node first, then the lower valued one, thus, swapping these will give us a strictly better score.

  Thus, by the exchange argument, sorting the nodes will give us an optimal solution.

  Alternate solution: We show the claim by this simple proof. Notice that for each edge (u, v), either u gets marked first or v gets marked first, thus an upper bound on the contribution from this edge is $\max(l(u), l(v))$. But, our algorithm achieves this upper bound, thus must be maximal.

  Alternate solution 2:

  Comments: Notice that this solution does not depend on the graph structure at all. Also, from the above argument, we can notice that if we wanted to calculate the actual score, we can do this in $O(E)$ time by computing $\sum_{(u,v)=e\in E} \max(l(u), l(v))$ (as opposed to following the algorithm above). Also, on correctness: Some optimal orderings are not sorted. Thus, if the list isn't sorted, it is not true that the score of this list will be strictly smaller. Also, some were doing exchanges on two nodes connected by an edge. This is a bit harder to show, and many people did not get the details of this completely. Some students did a BFS/DFS to first extract node values. This is unnecessary, since it has already been given to you in the input.

- (10 pts) Now, node values can be negative. Show this problem is NP-hard by giving a reduction from INDEPENDENT SET.

  Solution: (First, a quick comment, the original problem phrased is a search problem, but we should change it to a decision problem before starting. Sorry for not making this clarification during the test, but it seems most students who were able to get the solution were able to get around this detail.)

  An instance of independent set consists of a graph $G = (V, E)$ and a number b, which means we wish to find an independent set of size at least b. We transform it to this problem as follows: for each of the original nodes in the graph, assign it a value of -2. Add a "super-node" s and connect it to everything

else, and assign it a value of 1. We also set our target value to be b. We now claim that our score directly corresponds to maximizing the size of an independent set

- Solution to independent set to solution to graph marking: If there exists an independent set of size b, then our strategy for graph marking would be to mark s, then the nodes in the independent set, giving us a score of b.
- Solution to graph marking to solution to independent set: First, notice that it is always optimal to mark s first regardless. Now, suppose we've gotten a score of b. Now, if we have chosen any node that neighbors a marked node that is not s, we will receive a net score of -1, which is clearly not optimal. Thus, we can see that graph marking will never choose any two nodes that are adjacent. Thus, this directly corresponds to an independent set of size b.

Thus, we've completed both directions, so we've shown that graph marking is NP-hard.

Comments:

- Make sure your reduction is in the correct direction (some students were arguing that if you know how to solve ind. set you get a good algorithm for the marking problem. This is the wrong direction!)
- As part of your reduction, you are creating an instance of graph marking from an instance of independent set. This means, there are no given node values: you get to choose them yourselves.
- The -2 may seem a bit arbitrary, but if you choose -1, you may not necessarily get independent nodes (since adding neighbors gives you zero net gain). You need to deal with these cases explicitly if you chose to go this route. Alternatively, any value that is strictly less than -1 will work.
- You can't assume that graph marking will choose certain types of optimal solutions. All you can assume is that graph marking will return a solution with the maximum score. For instance, a common mistake is to say that graph marking will return the solution that first maximizes score, breaking ties by how many nodes are marked. This is something that was not in the problem statement itself, so unless you've shown that this is equivalent to simply maximizing score, this is not a valid way to go about this problem. Thus, simply assigning a large negative number to every node and no supernode is not a valid solution.

2 Short Questions (5 points each)

1. Does 5 have a multiplicative inverse modulo 111? If not, prove that there is no inverse. If there is an inverse, find it.

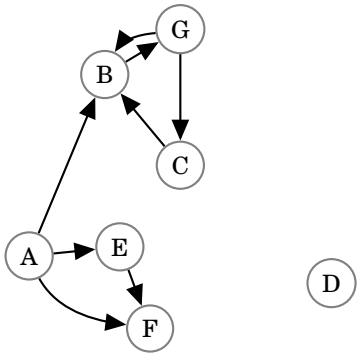
   Solution: Notice that $5 \cdot 22 \equiv -1 \pmod{111}$, so $5 \cdot -22 \equiv 1 \pmod{111}$, and $-22 \equiv \boxed{89} \pmod{111}$

2. Show that if $n \times n$ matrices can be squared in time $O(n^c)$, then any two $n \times n$ matrices can be multiplied in time $O(n^c)$.

   Solution: (short comment: it is implied that $c \geq 2$, since otherwise, we wouldn't be able to read in our input). We show that we can reduce multiplying to squaring. Given two $n \times n$ matrices X, Y, construct a $2n \times 2n$ matrix $A = \begin{pmatrix} X & Y \\ 0 & 0 \end{pmatrix}$. This will take $(2n)^2$ time. Now, from the statement, we can square A in time $O((2n)^c)$. Squaring A gives us the $2n \times 2n$ matrix $\begin{pmatrix} X^2 & XY \\ 0 & 0 \end{pmatrix}$, from which we can extract the solution to the original problem. Thus, we can do multiplication in $O(n^2 + (2n)^c) = O(n^c)$ time.

   Comments: This was a reduction problem in disguise. One common mistake was to assume that matrix multiplication is commutative (we can not say that $AB = BA$ for general matrices), to get the faulty reduction of $AB = ((A+B)^2 - A^2 - B^2)/2$ or some variant of it.

3. Partition the following directed graph into strongly connected components.

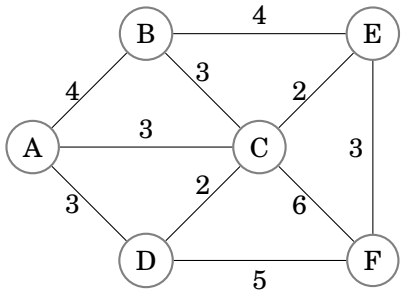


   Solution: The strongly connected components are: $\{A\}, \{B, C, G\}, \{D\}, \{E\}, \{F\}$

4. Given an **unweighted** undirected graph $G = (V, E)$ with $|V| = n$ and $|E| = m$, give an $O(n(n+m))$ algorithm to compute *all* pairwise graph distances (shortest path lengths measured by number of edges) between every pair of nodes.
   Solution: For each node u run BFS from the node u to compute the distances $d(u, v)$ for all $v \in V$. Since the running time of BFS is $O(n+m)$, the total running time of the algorithms is $O(n(n+m))$.

   Comments: Using DP, Belman-Ford (from every point) or Floyd-Marshal are all slower ; in particular, there is no faster parallel implementation of Belman-Ford that runs in time $O(n(n+m))$).

5. What is a minimal spanning tree of the following weighted undirected graph? What is its weight?



Solution: The MST consists of the following edges: { AC, BC, CD, CE, EF } or { AD, BC, CD, CE, EF }. The weight is 13.

6. Given a sequence of n integers $a_1, \ldots, a_n$ give an $O(n^2)$ algorithm to find the longest increasing-then-decreasing subsequence. For example for the sequence $1, 8, 3, 5, 9, 6, 7, 11, 6, 4, 8, 2, 1$ the longest increasing-then-decreasing sequence is $1, 3, 5, 6, 7, 11, 6, 4, 2, 1$.

   Solution: Run the longest increasing subsequence as described in a book to get $L(i)$ which represents the longest increasing subsequence using $a_1 \ldots a_i$ including element a_i. Now, we can symmetrically run this to get the longest decreasing subsequence $D(i)$ which represents the longest decreasing subsequence from $a_i \ldots a_N$ including element a_i. Then, we take $\max_{1 \leq i \leq N} L(i) + D(i) - 1$ to get the biggest length. The runtime of this is 2 runs of longest increasing subsequence each of which takes n^2 time, and one more linear pass which takes $O(n)$ time. Thus, the running time is $O(n^2)$. Correctness follows frmo the fact that there must be a "peak" to our sequence, and we are trying every one of them.

7. We want to sort n distinct items where comparisons may be faulty, that is for elements i, j, we may have the wrong answer to the query $a[i] < a[j]$. The goal is to find an order $a[1], a[2], \ldots, a[n]$ that maximize the number of $i < j$ with $a[i] < a[j]$. Give a simple deterministic $O(n^2)$ algorithm that will achieve an approximation ratio of $\frac{1}{2}$.

   Solution: First, start with the order given by the input and compute the score. If this score is less than $\frac{\binom{n}{2}}{2}$, then reverse the order. Then return whatever sequence we have. We now show this is correct. For any ordering, let its score be k. Then, the reversed ordering will have score $\binom{n}{2} - k$. Thus, we have $\max(k, \binom{n}{2} - k) \geq \frac{1}{2}\binom{n}{2}$, and since $\binom{n}{2} \geq OPT$,w e have $\max(k, \binom{n}{2} - k) \geq \frac{1}{2} OPT$, which is what we wanted. Computing the score takes $O(n^2)$ time.

8. If we measure the qubit $\frac{1}{\sqrt{2}} |0\rangle - \frac{i}{\sqrt{2}} |1\rangle$, what is the chance that the measurement will be 0?

   Solution: We can first optionally check that this is a valid qubit. We just need to check $|\frac{1}{\sqrt{2}}|^2 + |-\frac{i}{\sqrt{2}}|^2 = 1$. The chance that the measurement is zero is $|\frac{1}{\sqrt{2}}|^2 = \frac{1}{2}$.

3 (True $\vee$ False) (2×15 pts)

No need to justify. Mark the correct answer (if both TRUE and FALSE are marked the answer is incorrect)

1. In the range $[1, 10^{10}]$, there are more Carmichael numbers than there are prime numbers.
   FALSE: As stated in the book, Carmichael numbers are very rare. There are 1547 Carmichael numbers in this range, while the number of primes in this range is somewhere around 50 million.

2. The recurrence relation $T(n) = n^2 + T(\lfloor\sqrt{n}\rfloor), T(1) = 1$ satisfies $T(n) = O(n^2 \log n)$.
   TRUE: Since $\lfloor\sqrt{n}\rfloor < n$, we see that $T(n) \leq n^2 \times n = n^3$ and then $T(n) = n^2 + T(\lfloor\sqrt{n}\rfloor) \leq n^2 + \lfloor\sqrt{n}\rfloor^3 \leq n^2 + n^{3/2} \leq 2n^2$. So $T(n) = O(n^2) = O(n^2 \log n)$.

3. If u and v are two vertices of a graph, then running DFS on the graph could never result in $pre(u) < pre(v) < post(u) < post(v)$.
   TRUE: You must always finish visiting a child completely before you can return to the parent.

4. Let $G = (V, E)$ be a directed graph where the only edges with negative weights are outgoing edges of a vertex $s \in V$. Running Dijkstra's algorithm on G starting at s will give the correct shortest path lengths from s to all other vertices in G.
   TRUE: This problem should have said that there are no negative cycles (so everybody got the 2 points for this question)

5. Every connected weighted graph has a unique minimal spanning tree.
   FALSE: We don't have unique edge weights, e.g. in the cycle of length n with edge weights 1 there are n different MSTS.

6. Consider an instance of the knapsack problem *without replacement* with 8 items whose weight and values are
$$(1kg, 1\$), (2kg, 4\$), (2kg, 3\$), (3kg, 6\$), (5kg, 5\$), (5kg, 3\$), (7kg, 3\$), (9kg, 9\$)$$
with total weight $W = 10kg$ has optimal value $15\$$.
TRUE: Take items 2, 4, 5.

7. Let $G = (V, E)$ be a directed graph with positive edge weights, with edge $(u, v) \in E$. During a full run of the Ford-Fulkerson maximum flow algorithm, the residual edge (v, u) can be added to the graph no more than $|V| + |E|$ times.
FALSE: As we saw on one of the homework assignments, the number of times a residual edge could be added to a graph might depend on the maximum flow of the graph.

8. Given a graph $G = (V, E)$ with integer edge weight values (positive, zero, and/or negative), finding the shortest simple path (a simple path does not revisit any vertices) from a vertex $s \in V$ to $t \in V$ is NP-hard.
TRUE: We do a reduction from longest path. Given an instance of longest path, negate all the edges. We can see that the shortest path in this negated path is the same as the longest path in the original graph.

9. Given any instance of the set cover problem, a greedy algorithm that chooses an unchosen set with the largest number of uncovered elements at each iteration will always produce a solution consisting of at most twice as many sets as the optimal solution.
FALSE: As discussed in the book, we can find a solution that is $O(\log n)$ as much as the optimal solution. We have shown in section that this bound is tight.

10. RSA would be secure against quantum computers (if such computers could be built).
FALSE: Quantum computers can factor numbers using Shor's algorithm in polynomial time. RSA's security relies on the hardness of factoring.

11. If we strictly increase the capacity of all edges in *one* min $s - t$ cut, then we will strictly increase the capacity of the maximum $s - t$ flow (you can can assume all capacities are integer).
FALSE: Consider the graph $A \to B \to C$ with all capacities 1. Then, one min cut is $A \to B$, but increasing this edge will not increase our max flow.

12. Given that there is one unique maximum $s - t$ flow in a graph G, there is one unique minimum $s - t$ cut in G.
FALSE: Consider the graph above. It has one maximum flow, but two minimum cuts.

13. If an edge e is not used in any max $s - t$ flow, then that edge will not be in any min $s - t$ cut.
TRUE: This follows from complementary slackness.

14. Finding a starting vertex for a linear program to run simplex can be solved by running simplex on a modified linear program.
TRUE: This was discussed in class and demonstrated on the last question of the discussion 9 notes.

15. Suppose that problem A is NP-hard and has a known polynomial approximation algorithm with an approximation ratio of 2. Then, for all problems B which reduce to A, we can construct a polynomial approximation algorithm for problem B with approximation ratio at most 2.
FALSE: This problem should have stated that we are assuming $P \neq NP$. Assuming $NP \neq P$, this is false, since there exists different NP-complete problems with different approximation ratios: some have cannot be approximated at all and others can be approximated within 1.1. (Since we didn't state that we assume $P \neq NP$, every received points for this question)