Delaram Fartoot

Problem Representation:

One of the main problems is how to represent the question as input for our search algorithms. We solve this problem by representing our pattern of cuts with a permutation of the desired portions. Then, we calculate the number of needed rolls with this pattern of cuts. For each cut, we move to the next roll only if the next portion is not available on the rest of the current roll. Now, with this representation, we perform a search using three different algorithms.

Hill Climbing:

The main idea of the hill climbing algorithm is to randomly choose an initial state and make small changes to the state in different directions, or in other words, compute the value of all neighbors and return the one with the optimum amount as the current state. Hill climbing has many variants; in the one we implemented; our input is as discussed above. For the next move, we choose the first neighbor which has a lower value than the current state. The neighbor is calculated by swapping two indexes in our list. We continue this procedure until the maximum number of iterations is reached. Also, because the initial state can greatly influence our results, we run the algorithm multiple times and take the best result as our final result, which is as follows:

Input 1	Input 2	Input 3	Input4
50	82	97	220

Simulated Annealing:

In simulated annealing with randomization, we give more chances to patterns with higher values. In every state, we randomly build a neighbor (swapping two indexes). If the neighbor is better than the current state, we choose it as the current state. If not, by a probability, we still accept it, and we reduce that probability as we iterate more. In our implementation, we set our temperature variable to 10000 and use the Boltzmann distribution for our probability. Additionally, we reduce that probability by multiplying it by 0.98 (alpha) in each iteration. We continue this process until the temperature (t) is less than the minimum t we chose. Similar to hill climbing, because the initial points matter, we repeated the algorithm multiple times, and the final results were as follows:

Input 1	Input 2	Input 3	Input 4
52	81	102	233

As you can see the hill climbing algorithm did slightly better.

Genetic:

Genetic search is computationally more expensive and overall a more complicated algorithm. In our genetic algorithm, we chose a population of 50. Our initial population was randomly generated by different permutations of our input. Consider our population; in each iteration, we choose 10 pairs of the population to create a child through crossover. We randomly select the index of the crossover. After this step, we have 10 new children. With a probability of alpha (0.3), mutation occurs for each child, involving the swapping of two indexes. After constructing the children completely, we replace the children with the population if a child's fitness is better than the weakest member in the population. We continue this iteration until a maximum iteration is reached. The first member of our population will be our final answer. Like other algorithms, we repeat this algorithm multiple times to obtain the best result. It is important to mention that this algorithm is computationally heavier than others, so we couldn't try it on many iterations, and we are positive that without computational constraints, it would have outperformed other algorithms in all inputs. The results are as follows:

Input 1	Input 2	Input 3	Input 4
54	80	105	238