HS 2023

Prof. R. Wattenhofer
Robin Fritsch

Computational Thinking
Sample Solutions to Exercise 2

1 Egg dropping

a) If we only have one egg, the only possibility to definitely find the critical floor is to try all floors from bottom to top. In the worst case, this will take us n throws of the egg.

b) With $\lceil \log n \rceil$ eggs, we have enough eggs to do binary search, i.e. repeatedly choose the middle floor of all remaining possible floors and toss the egg from there. This will take $\lceil \log n \rceil$ throws in the worst case.

c) For the general case of k eggs, we can use dynamic programming to solve the problem. Let $dp[n][k]$ be the minimum number of tosses required to find the critical floor in the worst case for n floors and k eggs. Clearly, $dp[0][k] = 0$ for all $k \geq 0$ and $dp[n][0] = \infty$ for all $n \geq 1$.

To calculate $dp[n][k]$ we simply consider every possible floor i from which we can throw the next egg and check which floor leads to the smallest number of remaining tosses. If we throw the egg from a certain floor i, it can either break, meaning we need to search the $i-1$ floors below with $k-1$ eggs, or not break, meaning we need to search the $n-i$ floors above with k eggs. In the worst case this requires $\max(dp[i-1][k-1], dp[n-i][k])$ throws by the definition of dp. Therefore,

$$dp[n][k] = 1 + \min_{1 \leq i \leq n} \Big(\max \big(dp[i-1][k-1], dp[n-i][k] \big) \Big).$$

In the notebook this solution is implemented using memoization and storing the values of dp in a dictionary. Alternatively, one could simply construct the whole table of the values of dp up to the input values of n and k. → notebook

Finally, it remains to find the optimal sequence of floors from which to toss the eggs. Analogously to the computation of dp, the best floor to toss an egg from for n floors and k eggs is

$$floor[n][k] = \arg\min_{1 \leq i \leq n} \Big(\max \big(dp[i-1][k-1], dp[n-i][k] \big) \Big).$$

Alternatively to the implementation in the notebook, one could also build a table of the values of $floor$ at the same time as building the table for dp. → notebook

2 Pizza world record

We want to find the largest r among all radii $r \in \mathbb{R}$ and centers $c \in \mathbb{R}^2$ of the pizza such that the pizza still fits in the polygon. So the linear program will look something like the following.

$$\begin{aligned} \max \quad & r \\ \text{s.t.} \quad & \textit{“disc fits in polygon”} \\ & \begin{pmatrix} c \\ r \end{pmatrix} \in \mathbb{R}^3 \end{aligned}$$

We now need to write the fact that the pizza fits in the polygon as a linear constraint $A \begin{pmatrix} c \\ r \end{pmatrix} \leq b$ with suitable $A \in \mathbb{R}^{k \times 3}$ and $b \in \mathbb{R}^k$. Furthermore, most solvers for linear programs take a minimization problem as an input. So we use the fact that maximizing r is equivalent to minimizing $-r$, and write the linear program as

$$\begin{aligned} \min \quad & \begin{pmatrix} 0 & 0 & -1 \end{pmatrix} \begin{pmatrix} c \\ r \end{pmatrix} \\ \text{s.t.} \quad & A \begin{pmatrix} c \\ r \end{pmatrix} \leq b \\ & \begin{pmatrix} c \\ r \end{pmatrix} \in \mathbb{R}^3. \end{aligned}$$

The tricky part is to find suitable A and b for the constraint.

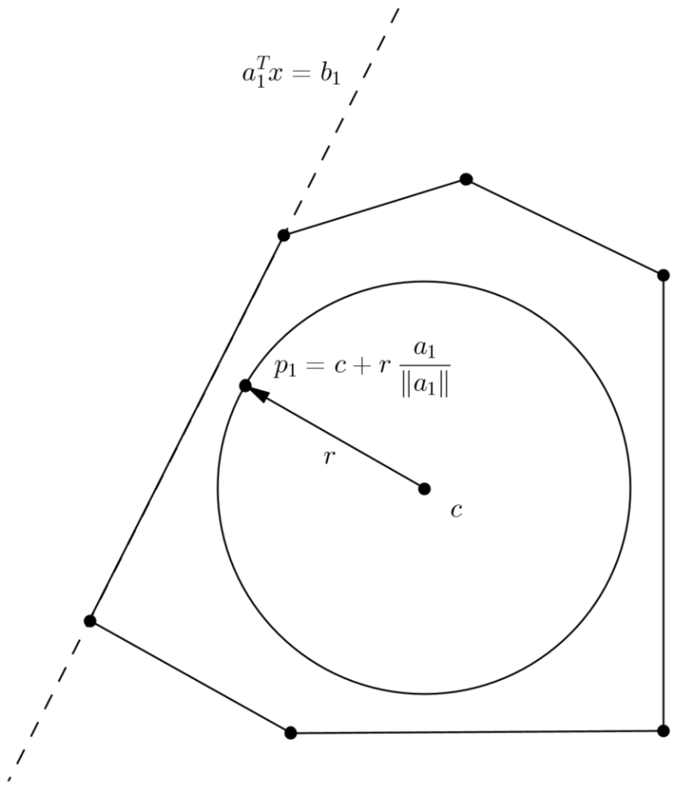
Every edge of the polygon gives us one constraint for the linear program: From the coordinates of every two neighboring points of the of polygon we can calculate $a_i \in \mathbb{R}^2$ and $b_i \in \mathbb{R}$ of the equation $a_i^T x = b_i$ of the line through those two points (see solution in notebook). Since the polygon is convex, the whole pizza needs to lie on one side of the this line. In other words, $a_i^T x \leq b_i$ must hold for all points $x \in \mathbb{R}^2$ on the pizza. → notebook

First, note that it suffices to check this condition for all points on the border of the pizza, i.e. all $x = c + ry$ for some unit vector $y \in \mathbb{R}^2$. Furthermore, the border comes closest to the line when this vector y is perpendicular to the line, i.e. $y = a_i / \|a_i\|$. So we only need to make sure the point $p_i = c + r \cdot a_i / \|a_i\|$ satisfies $a_i^T p_i \leq b_i$. This is equivalent to

$$a_i^T \left(c + r \frac{a_i}{\|a_i\|} \right) \leq b_i \quad \iff \quad a_i^T c + \|a_i\| r \leq b_i.$$

With this we can write the constraints of the linear program as

$$\begin{pmatrix} a_1^T & \|a_1\| \\ \vdots & \vdots \\ a_n^T & \|a_n\| \end{pmatrix} \begin{pmatrix} c \\ r \end{pmatrix} \leq \begin{pmatrix} b_1 \\ \vdots \\ b_n \end{pmatrix}.$$


$a_1^T x = b_1$
$p_1 = c + r \, \frac{a_1}{\|a_1\|}$
r
c