Algorithm Design Laboratory with Applications

Prof. Stefano Leucci

Problem: *Phylogenetic Trees.*

A phylogenetic tree is a rooted tree in which each vertex represents a species and an edge (u, v) encodes the fact that v is a direct descendant of u. You are working an interdisciplinary research project and you often need to consult phylogenetic trees in order to find the most recent common ancestor of two species.

Tired of dealing with printouts of the diagrams, you decide to design an algorithm that can preprocess a phylogenetic tree T once for all, and is then able to answer any number of queries of the following kind: *Given two vertices u and v of T, report the deepest vertex of T that is an ancestor of both u and v.*

Input. The input consists of a set of instances, or *test-cases*, of the previous problem. The first line contains the number T of test-cases. The first line of each test-case contains the number n of nodes of T, which are indexed from 0 to $n-1$, and the number q of queries. The next line contains $n-1$ integers $p_1, \ldots, p_{n-1}$ where p_i is the index of the parent of the vertex with index i in T. The root of T is vertex 0. Finally, the j-th of the following q lines describe a query and contains the indices of two vertices u_j and v_j of T.

Output. The output consists of T lines. The i-th line is the answer to the i-th test-case and contains q integers, $r_1, r_2, \ldots, r_q$ separated by a space. The j-th integer r_j is the index of the deepest vertex in T that is an ancestor of both u_j and v_j in T.

Assumptions. $1 \leq T \leq 10$; $1 \leq n \leq 2^{13}$; $1 \leq q \leq 2^{18}$.

Example.

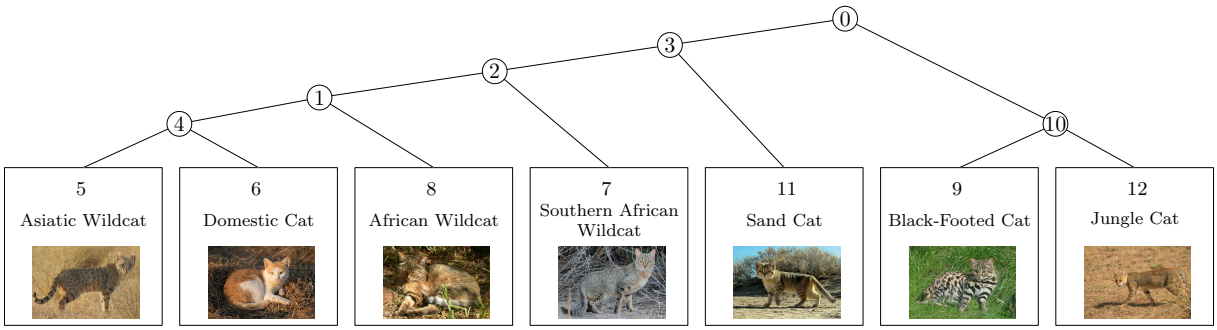

Figure 1: An example of phylogenetic tree. Note that, while this tree is binary, this does not need to be the case in general.

Input (corresponding to the tree in Figure 1):

```
1
13 3
2 3 0 1 4 4 2 1 10 0 3 10
5 8
1 3
9 0
```

Output:

```
1 3 0
```

Requirements. Your algorithm should have a preprocessing time of at most $O(n \log n)$, where n is the number of nodes of T, a query time of $O(1)$, and must be able to answer queries *online*, i.e., the answer to a query must be returned before the next query is read.

Notes. A reasonable implementation should not require more than 1 second for each input file. For the sake of simplicity, you can assume that the function `std::log2()` requires constant time.