

Interval Blast

This is a **regular task**. You must submit a PDF, which can be produced using the L^AT_EX template on Moodle, exported from a word processor, hand-written or any other method.

As a lover of puzzles, you have come up with a new strategy puzzle game called INTERVALBLAST. In this game, you are given a row of n squares, where each square is labelled with a positive integer. On each turn, you need to choose two squares with no missing square in between them. Once you choose the two squares, you earn points equal to the product of the integers on the labels. Afterwards, your two chosen squares *and* all of the squares in between are removed. Once a square is removed, it cannot be chosen in a future round. The game ends when there are no more valid turns left or if there are no more squares left to choose, and the aim of the game is to maximise the number of points by the end of the game.

To see this in action, consider the following game; each row shows a distinct turn. The red underlined numbers are the two squares chosen on that turn, and the grey squares are the ones which were removed in earlier turns. A partial score **prior** to the turn is supplied. No moves are possible on Turn 4.

Turn	Interval								Partial Score
1	3	1	4	1	<u>5</u>	<u>9</u>	2	6	0
2	<u>3</u>	1	<u>4</u>	1			2	6	45
3				1			<u>2</u>	<u>6</u>	57
4				1					69

In this game and following sequence of moves, the number of points earned is

$$5 \times 9 + 3 \times 4 + 2 \times 6 = 45 + 12 + 12 = 69.$$

Note that this is not necessarily the highest possible number of points that could be earned for this specific instance.

Design and analyse an $O(n^2)$ algorithm that computes the maximum number of points that can be earned in a given instance of INTERVALBLAST.

Advice. Your solution should include:

- A clear subproblem definition.
- Base cases for your subproblem definition.
- A well-defined recurrence with respect to your subproblem definition and base cases.
- Some output that solves the original problem, as a function of the results generated by your recurrence.
- A correct order of computation, with respect to your recurrence.
- Time complexity analysis for your algorithm.
- Justification that your algorithm solves the problem correctly, with specific reference to the correctness of the base case(s), recurrence and overall answer.

Expected length: Up to a page.