CSE 101 Homework 4

Winter 2023

This homework is due on gradescope Friday February 24th at 11:59pm on gradescope. Remember to justify your work even if the problem does not explicitly say so. Writing your solutions in LaTeXis recommend though not required.

Question 1 (Gameshow Strategy, 75 points)**.** *Dirk is participating in a gameshow. In this show there are n different challenges. For the i^{th} challenge, Dirk estimates a probability p_i that he can pass the challenge (and this is independent of which other challenges he may have done up to this point or their results) and there is a reward R_i that he will receive if he successfully completes it. Dirk may attempt the challenges in any order he chooses (though may not try any particular challenge more than once), but once he fails a challenge, he must stop and leave with any award money he has won thus far.*

(a) Give an $O(n \log(n))$ time algorithm that given the p_i and R_i determines the best order for Dirk to attempt these challenges in. Hint: For any two challenges A and B, if Dirk is doing one of them right after the other, which would he rather do first? [40 points]

(b) After a rules change, Dirk is allowed to repeat challenges, but is not allowed to compete in more than k of them total, even if he completes all of them successfully. Give an algorithm that given k, p_i and R_i, runs in $O(kn)$ time and determines which challenges Dirk should attempt in which order so that he maximizes his expected payout. Hint: The answer is not always to do the same challenge k times. Try starting with the last challenge to attempt. [35 points]

(a) **Algorithm**: For the i^{th} game, we calculated a score $R_i \cdot \frac{P_i}{1-P_i}$ (∞ if dividing by 0). Then, sort in descending value and return the sorted order.

Time Complexity: Calculating the score for each game takes $O(1)$ and sorting takes $O(n \log n)$. Thus, the final runtime is $O(n \log n)$

Proof of Correctness: Sorting by $R_i \cdot \frac{P_i}{1-P_i}$ gives the same order as if we select the highest $R_i \cdot \frac{P_i}{1-P_i}$ as our greedy strategy. Thus, we can show the correctness of the sorted order by showing the correctness of choosing order with greedy strategy. Using the exchange argument, we wish to show that if we have a solution that agrees with the first k decisions, then we can make it agree with the first $k+1$ decisions without making it worse.

We first make the observation that if we have two games A and B, then $P_A(R_A + P_B R_B) \geq P_B(R_B + P_A R_A)$ if and only if $R_A \cdot \frac{P_A}{1-P_A} \geq R_B \cdot \frac{P_B}{1-P_B}$. Also, we make the observation that the expected value of our order can be written as $\sum_{i=1}^{n} R_i(\prod_{j=1}^{i} P_j)$, where the each reward R_i happens with the probability of every game up to the i^{th} game is won. Since the games are independent, the probability of that is simply the product of those games' probability.

Denote the solution that agrees with the greedy solution to the k^{th} decision as S_k. Call the greedy choice when choosing the $k+1^{th}$ game A. Assume the $k+1^{th}$ choice of S_k is not game A. If we only look at where game A is chosen in S_k and one game before game A, denoting that as game B. Assume A is put at the l^{th} place in S_k. Then, the terms in the expected value for involving R_l and R_{l-1} are

$$
\begin{aligned}
& R_{l-1} \prod_{j=1}^{l-1} P_j + R_l \prod_{j=1}^{l} P_j \\
=& (\prod_{j=1}^{l-2} P_j) \cdot (P_{l-1} R_{l-1} + P_{l-1} P_l R_l) \\
=& (\prod_{j=1}^{l-2} P_j) \cdot (P_B R_B + P_B P_A R_A) \\
=& (\prod_{j=1}^{l-2} P_j) \cdot (P_B(R_B + P_A R_A))
\end{aligned}
$$

If we swap A and B, we wouldn't affect any term that comes before or after those two term, and we then have those two terms as:

$$
=(\prod_{j=1}^{l-2} P_j) \cdot (P_A(R_A + P_B R_B))
$$

By the nature of greedy choice A, we have that $R_A \cdot \frac{P_A}{1-P_A} \geq R_B \cdot \frac{P_B}{1-P_B}$ and thus $P_A(R_A + P_B R_B) \geq P_B(R_B + P_A R_A)$. Hence, making this swap will not make the solution any worse. We continue to make swaps like this between two neighbors until game A is put to the $k+1$ spot. This shows that A_{k+1} is at least as good as A_k thus completing our exchange argument.

(b) **Algorithm**: Let E_i denote the maximum expected value of playing $k-(i-1)$ games and D_i denote the i^{th} game Dirk should play. For example, D_k would be the last game Dirk wants to play and E_k is the best expected value of playing $k-(k-1)=1$ game. Compute $E_k = max_{g\in[1..n]} P_g R_g$. For $i\ in\ k-1, ..., 1$, we compute $E_i = max_{g\in[1..n]} P_g(R_g + E_{i+1})$. At each computation, we make $D_i = g$ where g is the game that maximized the expression.

Time Complexity: Calculating each E_i takes $O(n)$ time since we have to find maximum out of n games. We have to calculate E_i a total of k times. Thus, the total runtime is $O(kn)$

Proof of Correctness: We wish to prove that E_i is correctly computed as the maximum possible expected value Dirk can get by playing $k-(i-1)$ games. Hence, we induct on i in descending order with the base case of $i=k$.

- **Base case:** E_k is the maximum possible expected value if Dirk only play 1 game. Hence playing the one with the highest expected value gives, well, the highest expected value.
- **Inductive hypothesis:** Assume E_l correctly computes the maximum possible expected value Dirk can get by playing $k-(l-1)$ games.
- **Inductive step:** Consider E_{l-1}. If the game g is won, then Dirk would gain R_g dollars and face the rest of $k-(l-1)$ games where the maximum possible expected winning is E_l. Since E_l is correctly calculated by our inductive hypothesis, E_{l-1} is also correctly computed as the maximum possible expected value Dirk can get by playing $k-(l-2)$ games.

Using induction, we know that the case $i=1$ will be correctly computed. Thus, we have correctly calculated the maximum possible expected value Dirk can get by playing $k-(1-1)=k$ games. Because the optimal values are correct each step, the choices documented to get those values must also the optimal.

Question 2 (Missing Minimum Weight Edge, 25 points)**.** *Let G be a weighted graph and e an edge of G of minimum weight. Show that there is a minimum spanning tree of G that does not contain e if and only if there is a cycle C in G containing e consisting only of minimum weight edges.*

The "if" direction:

- Assume there is a cycle C in G containing $e=(u,v)$ consisting only of minimum weight edges. Let T be a minimum spanning tree of G. If T does not contain e, we are done. Assume T contains e. Then, let $T' = T-e$. Then, T' will have two connected components since every edge is a cut-edge in a tree. There must be another edge in the cycle C across these two connected component. This could be shown by walking along the cycle from u to v without using e. Since u and v are in two different connected components, there has to be an edge crossing the two in order to get from u to v. Add that edge e' in and we have connected the two connected component. Thus, the $T'+e'$ is a spanning tree. Because every edge in C is a minimum edge, $w(e)=w(e') \Rightarrow T'+e'$ has same weight as T. Thus, $T'+e'$ is an MST without e.

The "only if" direction:

- Assume that there is a minimum spanning tree T of G that does not contain e. We will show that there exists a cycle C in G containing e consisting only of minimum weight edges. Let's consider adding edge e to T. Since T is a tree, the addition of e creates a cycle C. Since e has minimum weight, it follows that all edges in C must also have minimum weight, otherwise we could replace

an edge in C with the lighter e , creating a new spanning tree with smaller weight, which is a contradiction to our assumption that T is an MST.