CS170 Midterm 2 – Solutions

Problem 1

1. *(4 points)* In the following instance of SET COVER, which sets will be returned by the greedy algorithm? (Break ties however you like.) What is the optimum?

   Solution: $S_1 = \{D, C\}, S_2 = \{D, E, F\}, S_3 = \{A, B, D, E\}, S_4 = \{A, B, C\}, S_5 = \{A, F\}$.

   The greedy solution can be any of the following : (S_3, S_1, S_2), (S_3, S_1, S_5), (S_3, S_4, S_5), (S_3, S_2, S_4).

   The optimum solution is (S_2, S_4).

   Grading: 1 mark was for the optimum. For the greedy part, if you chose S_3 you got 1 mark (even if you subsequently made a mistake). If you did not choose S_3 but returned a set-cover, you got 1 mark.

2. *(10 points)* Consider the dynamic programming algorithms familiar from the textbook and the lectures for solving these problems:

   - E: Edit distance of two strings
   - K: Knapsack (without repetitions)
   - C: Chain matrix multiplication
   - T: Traveling salesman problem
   - I: Independent set on a tree
   - F: Floyd-Warshall

   Consider the DAG of subproblems where the orientation of the edges is from "smaller" subproblems to "bigger" ones. For which of these algorithms the DAG of the subproblems:

   (a) has all in-degrees at most two: K
   (b) has all in-degrees bounded from above by some constant independent of the instance : E,F,K
   (c) "is like a square grid with diagonals" : E
   (d) "looks like a pyramid" : C
   (e) is exponential in the length of the input : K,T
   (f) the algorithm runs in linear time : I
   (g) the algorithm runs in $\Theta(n^3)$ time : C,F

   For each property there may be none or more than one algorithms, make sure you write them all next to the statement of the property.

   Grading: 3 marks were there for (a) and (b). 2 marks for (c) and (d). 5 marks for (e), (f) and (g). In each of these parts, depending upon the "symmetric difference" between the correct answer and your answer, you got some marks.

3. *(4 points)* Transform the following linear program, called LP1, into one in the form: minimize $\mathbf{c} \cdot \mathbf{x}$ subject to $\mathbf{Ax} = \mathbf{b}$, $\mathbf{x} \geq \mathbf{0}$:

   maximize $x_1 + x_2 - x_3$
   subject to $x_1 - x_2 = 3$
   $x_2 + x_3 \leq 1$
   $x_2, x_3 \geq 0$.

   Idea: The form means that (1) the objective function is a minimization; (2) all contraints are equality; and (3) all variables are non-negative. To achieve (1), we negate (multiply by minus one) the objective function. To achieve (2), we introduce a slack variable $s \geq 0$ to turn the inequality constraint an equality. To achieve (3), we introduce new varibles $x_1^+ \geq 0$ and $x_1^- \geq 0$, and replace every occurence of the uncontrained variable x_1 with $x_1^+ - x_1^-$.

Solution: minimize $-x_1^+ + x_1^- - x_2 + x_3$
subject to $x_1^+ - x_1^- - x_2 = 3$
$x_2 + x_3 + s = 1$
$x_1^+, x_1^-, x_2, x_3, s \geq 0.$

Grading: One mark each for demonstrating (1), (2) and (3). One mark for correctly combining the above ideas into a valid linear programme of the required form.

4. *(4 points)* Does LP1 have an optimum? If so, what are the optimum values of x_1, x_2, x_3? (No need to show your work, but explain briefly.)

   Solution: LP1 has an optimium, because (1) LP1 is feasible with $(x_1, x_2, x_3) = (3, 0, 0)$; and (2) the feasible region is bounded between $0 \leq x_2 \leq 1$ and $0 \leq x_3 \leq 1$ and $3 \leq x_1 \leq 4$ (recall $x_1 = x_2 + 3$). The optimum values are $(x_1, x_2, x_3) = (4, 1, 0)$, giving an objective value of 5.

   Grading: Two marks for stating that LP1 has an optimum (better with a valid explanation). Two marks for the correct optimum values of (x_1, x_2, x_3).

5. *(4 points)* Write the dual of the linear program LP1 given above (not the transformed one). Use variable names $y_1, \ldots$.

   Idea: Multiply the first constraint $x_1 - x_2 = 3$ by y_1, which is not sign-constrained because the constraint is an equality, to get $y_1x_1 - y_1x_2 = 3y_1$. Multiply the second constraint $x_2 + x_3 \leq 1$ by $y_2 \geq 0$, which is non-negative because the constraint is an inequality, to get $y_2x_2 + y_2x_3 \leq y_2$. Add them to get $(y_1)x_1 + (-y_1 + y_2)x_2 + (y_2)x_3 \leq 3y_1 + y_2$. Now enforce constraints (1) $y_1 = 1$, an equality because x_1 is not sign-constrained; (2) $-y_1 + y_2 \geq 1$, an inequality because x_2 is non-negative; and (3) $y_2 \geq -1$, an inequality because x_3 is non-negative. The objective value is to minimize $3y_1 + y_2$.

   Solution: minimize $3y_1 + y_2$
   subject to $y_1 = 1$
   $-y_1 + y_2 \geq 1$
   $y_2 \geq -1$
   $y_2 \geq 0.$

   Grading: One mark for the objective function, one mark for demonstrating how to handle a inequality primal constraint with a non-negative dual variable (or vice versa), one mark for demonstrating how to handle an equality primal constraint with an unconstrained dual variable (or vice versa), and one mark for demonstrating full understanding of duality.

   It is okay to simplify the dual linear programme (e.g. by crossing out $y_2 \geq -1$, which is implied by $y_2 \geq 0$; or by crossing out $y_2 \geq 0$, which is implied by $y_2 \geq 2$ as $y_1 = 1$ and $y_2 \geq 1 + y_1$), but a mark is deducted for enforcing $y_1 \geq 0$, which despite giving an equivalent linear programmme, demonstrates a lack of understanding that y_1 should not be sign-constrained in the formal dual linear programme.

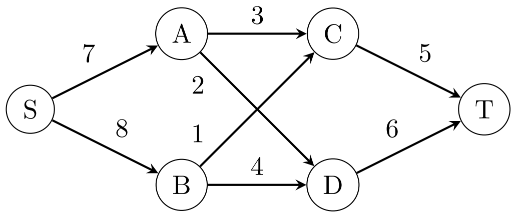
6. *(4 points)* Does the dual of the LP1 have an optimum? If so, give the optimum values of the y_i's (no need to show your work) and show that the equation predicted by duality holds.

   Solution: The dual has an optimum, because (1) the primal has one and (2) as a result of the strong duality. The optimum values are $(y_1, y_2) = (1, 2)$. The equation predicted by the strong duality is

   $$\text{optimal value of primal} = 4 + 1 - 0 = 5 = 3(1) + 2 = \text{optimal value of dual.}$$

   Grading: One mark for stating that the dual has an optimum (better with a valid explanation), one mark for the optimal values of (y_1, y_2), and two marks for the equation of strong duality. One mark is deducted for demonstrating only the weak duality (an inequality).

7. *(4 points)* What is a maximum flow and a minimum cut in the network below? (No need to show your work.) How do you know that this is a maximum flow and a minimum cut?



Solution: The maxflow routes 5 units of flow along each of $S \to A$ and $S \to B$. Then, it routes 3 units along $A \to C$, 2 units along $A \to D$, 1 unit along B to C and 4 along B to D. Finally, 6 units are routed along $D \to T$ and 4 units along $C \to T$.

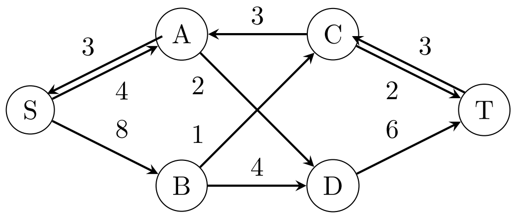
A min-cut was either the one separating $\{S, A, B\}$ and $\{C, D, T\}$ or the one separating $\{C, T\}$ from $\{S, A, B, D\}$.

The answer to the last question is any of the following: By duality, we know max-flow $\leq$ min-cut and we have exhibited a flow and a cut of the same value 10. Another reason which is acceptable is that there is no path left in the residual graph to go from S to T (which proves the max-flow min-cut theorem).

Grading: 1 mark was deducted if you did not specify the max flow. 1 mark was deducted if you gave a wrong value for the max-flow. 1 mark was deducted if you indicated a cut other than one of the min-cuts. 1/2 or 1 mark (depending on how vague you were) was deducted if you failed to assign any reason as to how you know that the flow you indicated is the max-flow.

8. (*4 points)* In the same network, perform the first iteration of the max flow algorithm: Find a path from S to T, increase the flow by the amount allowed, and construct the residual network.

   Solution: *There are many possible answers. For example:*



   Grading: 1 mark was deducted if you did not indicate the back edges and 1 mark was deducted if you drew the final residual network after the entire max-flow is routed. 1 mark was also deducted if instead of routing the maximum flow, you routed just a unit of flow along a path (unless, of course the capacity of the path was 1).

9. (*4 points)* Given a graph with edge weights (which can be negative) and a vertex v in it, how can you tell whether there is a negative cycle through v by running a slight modification of a shortest path algorithm? (A very brief statement suffices here.)

   Solution:

   Run Bellman-Ford (the shortest path algorithm that works with negative edges – this is very important) rooted at node v. Remember that the loop in Bellman-Ford updates distances based on all edges $|V| - 1$ times. Run the loop one additional time. If dist(v) changes from iteration $|V| - 1$ to $|V|$, then there is a negative cycle through v.

   Grading:

   4 points for the above solution and related solutions. -2 points for just saying run Bellman-Ford and checking to see if any edges update with an additional iteration through the loop. The issue here is that you aren't necessarily finding negative cycles through v if you do this. You are just finding *a* negative cycle in the graph. You must root at v and check if the value of v updates. We also deducted varying numbers of points if you were vague in your description, did not specify the shortest path algorithm used (often saying, "run *the* shortest path algorith") since it is very important to remember that not all algorithms work with negative edges, or were incorrect in any other way.

10. (*4 points)* What is the value of this game? (Justify briefly.)

$$\begin{matrix} 1 & 0 \\ 0 & 1 \end{matrix}$$

Solution:

Value: 1/2. Justification: A randomized algorithm for each player selecting each move with probability 1/2 achieves this score. If either player wavers from this strategy, the other one can exploit it. Therefore, by the min-max theorem, this must be the optimal strategy and 1/2 must be the value of the game.

Grading:

Two points for the value, two points more for the justification if the value was correct. Please note that this is a zero-sum game which means that if player one loses one, then player two wins one. It does not mean the value is zero.

Dynamic Programming

(30 points) Your company, Stuckbars Coffee and Toffee, is planning to open several stores on Main Street. Main Street has n blocks $1, 2, \ldots, n$ from East to West, and for each block i you know the revenue $r_i > 0$ you expect from a store on this block. But if you open a store on block i, you cannot open another store in block i, or the two blocks to its East or the two blocks to its West. You have designed a dynamic programming algorithm for finding the store locations that maximize total revenue, but you spilled coffee on it and now you can't read some parts. Fill the blanks:

Choice of subproblem: $R[i]$ = **the maximum total**

- $\boxed{\text{revenue if stores can only be opened on blocks } 1, \ldots, i}$

Other subproblems might work too; for example, one could add to the above subproblem the condition that a store must be built at block i.

Grading: 5 points for a precise description of a subproblem where "larger" subproblems can be efficiently computed based on "smaller" ones, for a some simple ordering of subproblems. Minus one or more points if the description is unclear, e.g. "maximum total up to block i" without saying the word "revenue".

$R[1] = \boxed{r_1}$ $R[2] = \boxed{\max\{r_1, r_2\}}$

Your base cases should match the subproblem you chose; these are the base cases for the subproblem listed above. (Note: most solutions to this problem require a third base case $R[0] = 0$ or $R[3] = \max\{r_1, r_2, r_3\}$. This was an error on our part, and we ignored the issue when grading.)

For $i = 3, \ldots n, R[i] = \boxed{\max\{R[i-1], r_i + R[i-3]\}}$

Grading: 15 points. Your base cases and recurrence should match the subproblem you chose; we have listed the correct solutions for the example subproblem given above. One or two points were taken off for minor errors in the base cases or the recurrence: for example, writing $r_i + R[i-2]$ instead of $r_i + R[i-3]$. Subproblems with more significant errors scored lower; partial marks were given in some cases especially if the subproblem dependencies didn't have any cycles.

Now add a line that will help you recover the optimum solution.

$$\boxed{prev[i] := \begin{cases} i-3, & \text{if } R[i] = r_i + R[i-3] \\ i-1 & \text{otherwise} \end{cases}}$$

Grading: 5 points. Marks were given for a line (or other modification) that recorded whether the optimal solution to the subproblem $R[i]$ includes the store at block i. The idea is that your modification should help recover the solution (set of store positions) at the end. No marks were given for finding just the value of the solution.

What is the running time of your algorithm? Justify briefly.

$\boxed{O(n)\text{, since each subproblem takes constant time to compute based on smaller subproblems, and there are } n \text{ subproblems to compute.}}$

Grading: 5 points. Your answer should depend on the recurrence you wrote. At least 4/5 were given for a correct answer, and the final mark was given for justifying your answer by giving either the running time of each subproblem computation or the total number of subproblems to compute.

Suppose now that you are given not one number r_i for each block, but several numbers r_{ij}, standing for the expected revenue if the closest other store is j blocks away (for example, r_{in} is the revenue if this is the only store in the street). How can you solve this version of the problem, and how much time is required? If you want, you can give a very high level description, or just one equation.

Solution:

We will define a subproblem with two parameters. $S[i,j]$ will be the maximum revenue that can be achieved with stores only on blocks 1 through i, with a slight change: the revenue from the furthest-East store will be $r_{k\ell}$, where k is the position of the store, and ℓ is the minumum of the following two numbers: first, the distance to the closest other store, and second, $|j-k|$. The intuition is that we imagine a rival store at position ℓ from which we do not derive profit, but which still counts as a store when deciding which revenue $r_{k\ell}$ we get. Here is a recurrence for this subproblem; we will skip the base cases:

$$S[i,j] = \max\{r_{ij} + S[i-j,j], S[i,j-1], S[i-1,j+1]\}.$$

The first choice represents building a store at location i and requiring the next store to the East to be at least j units away. The second choice represents deciding to reduce the considered distance to the nearest store. The third choice represents choosing to not build a store at location i.

Grading:

This was a bonus question, so grading was strict. Five points were awarded for a correct solution that was precisely stated, meaning a very precise description of the subproblem, or the recurrence relating the subproblems. Incorrect or imprecisely stated solutions were awarded zero points.

Now you want to repeat the original problem (with the r_i's) in a city whose Main Street branches like a tree. *Very briefly,* how would you go about doing that?

Solution:

Assume the tree is rooted at block v. For every node u and integer $d \in \{0,1,2\}$, let $R[u,d]$ be the revenue that can be gathered from the subtree rooted at u if the first d levels are required to be empty. We have the following recurrence, stated separately for each value of d:

$$\begin{aligned}
R[u,0] &= \max\{r_u + \sum_{c \in \text{children}(u)} R[c,2], R[u,1]\} \\
R[u,1] &= \max_{c \in \text{children}(u)} R[c,0] + \sum_{\substack{d \in \text{children}(u) \\ d \neq c}} R[d,1] \\
R[u,2] &= \sum_{c \in \text{children}(u)} R[c,1].
\end{aligned}$$

(In the second line of the recurrence, notice that at most one child of the root is allowed to have a store; we've called that child c and indexed all the other children by d.)

Grading:

This was a bonus question, so grading was strict. Five points were awarded for a correct solution that was precisely stated, meaning a very precise description of the subproblem, or the recurrence relating the subproblems. Incorrect or imprecisely stated solutions were awarded zero points.