We are given n ads and a list of m ad slots on a web page. We know the probability a_i that a user sees the ith ad slot, the probability b_j that the user clicks on ad j given that they see the ad, and the revenue r_j that the web owner receives if the user clicks on the ad. Every ad may be displayed at most once. We can assume that $n \geq m$. We need to find an injective map from the set of slots to ads to maximize

$$\sum_{i=1}^{m} a_i b_{f(i)} r_{f(i)}.$$

1. A greedy algorithm for this problem is as follows:

   First, create and sort a list $L = \{b_1 r_1, \ldots, b_n r_n\}$, where each element is the product $b_j r_j$ of an ad, and the elements are sorted in decreasing order. Create and sort a list $S = \{a_1, \ldots, a_m\}$, where each element a_i is the probability the user sees an ad slot, sorted in decreasing order.

   While S is not empty, remove the first ad (i.e. the ad with the largest $b_j r_j$) from L, and remove the first ad slot (i.e. the slot with the largest a_i) from S. Assign this ad to this slot. Repeat until all slots have been assigned ads.

   Note that this algorithm displays each ad at most once, as desired.

2. Creating and sorting L takes $O(n \log n)$ time. Creating and sorting S takes $O(m \log m)$ time. Assigning ads to slots takes $O(m)$ time. In total the algorithm takes

   $$O(n \log n) + O(m \log m) + O(m) = O(n \log n + m \log m)$$

   time.

3. Finally we prove that the algorithm is correct.

   Proof. Fix an ordering of the m slots by decreasing probability that the user sees that slot. We can represent a solution to the problem as a list of m elements of the form $b_j r_j$, where the index of the element corresponds to the fixed ordering of ad slots. For example, let A be the solution given by the algorithm, then

   $$A = \{b_1 r_1, \ldots, b_m r_m\}.$$

   Now let an optimal solution be $O = \{b'_1 r'_1, \ldots, b'_m r'_m\}$ where again each element of O represents an ad which is assigned to the corresponding ad slot given by our fixed ordering.

   What is the expected ad revenue of each of these solutions?

   $$\text{revenue}(A) = \sum_{i=1}^{m} a_i b_i r_i, \quad \text{revenue}(O) = \sum_{i=1}^{m} a_i b'_i r'_i$$

   First, we claim that the m ads in the algorithm solution are the same as the m ads in the optimal solution. Then, we'll show that each of the m ads in the algorithm solution

are assigned to the same ad slots as in the optimal solution. Then, we'll be able to conclude that the algorithm and optimal solutions are the same.

First, we know that the m ads in the algorithm solution must be the same as the m ads in the optimal solution. By construction, we know that the ads in the algorithm solution are those with the highest products $b_i r_i$ possible. If some ad was in the algorithm solution but not in the optimal solution, say ad $j \in A$, ad $j \notin O$, but ad $k \in O$, ad $k \notin A$, then it must be that $b_j r_j > b_k r_k$. Then we could replace ad k in the optimal solution with ad j, and thus increase revenue(O), a contradiction to the optimality of O. Thus, the m ads in A must be the same as the m ads in O.

Now we want to show that the ordering of A and O are the same, i.e. that each ad is assigned to the same slot. We claim that the elements of O must be sorted in descending order. Consider the following lemma:

> Suppose we have $X = \{x, x+\delta\}$, $Y = \{y, y+\epsilon\}$, for positive x, y, δ, ϵ, so $x < x+\delta$ and $y < y+\epsilon$. We need to pair elements of X and Y so that the sum of the products of the elements of the pairs is maximized. There are two ways to pair these elements, we can consider the resulting sums:
>
> $$xy + (x+\delta)(y+\epsilon) = 2xy + \epsilon x + \delta y + \epsilon\delta$$
> $$x(y+\epsilon) + (x+\delta)y = 2xy + \epsilon x + \delta y$$
>
> seeing that the first sum is greater than the second, we conclude that to maximize the sum of the products of the elements of the pairs, we need the higher element of X to be paired with the higher element of Y.

The result of the lemma means that given any two ad slots, we can assign ads to maximize the revenue from those slots: we need to pair ads with higher $b_i r_i$ products to slots with higher a_i. Now, given a fixed decreasing ordering for the ad slots, when we assign ads to slots in the optimal solution, we can apply a bubblesort algorithm to swap adjacent ads as necessary, and in the end, we know that our solution has the highest possible revenue. It follows that the optimal solution will have $b'_i r'_i > b'_{i+1} r'_{i+1}$, i.e. that the optimal solution will be sorted in descending order.

By the construction of the algorithm, the elements of A are also sorted in descending order. Then since A and O contain the same elements in the same order, then the algorithm and optimal solutions are the same. Our algorithm is optimal. □