Romanian Master of Informatics
6th Edition, Bucharest, 10th - 13th of October 2018

Climbers – solution

Author: Cătălin Frâncu

Without changing the answer to the problem, we can collapse sequences of three or more sorted altitudes, keeping only the first and the last one. For example, we can shorten `0 1 3 7 5 2 8 5 0` to just `0 7 2 8 0`. For the rest of this document, we will assume that altitudes are strictly zigzagging.

Let **nodes** be the inflection points of the mountain range, numbered from 1 to n, having altitudes $h_1, \ldots, h_n$, and let **segment** s refer to the slope between nodes s and $s+1$, including its endpoints. (We will sometimes refer to segment n to mean just the node n itself.)

Notice that any optimal solution implies that climbers change direction only when at least one of them is in a node. Indeed, if they both stop mid-slope, then there is no way to go but back, which simply increases the cost with no benefit.

Therefore, it suffices to explore **climber states** of the form (x, y), meaning that one of the climbers (not necessarily Alice) is at node x and the other is on segment y.

For example, for the landscape in Figure 1 the optimal solution passes through the states $(1,7) \to (2,6) \to (3,6) \to (6,3) \to (5,3) \to (4,4)$. State (6, 3) is shown explicitly.

Note that not all states (x, y) are defined, only those for which segment y spans altitude h_x. Furthermore, if state (x, y) is defined, then it uniquely identifies the climbers' positions, since segment y cannot span altitude h_x more than once. It follows that there are at most n^2 states.

Thus, we can view the landscape as a weighted graph where

- vertices correspond to climber states;
- edges correspond to movements between states;
- the weight of an edge is the vertical distance between its adjacent states;
- the climbers start in state $(1, n)$;

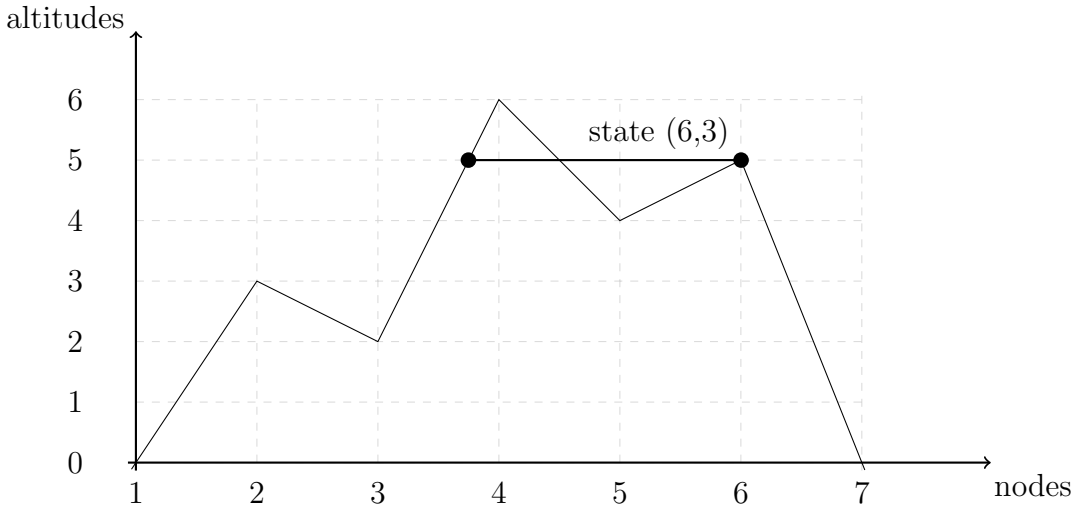

Figure 1: A sample landscape.

- the climbers wish to arrive at the nearest state (x, x).

A vertex (x, y) in this graph can have a degree of:

- 2 if $h_y < h_x < h_{y+1}$. In this case, the climber at x can move forward and back, while the climber on y only has one way to go.
- 4 if $h_x = h_y$ or $h_x = h_{y+1}$. In this case, both climbers can move forward or back.
- 1 for state $(1, n)$, since both climbers can only move up.
- 1 for states of the form (x, x), since the climbers can only part ways and go back where they came from.

(For completeness, note that there are an odd number of altitudes, since they are strictly zigzagging. Therefore, there is an even number of nodes of degree 1, so the sum of all degrees is even as expected.)

In conclusion, we can use Dijkstra's algorithm to find the shortest path from vertex $(1, n)$ to any vertex (x, x). Enumerating the edges of the graph (i. e. finding all the climbers' movements from a given state) is tricky, but doable. This requires $O(n^2 \log n)$ time and $O(n^2)$ space.

As an optimization, we could avoid inserting vertices of degree 2 into the priority queue in Dijkstra's algorithm. When we encounter such a vertex, we keep following it to a vertex of degree 4 or 1. This cuts the running time significantly, but is not required for the maximum score.

There will always exist a solution. The connected component containing the initial state must contain at least another vertex of degree 1, which means that starting in the initial state and following any simple path must lead us to a final state.

Special case: all altitudes are distinct

When all the altitudes are distinct, there are no vertices of degree 4 in the graph model above. This means that the path from $(1, n)$ to (x, x) passes only through nodes of degree 2. Thus, we can blindly walk from state to state, taking care not to go back to the previous state, and we will eventually arrive at the final state (which is also unique, since there is a single highest peak). This requires $O(n^2)$ time and $O(n)$ space.

Special case: $n \times H \leq 40,000$

We feel that the trickiest part of implementing the program is enumerating the transitions between states. When $n \times H$ is small, we can avoid this complication. Let us interpolate the altitudes from h_x to h_{x+1} on every segment x. For example, `0 3 1 4 0` becomes **0** `1 2` **3** `2` **1** `2 3` **4** `3 2 1` **0**.

There will be at most $n \times H$ nodes in this interpolated landscape and consecutive altitudes will differ by exactly one. State transitions become a lot easier to define, as climbers can only move from (x, y) to $(x \pm 1, y \pm 1)$ as long as the altitudes match. Also, all edges in the equivalent graph have a weight of 1, so we can use a breadth-first search instead of Dijkstra's algorithm. This only requires one bit per vertex to keep track of visited vertices, as well as a queue of active nodes, which will be very small in practice.

The running time is $\mathrm{O}(n^2H^2)$.