Code Jam 2018 - Round 1A

Analysis: Edgy Baking

Cookie-cutting

A cookie with width W and height H has a perimeter of 2 × (W + H). If we make a straight cut of length C that divides the cookie in half, each piece will have one side of length C, and the remaining sides of the pieces will represent the perimeter of the original cookie. So, after cutting, we will have a combined perimeter X = 2 × (W + H + C).

Clearly X is smallest when we leave the cookie alone. However, if we want to minimize X given that we are making a cut, we must minimize C; no cut can be smaller than the smallest of W and H, so we should cut through the midpoints of the two larger sides, which gives the cut a size of the smaller of the two sides. Then X = 2 × (W + H + min(W, H)). On the other hand, if we want to make a cut that maximizes X, we must maximize C by cutting through two opposite corners of the cookie. Then C = sqrt(W^2 + H^2), so X = 2 × (W + H + sqrt(W^2 + H^2)). If we start our cut somewhere between one of those larger-side midpoints and a corner, we will get a value of X somewhere between these two extremes; since we can cut anywhere we want in that interval, we can get any intermediate value of X that we want.

So, depending on what we do with our cookie, its contribution to our overall perimeter sum will either be 2 × (W + H), or a value in the range 2 × (W + H) + [2 × min(W, H), 2 × sqrt(W^2 + H^2)]. From now on, we will abbreviate the extremes of that range as [L, R]. (We could divide all values in the problem by 2 as well, removing that factor, but we will retain it in this analysis for clarity.)

Test set 1

In test set 1, all of the cookies have the same dimensions; we will call them W and H. Let us define P' as **P** - 2 × (W + H) — that is, the amount of extra perimeter that we need to add to reach the target **P**. So, we can reformulate the problem as starting with a total perimeter of 0, and trying to reach P'. For each cookie, we can choose to either not cut it, which leaves our total perimeter unchanged, or cut it, which increases our total perimeter by some value of our choice in the range [L, R]. If we cut K cookies, then our total perimeter can be anywhere in the range [K × L, K × R], depending on how we make our cuts. We will think in terms of this range and not in terms of the individual cuts.

How many cookies should we cut? Of course, we should never cut so many that K × L exceeds P'. However, we might as well keep cutting cookies up until that point, since doing so both moves our range closer to the target and makes the range wider. So, we can solve for X in the equation X × L = P', and then choose K to be the floor of X — that is, floor(P' / L). (Since L is an integer representing one of the original side lengths, and P' is also guaranteed to be an integer, we should use integer-based division here to avoid the usual problems with flooring floating-point numbers!)

Then, we can check whether the range [K × L, K × R] includes P'. If it does, the answer is 0; otherwise, it is P' minus the right end of the range, i.e., P' - K × R. Even though R is generally not an integer, we do not need to worry about floating-point issues; even if P' is very close to K × R and we mistakenly conclude that it is not in the range, we will still get an answer that is sufficiently close to 0 under our floating point rules.

Test set 2

In test set 2, different cookies might have different dimensions, so we get different total ranges depending on which ones we cut. We have no a priori basis for deciding which cookies to cut, and we cannot directly check all 2^{100} possibilities, but we can use dynamic programming to ensure that we find the best possibility without explicitly checking them all; the problem is very similar to the knapsack problem. For each cookie, we are deciding whether to leave it as is, or cut it, which adds L to our total perimeter and gives us up to R - L units of additional "slack". Once we have finished deciding which cookies to cut, we can include as much or as little of this "slack" as we want. All other things being equal, having more slack is always better for us. The large limit on **P** may seem daunting, but you can convince yourself that the problem space is actually small enough for methods like this to work.

However, the problem can also be solved in a different way. As in the previous solution, each cookie corresponds to a range [L, R] by which we can increase the total perimeter if we decide to cut that cookie. Let S(K) be the list of intervals we can reach after processing the first K cookies. We start with S(0) = {[0, 0]}.

When we consider the K'th cookie, we can choose to cut it or not. Suppose that this cookie corresponds to the interval [L, R]. If we do not cut it, we can obtain any perimeter that is already in an interval in S(K - 1). If we do cut it, we can reach any interval in the set S' given by S' = {[l + L, r + R] : [l, r] is an interval in S(K - 1)}.

To get S(K), we first take the union of S(K - 1) and S', followed by merging overlapping intervals. For example, if S(K - 1) = {[0, 0], [3, 6]}, and we add the interval [L, R] = [1, 2], we obtain S' = {[1, 2], [4, 8]} and S(K) = {[0, 0], [1, 2], [3, 8]}. Note that the intervals [3, 6] from S(K - 1) and [4, 8] from S' merge into a single interval [3, 8] here.

We can drop any intervals that start after P', so that after processing all N cookies, the answer to the question will be the distance from P' to the last interval.

Without any further analysis, we might expect the size of S(**N**) to be $2^{\mathbf{N}}$, as it could double in each step. It turns out that we can provide a much stronger bound of (log(P') / log(sqrt(2))) + 1.

We first need the observation that any interval [L, R] corresponding to a cookie will satisfy R ≥ sqrt(2) × L, where equality holds for a square. Using induction, we can infer that every interval [l, r] in each S(K) also satisfies r ≥ sqrt(2) × l.

The second observation is that all intervals in S(N) are disjoint. Let us now sort intervals in S(N) and enumerate them starting from $[l_0, r_0]$=[0, 0]. Then the observations together imply that $l_{\{i+1\}} > r_i \geq \text{sqrt}(2) \times l_i$. Since the lower bound of each interval is an integer, we also have $l_1 \geq 1$. Hence we have $l_i \geq \text{sqrt}(2)^{i-1}$. Since we can discard any intervals starting after P', we may assume that $l_i \leq$ P'. We conclude that there are at most (log(P') / log(sqrt(2))) + 1 intervals in S(**N**).

This solution takes O(**N** log(P')) time: for each of the **N** cookies, we have to perform one merging step that takes O(|S(K)|) time.

It can also be shown that |S(K)| ≤ 2K + 1. We leave this as an exercise to the reader!