Maximize Payout **Jonathan Wenger**

1. The algorithm used for solving this problem.

The way I went about solving this problem is taking both lists, A and B, and sorting them from greatest value to lowest value (in descending order). I then iterated through both sorted lists consecutively, multiplying $(a_i)^{b_i}$ at every index in the lists (in order). That value will be the maximized payout. This algorithm (intuitively) maximizes payout, because it takes the highest possible values from Lists A and B at every step of i, and it is impossible to get a higher value for $(a_i)^{b_i}$ than (the maximum value of A)$^{\text{the maximum value of B}}$. This algorithm is in fact greedy, because this algorithm doesn't take any previous steps into account for deciding which next value to take from Lists A or B; it just takes the current maximum values that haven't been used yet in the multiplicative series.

2. Prove (rigorously) the correctness of your algorithm: i.e., that it maximizes payout.

Let's call the product produced by this greedy algorithm S, and let's call the product produced by the most optimal solution S*. It is a given that S*≥S, as if this statement was incorrect, than S* would not be optimal.

Now, we need to prove that S*≤S, which will be proven now.

Lemma #1 – Given two positive integers, X and Y, and X≥Y, then $(X^X*Y^Y)\geq(X^Y*Y^X)$. This will be proven using a direct proof.

- Let the first term=X^X, the second term=Y^Y, the third term=X^Y, and the fourth term=Y^X.
- X^X (the first term) is ≥ X^Y (the third term) by a factor of X^{X-Y}, and Y^X (the fourth term) is ≥ Y^Y (the second term) by a factor of Y^{X-Y}. Because X≥Y, $X^{X-Y}\geq Y^{X-Y}$. Because the first term is ≥ the third term by a larger scale than the fourth term is ≥ the second term (i.e. $X^{X-Y}\geq Y^{X-Y}$), then $(X^X*Y^Y)\geq(X^Y*Y^X)$.

Lemma #2 – For any K such that 1≤K≤N, ($\prod_1^K (a_i)^{b_i}$ of S*)≤($\prod_1^K (a_i)^{b_i}$ of S) (i.e. "greedy stays ahead").

This will be proven by induction.

- **The base case**, or when K=1, or $\prod_1^1 (a_i)^{b_i}$ of the greedy algorithm (S), will just be an exponent, which equals (the max value of A)$^{\text{the max value of B}}$. Because there are no greater values of A or B in the lists, there is no greater possible value of $(a_i)^{b_i}$, so the base case for greedy is certainly optimal (meaning ($\prod_1^1 (a_i)^{b_i}$ of S*)≤($\prod_1^1 (a_i)^{b_i}$ of S)).
- **The inductive step**: We want to prove that if ($\prod_1^K (a_i)^{b_i}$ of S)≥($\prod_1^K (a_i)^{b_i}$ of S*), then ($\prod_1^{K+1} (a_i)^{b_i}$ of S)≥($\prod_1^{K+1} (a_i)^{b_i}$ of S*). Let T = the next "term" (i.e. the "K+1'th term") that will be taken into the series. So for the greedy algorithm (S), T=(the next unused maximum of A)$^{\text{the next unused maximum of B}}$. I will now prove that (($\prod_1^K (a_i)^{b_i}$) * $(a_{K+1})^{b_{K+1}}$ of S)≥(($\prod_1^K (a_i)^{b_i}$) * $(a_{K+1})^{b_{K+1}}$ of S*), where T= $(a_{K+1})^{b_{K+1}}$ of S, and T*= $(a_{K+1})^{b_{K+1}}$ of S*. Even if T*>T, S*≤S (up to and through K).
  - It has already been stated that ($\prod_1^K (a_i)^{b_i}$ of S)≥($\prod_1^K (a_i)^{b_i}$ of S*). While it is certainly possible that T* (the "K+1'th term" for S*) will have higher values of a^*_i and b^*_i than T would (meaning T*>T), that just means that the greedy algorithm (S) used those values of a^*_i and b^*_i at an earlier step in the series (before K). This is true because the greedy algorithm takes the maximum unused A

and B at every step. In that case where T*>T, then ($\prod_{1}^{K}(a_i)^{b_i}$ of S)>($\prod_{1}^{K}(a_i)^{b_i}$ of S*). Let's say that T*>T (which is certainly possible, and is a worst case). Based on Lemma #1, let's call the first term= $\prod_{1}^{K}(a_i)^{b_i}$ of S, the second term= $(a_{K+1})^{b_{K+1}}$ of S (i.e. T), the third term= $\prod_{1}^{K}(a_i)^{b_i}$ of S*, and the fourth term= $(a_{K+1})^{b_{K+1}}$ of S* (i.e. T*). If T*>T, then $\prod_{1}^{K}(a_i)^{b_i}$ of S is greater than $\prod_{1}^{K}(a_i)^{b_i}$ of S* by a larger scale than T* is greater than T. As stated in Lemma #1, because the first term is greater than the third term by a larger scale than the fourth term is greater than the second term, (($\prod_{1}^{K}(a_i)^{b_i}$) * $(a_{K+1})^{b_{K+1}}$ of S)≥(($\prod_{1}^{K}(a_i)^{b_i}$) * $(a_{K+1})^{b_{K+1}}$ of S*), meaning ($\prod_{1}^{K+1}(a_i)^{b_i}$ of S)≥($\prod_{1}^{K+1}(a_i)^{b_i}$ of S*), which proves the inductive step.

Because we proved Lemma #2 (that "greedy stays ahead" up to and through K), when K=N, this greedy algorithm "stays ahead" through N (which is all numbers in the series), which shows that greedy is optimal (because it stays ahead throughout the entire algorithm - that S*≤S).
Finally, because we proved that S*≥S, as well as S*≤S, it must be that S*=S, and the greedy algorithm is optimal.

3. State (and prove) the running time of your algorithm.

Given the size of Lists A and B are size n, the runtime of this algorithm is O(n log n). This is because my algorithm…

- Copies both final lists into new lists (which is O(n) for each list, totaling 2n)
- Sorts both A and B in descending order, which is O(n log n) for each list, totaling 2(n log n)
- Iterates through A and B to find the product of the multiplicative series (which is O(n) for each list, totaling 2n)

Therefore, the total runtime is 2n+2(n log n)+2n. Because 2(n log n) grows faster than all of the other operands, and for Big-O we don't care for constants, the runtime of this algorithm is O(n log n).