CERC 2020

Presentation of solutions

September 26, 2021

Rescue Mission

Rescue Mission

- Task: Given an array $a = a_1, a_2, \cdots, a_N$ of integers, for each positition i in the array find the smallest subarray starting at i with sum equal to zero modulo 10.

Rescue Mission

- Task: Given an array $a = a_1, a_2, \cdots, a_N$ of integers, for each positition i in the array find the smallest subarray starting at i with sum equal to zero modulo 10.
- Observation: Consider the array of prefix sums modulo 10 f such that $f_i = \sum_{j=1}^{i} a_i \mod 10$ with $f_0 = 0$.
  - A subarray $a_i, a_{i+1}, \cdots, a_j$ has $\sum_{k=i}^{j} a_k = 0 \mod 10$ if and only if $f_{i-1} = f_j \mod 10$.

- Solution:
  - For each position i we want to find the closest next position j such that their prefix sums are equal modulo 10, that is $f_{i-1} = f_j$.
  - For each $i \in \{0, 1, \cdots, 9\}$ keep a queue q_v of all positions j where $f_j = v$, sorted in ascending order.
  - When finding the closest endpoint for subarray starting at i, keep popping elements from $q_{f_{i-1}}$ until you reach an element greater or equal to i.
  - The resulting complexity is $\mathcal{O}(N)$.
- Similar solution using binary search with complexity $\mathcal{O}(N \log(N))$ also easily passes.

Pizzo Collectors

Pizzo Collectors

- Task: Find the maximum profit which can be achieved after replacing all '?' in the cyclic string s by valid letters
- The length of s is an integer power of a prime number
- Profit for string s is calculated as

$$\sum_{d \in D} \sum_{i=0}^{d-1} I(i,d) \cdot \sum_{j=0}^{N/d-1} c[s[(i + d \cdot j) \mod N]]$$

  - where N is the length of s, D is the set of divisors of N and c is given on the input.
  - $I(i,d) = 1$ if all letters in the subcycle $a_i, a_{i+d}, a_{i+2d}, \cdots, a_{i+N}$ are equal. Otherwise $I(i,d) = 0$.
- In other words, the resulting profit is equal to the sum of the costs of letters along each possible cycle whose length is a divisor of N (including 1 and N).

Pizzo Collectors

- Observation: The profit induced by the cycle C of size p^{i+1} depends only on the assignment of letters inside the p different cycles $C_1, C_2, \cdots, C_p$ of size p^i that it contains.
- No position in the string is shared between two different cycles of size p^i and their solution can be computed independently.
- This leads to the following recursive solution:
  - For each cycle of size p^{i+1} compute its maximum total profit and its profit if all letters are set to a (set to $-\infty$ if not possible).
  - The maximum total profit of C is the sum of maximums for all the cycles $C_1, C_2, \cdots, C_p$ it contains.
  - The profit of C if all its letters are set to a is the profit of all the cycles $C_1, C_2, \cdots, C_p$ it contains when set to a plus $p^{i+1} \cdot c[a]$.
  - Easy to compute for cycles of size 1. The final solution equals to solution for cycle of size $p^k = N$.

- Complexity: The size of alphabet is bounded and small ($a \in \{A, B, \cdots, Z\}$) and for $N = p^k$ each letter is contained in exactly $k + 1 = \mathcal{O}(\log(N))$ different cycles.
- The resulting complexity of the solution is $\mathcal{O}(N \log(N))$

Storage problem

Storage problem

- Task: Given a set of N items with weights $w_1, w_2, \cdots, w_N$ and a threshold K, for each item p and integer j $(1 \leq j \leq N-1)$ compute the number of subsets $S \subseteq N \setminus \{p\}$ of j items such that $W(S) \leq K$ and $W(S \cup \{p\}) > K$, where $W(S)$ is the total weight of S.

Storage problem

- DP solution in $\mathcal{O}(N^3 K)$.
- Let $f(S, i, j)$ be the number of subsets such that the following holds
  - All the items are in S.
  - The total weight is i.
  - The subsets have exactly j items
- It holds $f(\emptyset, 0, 0) = 1$.
- Let S_p be the set of items $S_p = \{1, 2, \cdots, p\}$.
- The DP transition: $f(S_p, i, j) =$

$$\begin{cases} f(S_{p-1}, i, j) & \text{if } i < w_n \text{ or } j = 0 \\ f(S_{p-1}, i, j) + f(S_{p-1}, i - w_n, j - 1) & \text{otherwise} \end{cases}$$

- Having $f(S_{N-1}, i, j)$ we can compute the answer for the last item and subsets size j:
  - $res_{i,j} = \sum_{i=K-w_n}^{K-1} f(S_{N-1}, i, j)$
- The order of items in the DP does not matter, we can recalculate for each of the N items using the same formula.

Storage problem

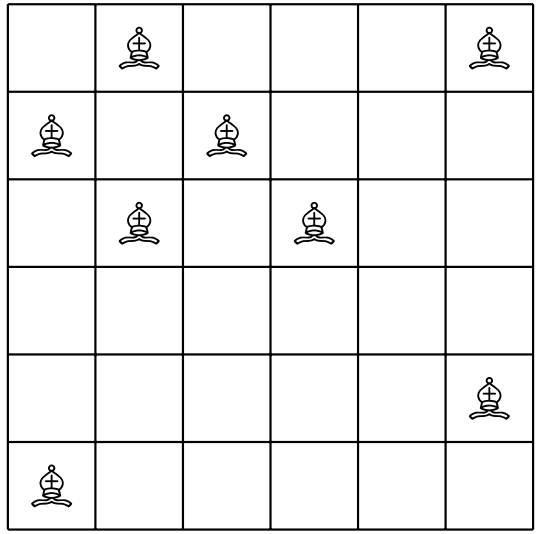
- One possible optimization to $\mathcal{O}(N^2K)$:
- For each item p set $w_p = min(w_i, K)$.
- Compute $f(S_N, i, j)$ using the procedure above for $0 \leq j \leq 2 \cdot K - 1$.
- Copy the table for $f(S_N, i, j)$ and recalculate for $f(S_N \setminus \{p\})$ for each item p:
- $f(S_N \setminus \{p\}, i, j) =$
$$\begin{cases} f(S_N, i, j) & \text{if } i < w_p \text{ or } j = 0 \\ f(S_N, i, j) - f(S_N \setminus \{p\}, i - w_p, j - 1) & \text{otherwise} \end{cases}$$
- Same as before, having $f(S_N \setminus \{p\}, i, j)$ we can compute the answer for item p and subsets size j:
  - $res_{i,j} = \sum_{i=K-w_n}^{K-1} f(S_N \setminus \{p\}, i, j)$

Storage problem

- The initial computation of $f(S_N, i, j)$ takes $\mathcal{O}(N^2 K)$
- Copying the table and recalculation to $f(S_N \setminus \{p\}, i, j)$ takes $\mathcal{O}(N \cdot K)$. This is repeated for each item.
- In total $\mathcal{O}(N^2 K)$.

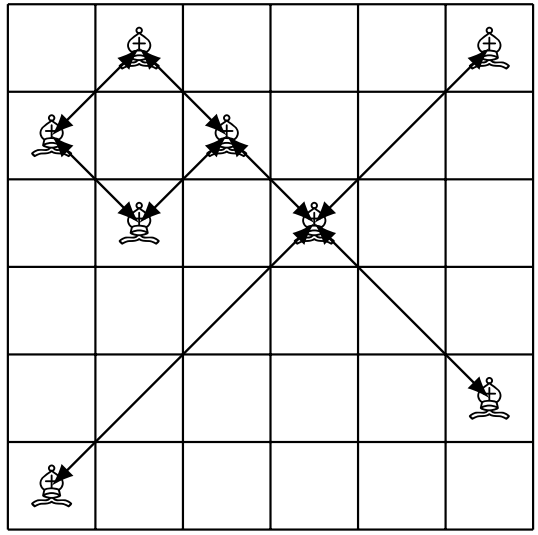
Excavation

Excavation



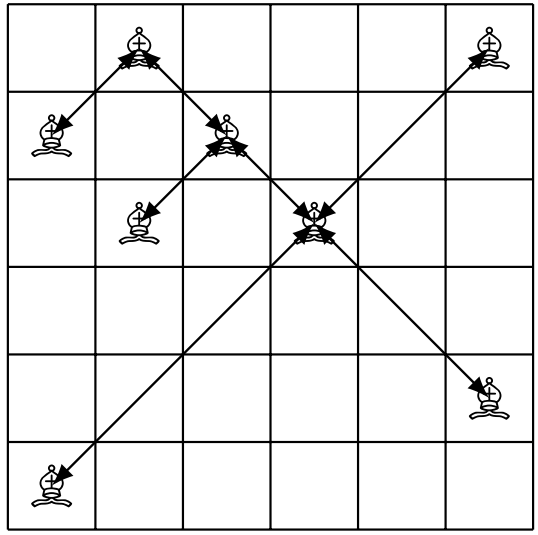
- We model the situation as a graph with edges between adjacent excavators (movement pattern-wise)
- If such graph is not connected, there is no solution
- Otherwise the graph is connected and has a spanning tree
- One solution is to successively pluck leaves of some spanning tree leaf-by-leaf

Excavation



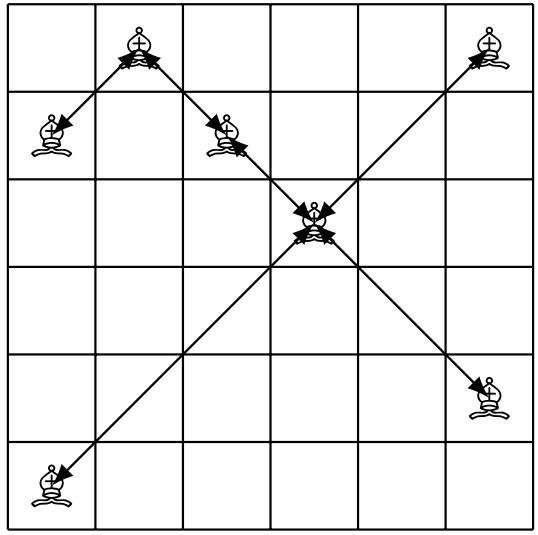
- We model the situation as a graph with edges between adjacent excavators (movement pattern-wise)
- If such graph is not connected, there is no solution
- Otherwise the graph is connected and has a spanning tree
- One solution is to successively pluck leaves of some spanning tree leaf-by-leaf

Excavation



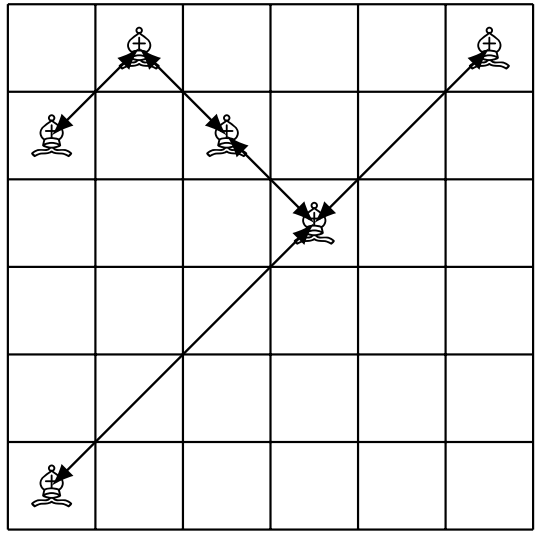
- We model the situation as a graph with edges between adjacent excavators (movement pattern-wise)
- If such graph is not connected, there is no solution
- Otherwise the graph is connected and has a spanning tree
- One solution is to successively pluck leaves of some spanning tree leaf-by-leaf

Excavation



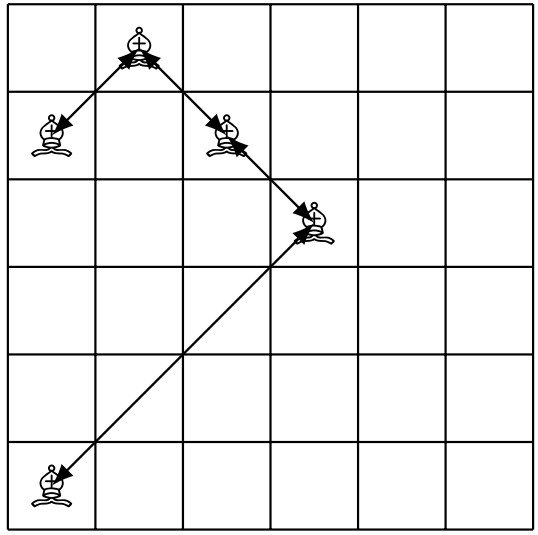
- We model the situation as a graph with edges between adjacent excavators (movement pattern-wise)
- If such graph is not connected, there is no solution
- Otherwise the graph is connected and has a spanning tree
- One solution is to successively pluck leaves of some spanning tree leaf-by-leaf

Excavation



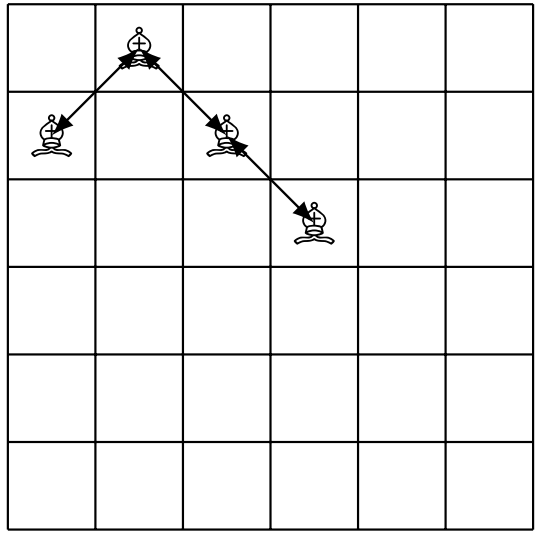
- We model the situation as a graph with edges between adjacent excavators (movement pattern-wise)
- If such graph is not connected, there is no solution
- Otherwise the graph is connected and has a spanning tree
- One solution is to successively pluck leaves of some spanning tree leaf-by-leaf

Excavation



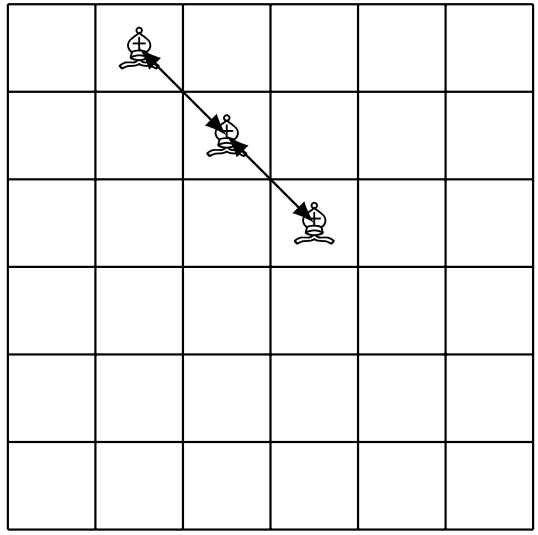
- We model the situation as a graph with edges between adjacent excavators (movement pattern-wise)
- If such graph is not connected, there is no solution
- Otherwise the graph is connected and has a spanning tree
- One solution is to successively pluck leaves of some spanning tree leaf-by-leaf

Excavation



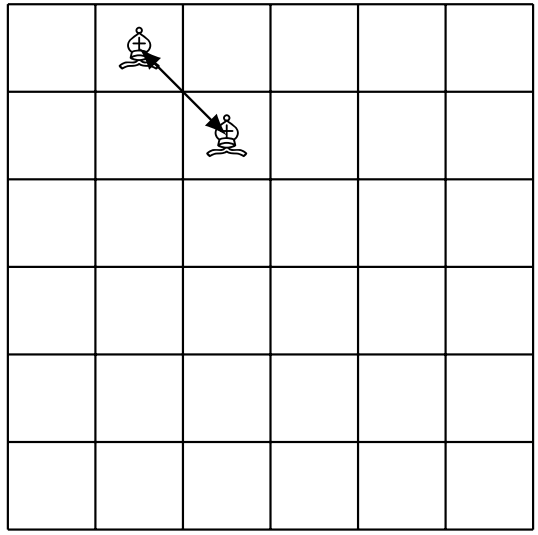
- We model the situation as a graph with edges between adjacent excavators (movement pattern-wise)
- If such graph is not connected, there is no solution
- Otherwise the graph is connected and has a spanning tree
- One solution is to successively pluck leaves of some spanning tree leaf-by-leaf

Excavation



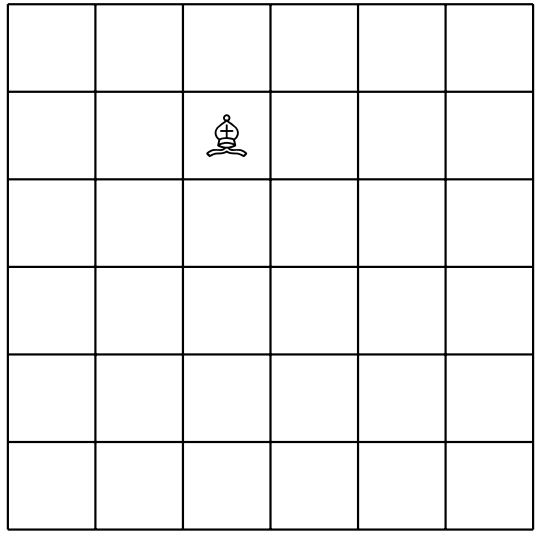
- We model the situation as a graph with edges between adjacent excavators (movement pattern-wise)
- If such graph is not connected, there is no solution
- Otherwise the graph is connected and has a spanning tree
- One solution is to successively pluck leaves of some spanning tree leaf-by-leaf

Excavation



- We model the situation as a graph with edges between adjacent excavators (movement pattern-wise)
- If such graph is not connected, there is no solution
- Otherwise the graph is connected and has a spanning tree
- One solution is to successively pluck leaves of some spanning tree leaf-by-leaf

Excavation



- We model the situation as a graph with edges between adjacent excavators (movement pattern-wise)
- If such graph is not connected, there is no solution
- Otherwise the graph is connected and has a spanning tree
- One solution is to successively pluck leaves of some spanning tree leaf-by-leaf

Tobacco Growing

Tobacco Growing

- Notice that if we plant a single tobacco in a grass corner with flower tile borders, we keep obtaining rows of Pascal triangle = "Corner gadget" to obtain powers of two

0	0	0	0	0	0
0	**1**	**0**	**0**	**0**	**0**
0	**0**	**0**	**0**	**0**	**0**
0	**0**	**0**	**0**	**0**	**0**
0	**0**	**0**	**0**	**0**	**0**
0	**0**	**0**	**0**	**0**	**0**

Tobacco Growing

- Notice that if we plant a single tobacco in a grass corner with flower tile borders, we keep obtaining rows of Pascal triangle = "Corner gadget" to obtain powers of two

0	0	0	0	0	0
0	**1**	**1**	**0**	**0**	**0**
0	**1**	**0**	**0**	**0**	**0**
0	**0**	**0**	**0**	**0**	**0**
0	**0**	**0**	**0**	**0**	**0**
0	**0**	**0**	**0**	**0**	**0**

Tobacco Growing

- Notice that if we plant a single tobacco in a grass corner with flower tile borders, we keep obtaining rows of Pascal triangle = "Corner gadget" to obtain powers of two

0	0	0	0	0	0
0	**3**	**2**	**1**	**0**	**0**
0	**2**	**2**	**0**	**0**	**0**
0	**1**	**0**	**0**	**0**	**0**
0	**0**	**0**	**0**	**0**	**0**
0	**0**	**0**	**0**	**0**	**0**

Tobacco Growing

- Notice that if we plant a single tobacco in a grass corner with flower tile borders, we keep obtaining rows of Pascal triangle = "Corner gadget" to obtain powers of two

0	0	0	0	0	0
0	**7**	**8**	**3**	**1**	**0**
0	**8**	**6**	**3**	**0**	**0**
0	**3**	**3**	**0**	**0**	**0**
0	**1**	**0**	**0**	**0**	**0**
0	**0**	**0**	**0**	**0**	**0**

Tobacco Growing

- However we need multiple powers of 2 at the same time $\Rightarrow$ "Delay gadget" to deliver starting 1 to a "Corner gadget" for a particular power

0	0	0
0	**1**	0
0	**0**	0
0	**0**	0
0	**0**	0

Tobacco Growing

- However we need multiple powers of 2 at the same time $\Rightarrow$ "Delay gadget" to deliver starting 1 to a "Corner gadget" for a particular power

0	0	0
0	**1**	0
0	**1**	0
0	**0**	0
0	**0**	0

Tobacco Growing

- However we need multiple powers of 2 at the same time $\Rightarrow$ "Delay gadget" to deliver starting 1 to a "Corner gadget" for a particular power

0	0	0
0	**2**	0
0	**2**	0
0	**1**	0
0	**0**	0

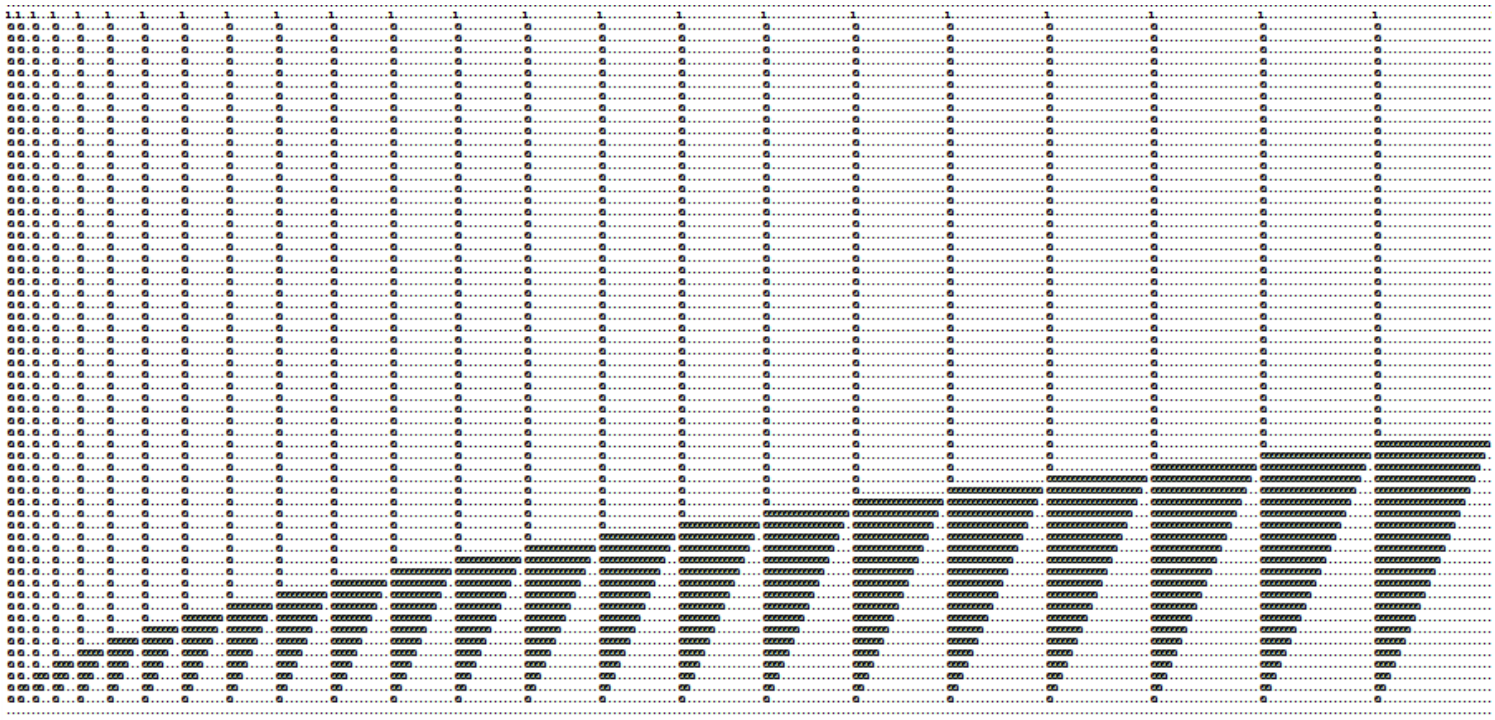
Tobacco Growing

- However we need multiple powers of 2 at the same time $\Rightarrow$ "Delay gadget" to deliver starting 1 to a "Corner gadget" for a particular power

0	0	0
0	**4**	0
0	**5**	0
0	**3**	0
0	**1**	0

Tobacco Growing

- We combine Corner gadgets for all powers with Delay gadgets corresponding to the maximal needed power 2^{59}
- We grow tobacco for 59 days and Corner gadget for power 2^i is assigned a Delay gadget of length $59 - i$
- Solution consists of Pascal triangle rows from Corner gadgets representing powers present in the target number



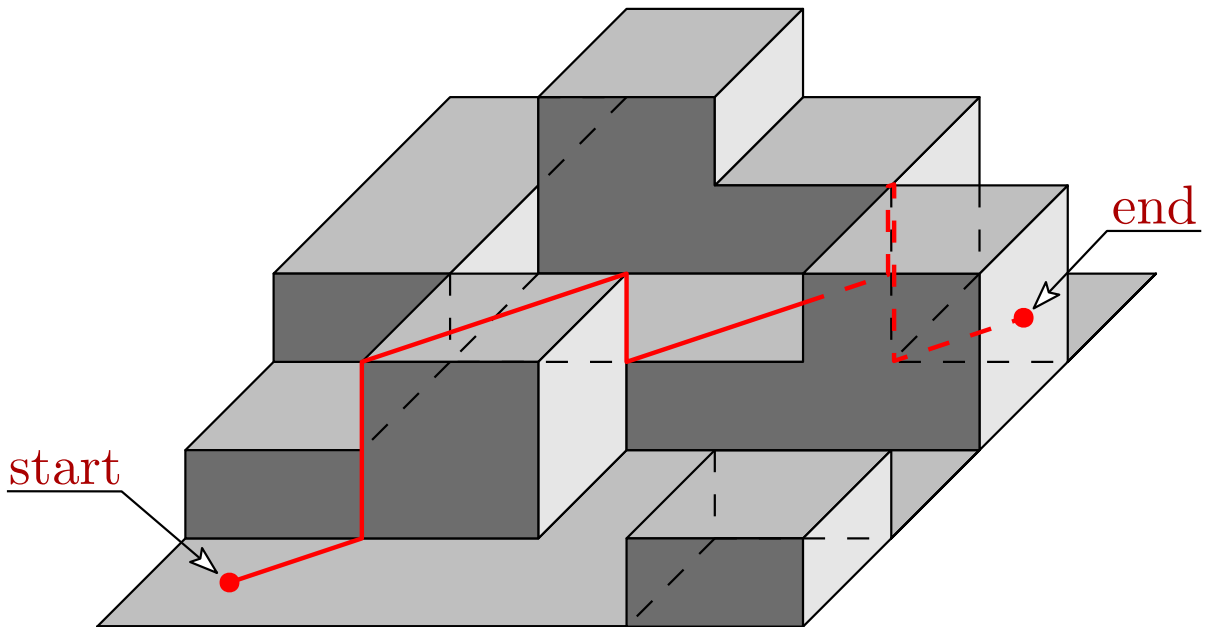
Roof

Roof

- Task: Compute the total length of a path which goes from S to E.

Roof

- Task: Compute the total length of a path which goes from S to E.



Roof

- Shortest sum of horizontal paths $\Rightarrow$ line segment form S to E from the bird's view perspective.

Roof

- Shortest sum of horizontal paths $\Rightarrow$ line segment form S to E from the bird's view perspective.
- Can be decomposed into horizontal and vertical part.

Roof

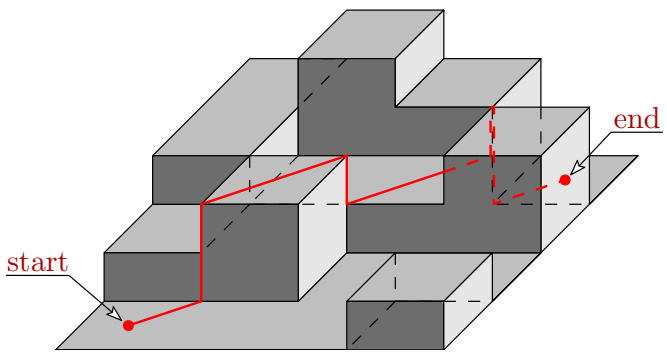
- Shortest sum of horizontal paths $\Rightarrow$ line segment form S to E from the bird's view perspective.
- Can be decomposed into horizontal and vertical part.
- Horizontal part is $\sqrt{(S_x - E_x)^2 + (S_y - E_y)^2}$

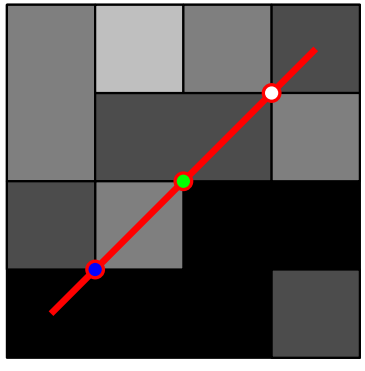
Roof

- Shortest sum of horizontal paths $\Rightarrow$ line segment form S to E from the bird's view perspective.
- Can be decomposed into horizontal and vertical part.
- Horizontal part is $\sqrt{(S_x - E_x)^2 + (S_y - E_y)^2}$
- Vertical part is composed of differences in height of "visited" cuboids.

Roof

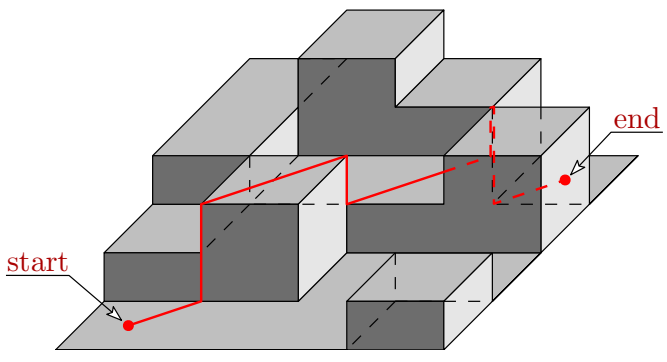
- Shortest sum of horizontal paths $\Rightarrow$ line segment form S to E from the bird's view perspective.
- Can be decomposed into horizontal and vertical part.
- Horizontal part is $\sqrt{(S_x - E_x)^2 + (S_y - E_y)^2}$
- Vertical part is composed of differences in height of "visited" cuboids.

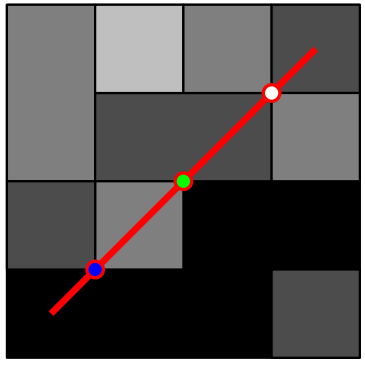




Roof

- Shortest sum of horizontal paths $\Rightarrow$ line segment form S to E from the bird's view perspective.
- Can be decomposed into horizontal and vertical part.
- Horizontal part is $\sqrt{(S_x - E_x)^2 + (S_y - E_y)^2}$
- Vertical part is composed of differences in height of "visited" cuboids.





- There is a special case when going through a corner of four cuboids.

Pickpockets

Pickpockets

- As the blocks are being layed down horizontally, we "convert" the input into consecutive horizontal blocks which we will try to fill.

Pickpockets

- As the blocks are being layed down horizontally, we "convert" the input into consecutive horizontal blocks which we will try to fill.
- Now an easy "brute-force" solution could be to to try all permutations of blocks beginning on position of last block.

Pickpockets

- As the blocks are being layed down horizontally, we "convert" the input into consecutive horizontal blocks which we will try to fill.
- Now an easy "brute-force" solution could be to to try all permutations of blocks beginning on position of last block.
- This can be optimized by bit-mask dynamic programming:
- $\mathcal{O}(T2^T + +HH)$

Offices

Offices

- Task: Given a small initial graph G, simulate expansion of the graph using the following operation multiple times:
  - Pick an ordered pair of vertices $q = (v_i, v_j)$ which do not share an edge
  - Create a new vertex v_{n+1} and connect it to each of the existing vertices v_a according to the following table.

$v_j \setminus v_i$	$\emptyset$	O	Q
$\emptyset$	$\emptyset$	O	Q
O	Q	$\emptyset$	O
Q	O	Q	$\emptyset$

- The column specifies the type of connection that v_i has to v_a, the row specifies the type of connection that v_j has to v_a and the inner cell is the resulting type of connection that v_{n+1} will have to v_a.
- After a new vertex is created, compute the sum of distances of reachable vertices from vertex 0.

Offices

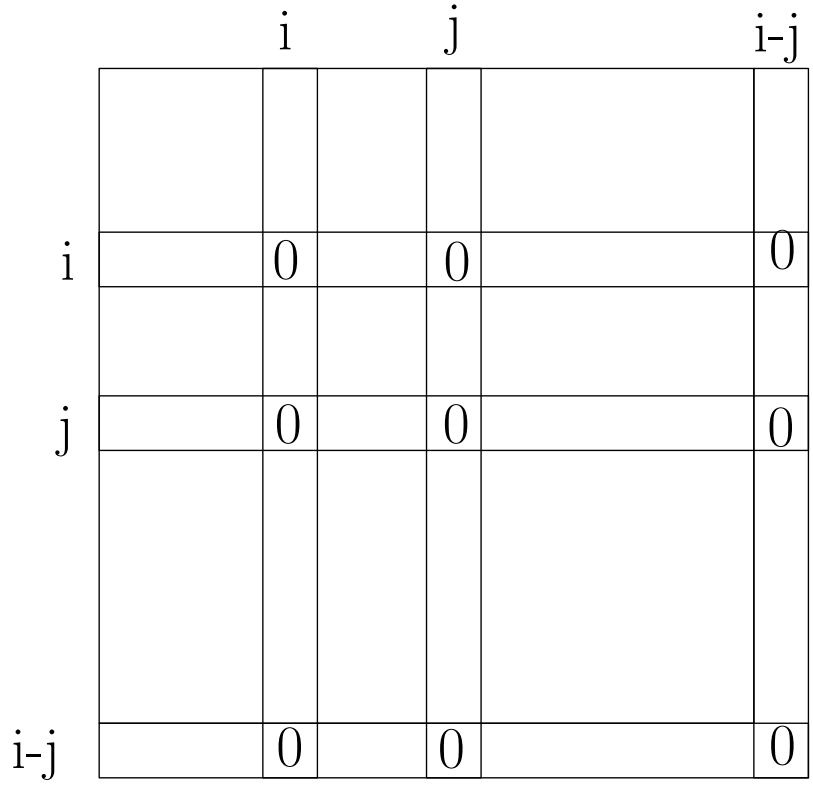
- Suppose O $= 1$ and Q $= 2$. Then the operation is isomorphic to subtraction in Z_3.

$v_j \setminus v_i$	$\emptyset$	O	Q
$\emptyset$	$\emptyset$	O	Q
O	Q	$\emptyset$	O
Q	O	Q	$\emptyset$

$v_j \setminus v_i$	0	1	2
0	0	1	2
1	2	0	1
2	1	2	0

Offices

- Consider the incidence matrix $M \in Z_3^{n \times n}$ of G. Note how the operation extends M.
  - The new row is a linear combination of the original rows.
  - The condition that the detectives come from offices not connected by a cable guarantees that we create no loops.



Offices

- Therefore the row of every newly created vertex is a linear combination of the first $N \leq 5$ initial rows.
- The matrix has at most 3^N different rows.
- Therefore at most 3^N possible neighborhoods exist in the graph.
- If two vertices have the same neighborhood, the shortest paths to other vertices will not change
  - The case when the neighborhood is equal to the one of v_0 must be handled separately, as the paths start only from v_0.
- After adding a vertex v with a new neighborhood, check if some other vertex already has the same neighborhood.
  - If another vertex v_i already has the same neighborhood, to the resulting sum of distances simply add the distance of v_i
  - Otherwise recalculate the shortest paths e.g. using Dijkstra's algorithm.

Offices

- To find the vertex with the same neighborhood, for each v keep the track of coefficients $\alpha_{v,1}, \cdots, \alpha_{v,N}$ which determine the linear combination of the initial N rows that results in the row for v.
- The resulting complexity:
- We add a new neighborhood vertex at most 3^N times, each time recalculating the distances with complexity $\mathcal{O}(3^{2N} \log(3^N))$.
- The total complexity is $\mathcal{O}(27^N N)$.
- With $N \leq 5$ fits into the timelimit.

Bank

Bank

- Task: Choose a role and beat the judge in a game.

Given a tree graph ($\deg(v) \neq 2$) decide which role to pick. Then:

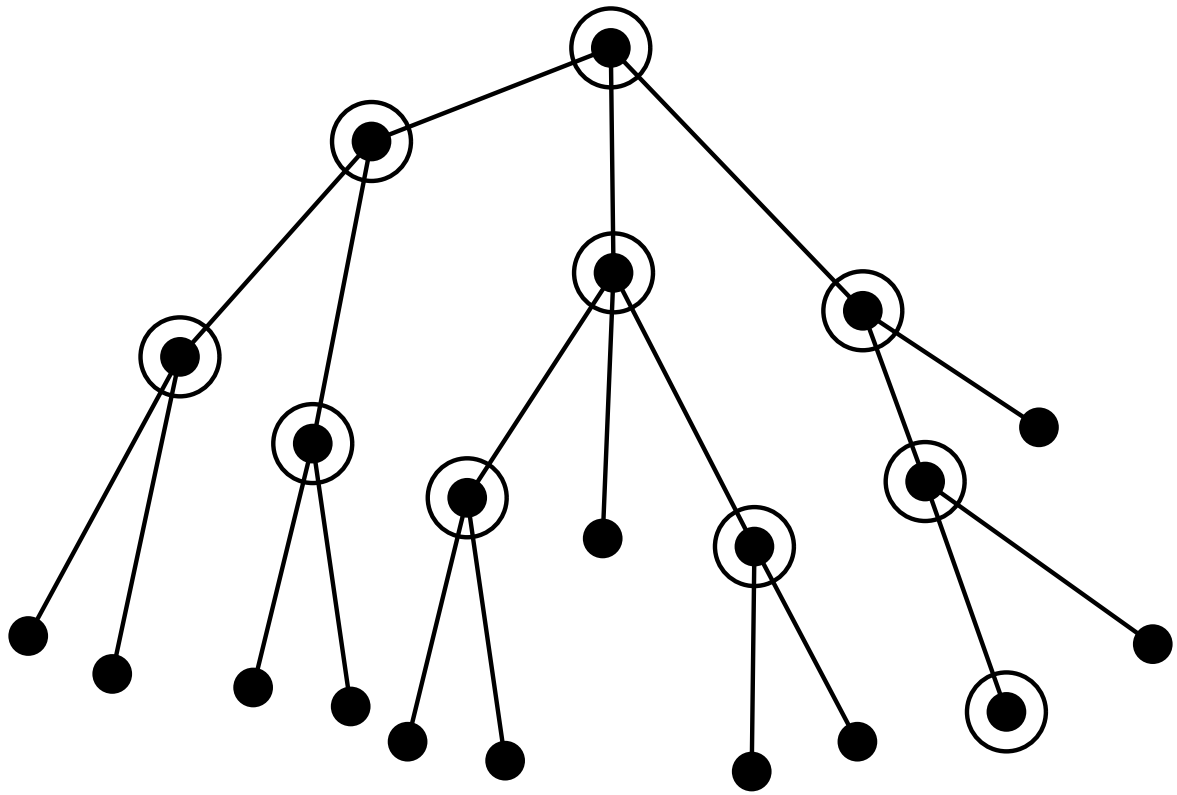
- K detectives are placed on the tree by the defender.
- Each turn:
  - Attacker announces which vertex of the tree is attacked.
  - Defender moves each detective over at most one edge.
  - If a detective is not present in the attacked vertex, then the attacker wins.
- If the defender holds for 365 turns, then he wins.

Correct approach – threshold is: number of inner vertices $+1$

- `DEFEND` every inner vertex and 1 leaf
- `ATTACK` can propagate non-occupied inner vertex closer to the leaves

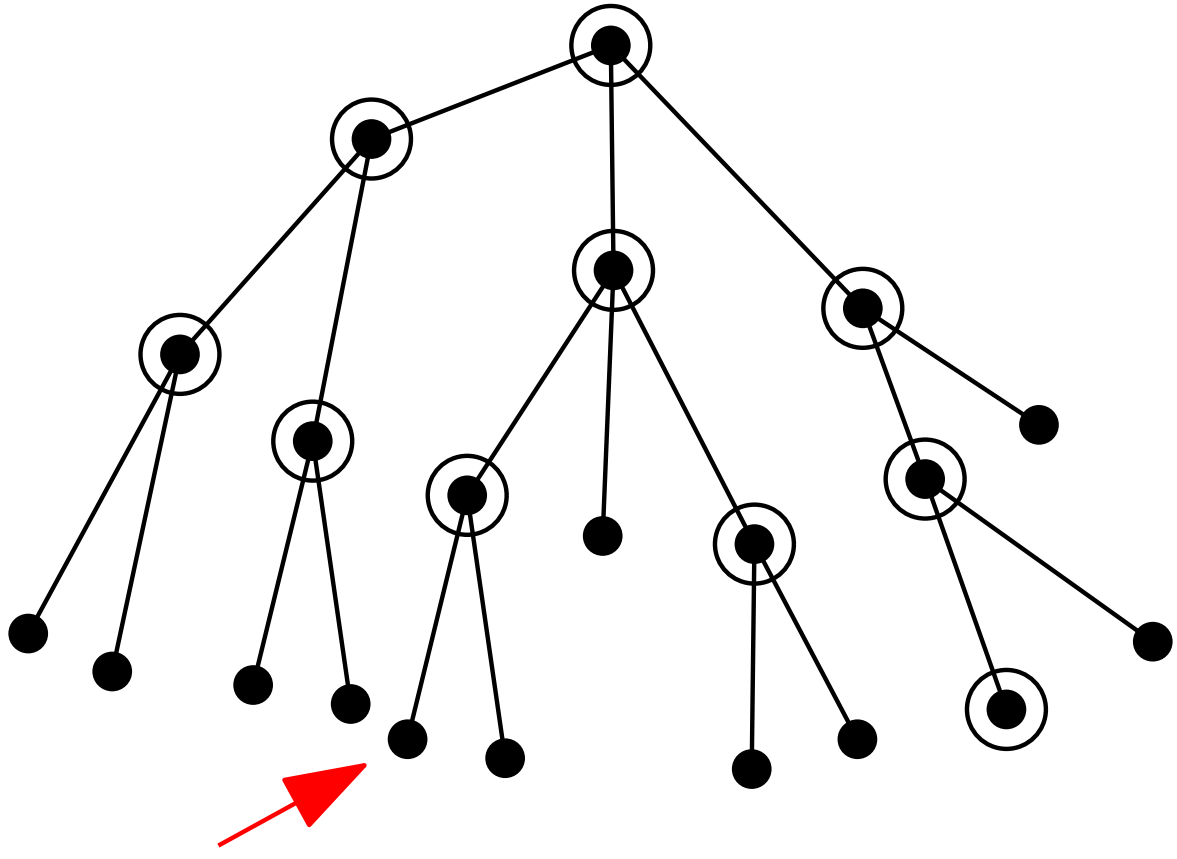
Bank

`DEFEND` every inner vertex and 1 leaf



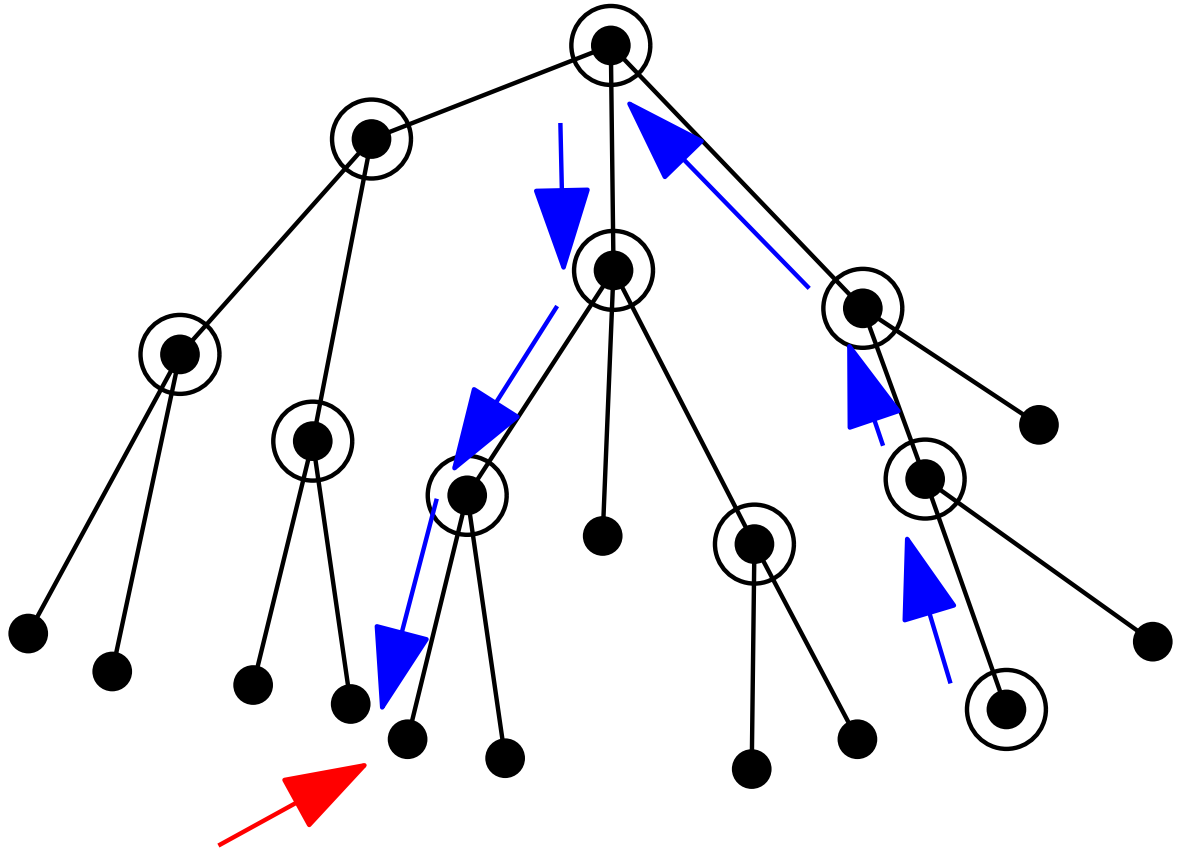
Bank

`DEFEND` every inner vertex and 1 leaf



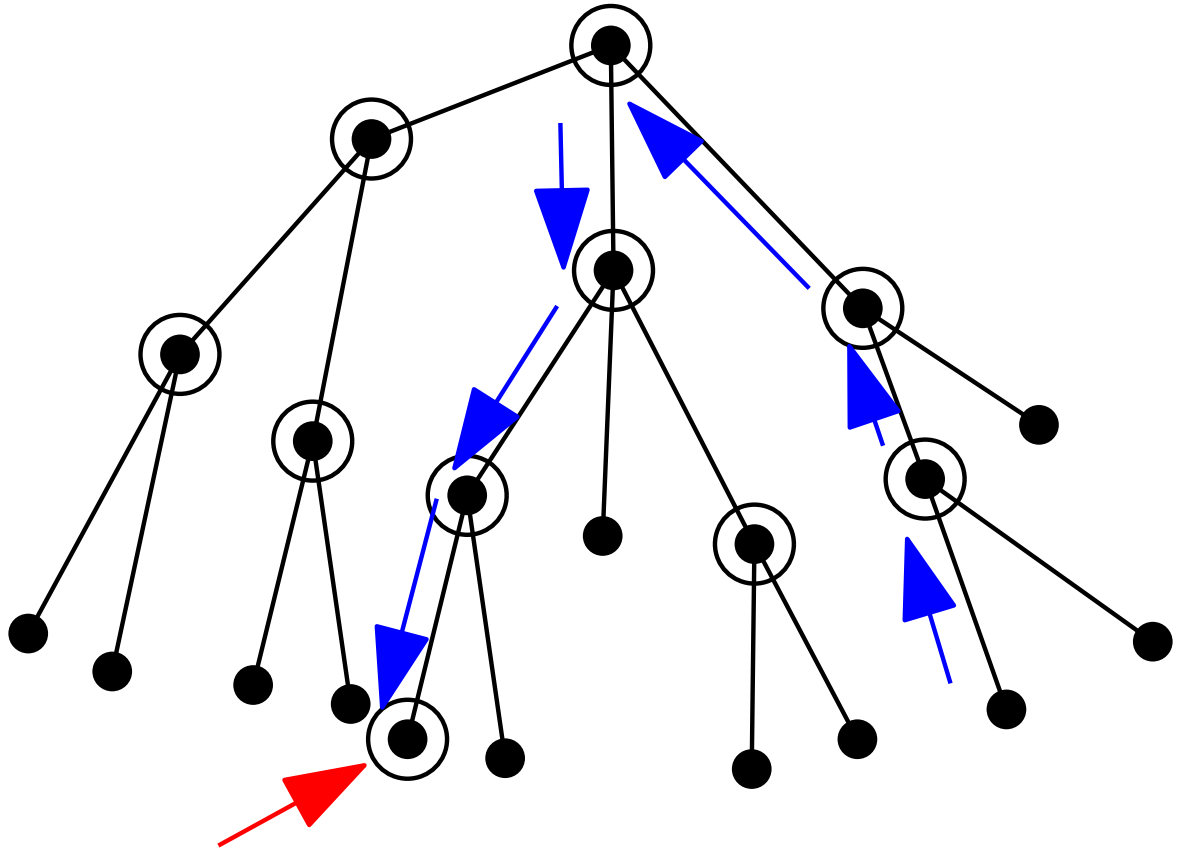
Bank

`DEFEND` every inner vertex and 1 leaf



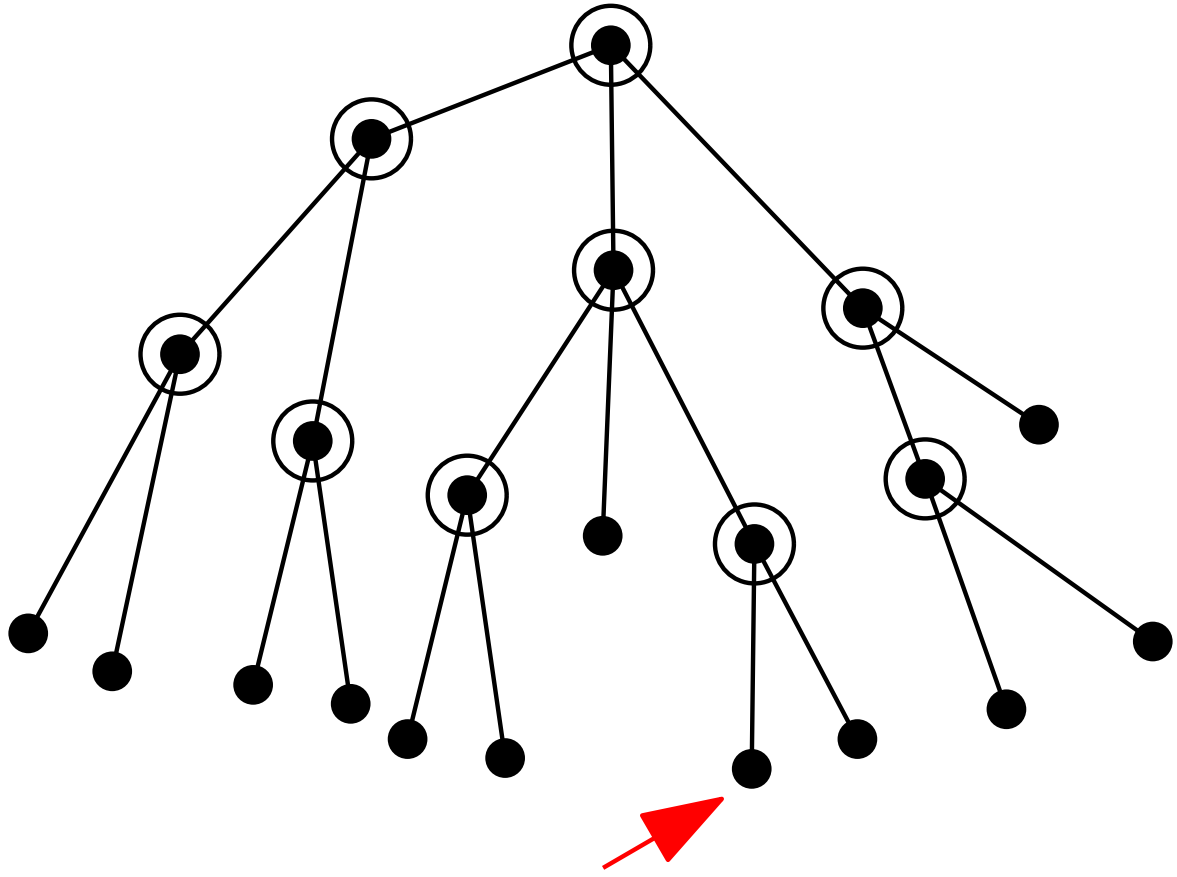
Bank

`DEFEND` every inner vertex and 1 leaf



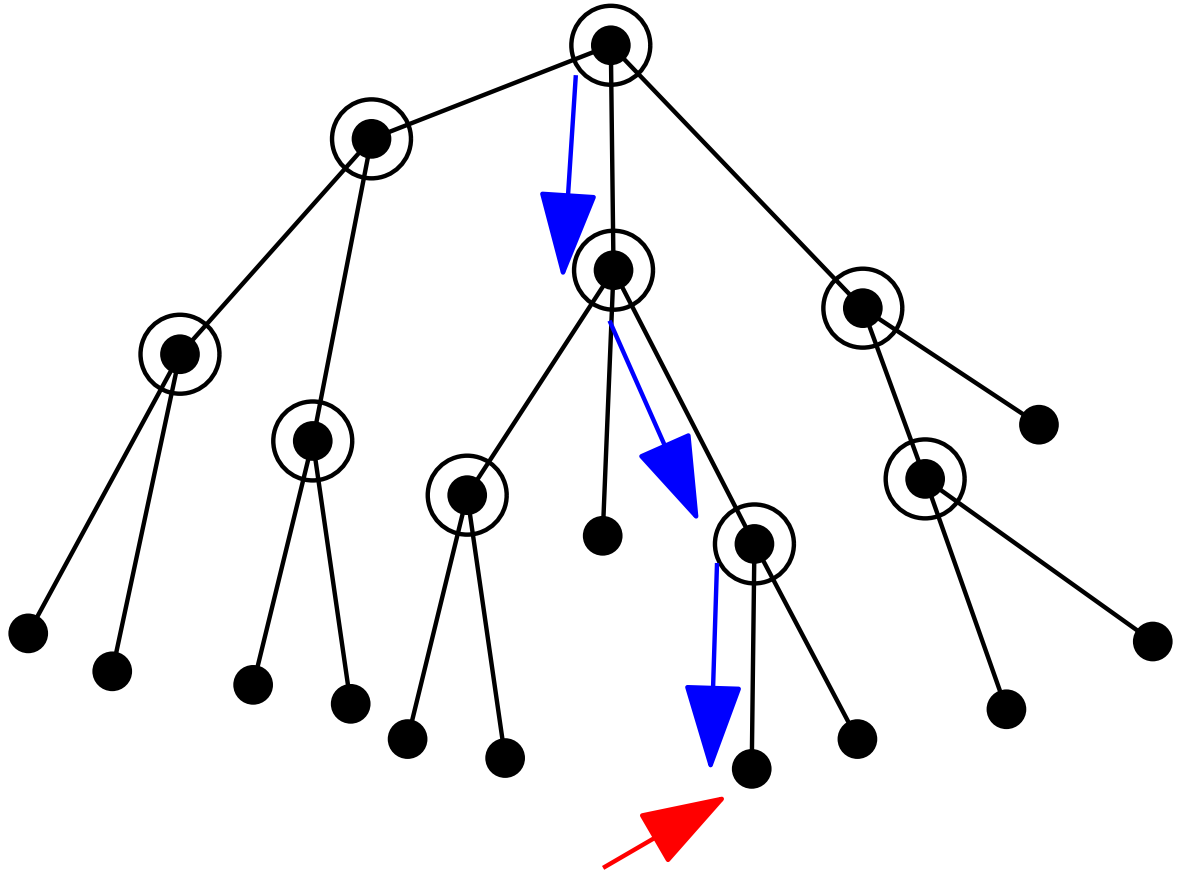
Bank

ATTACK can propagate non-occupied inner vertex closer to the leaves



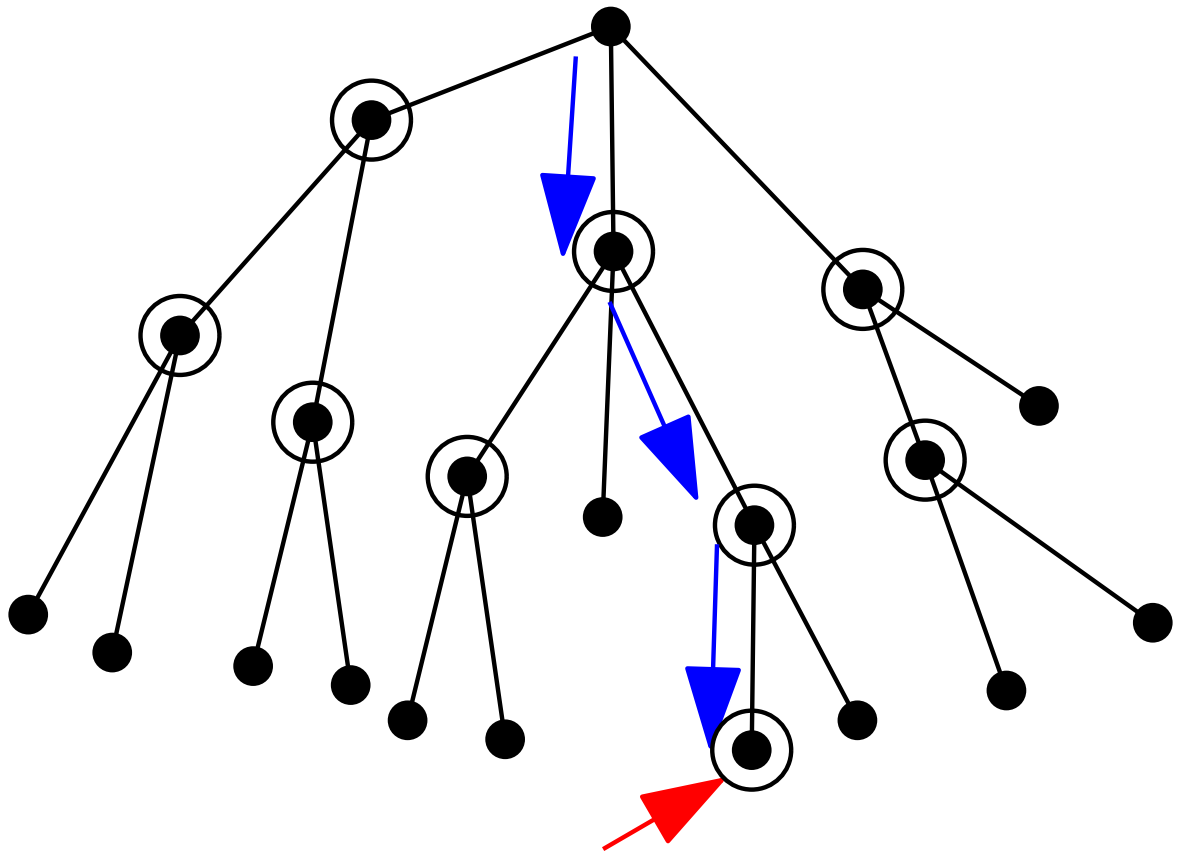
Bank

ATTACK can propagate non-occupied inner vertex closer to the leaves



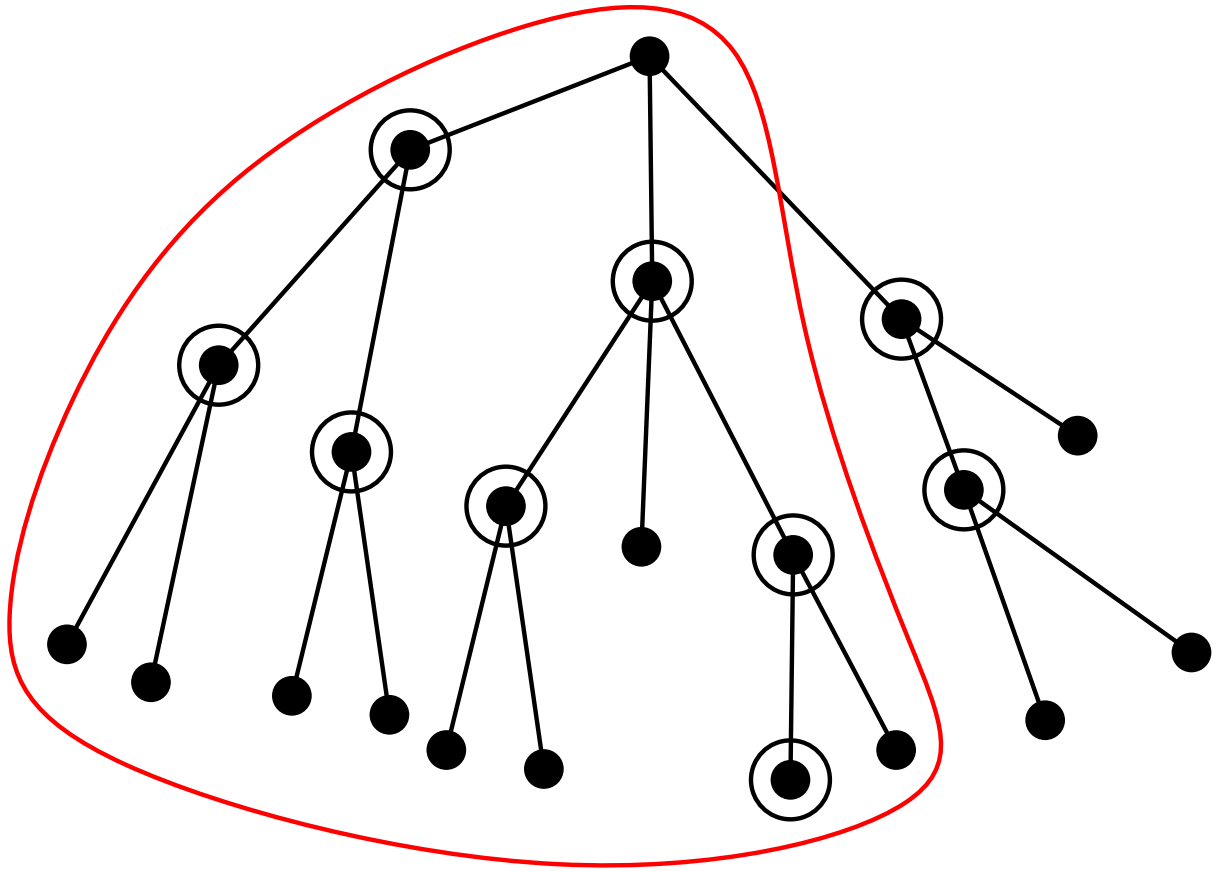
Bank

ATTACK can propagate non-occupied inner vertex closer to the leaves



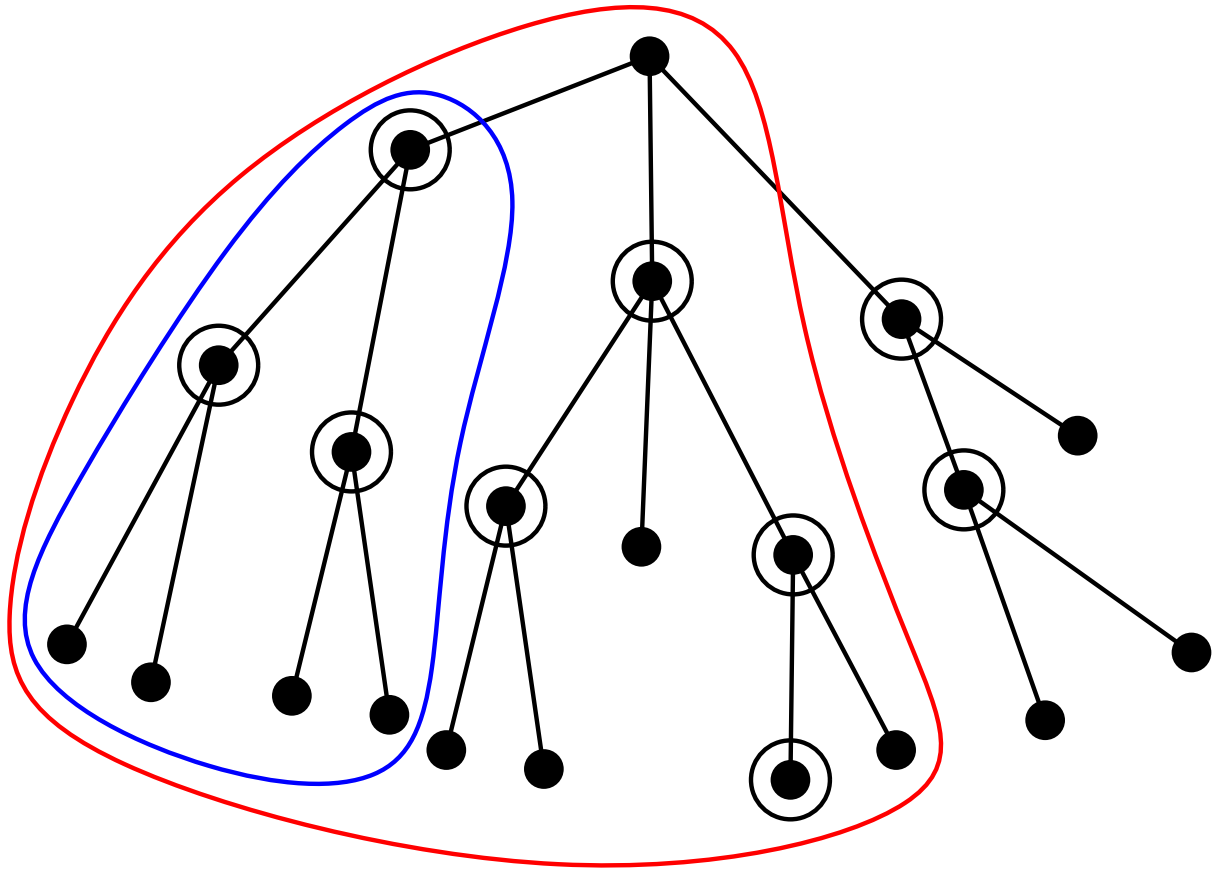
Bank

ATTACK can propagate non-occupied inner vertex closer to the leaves



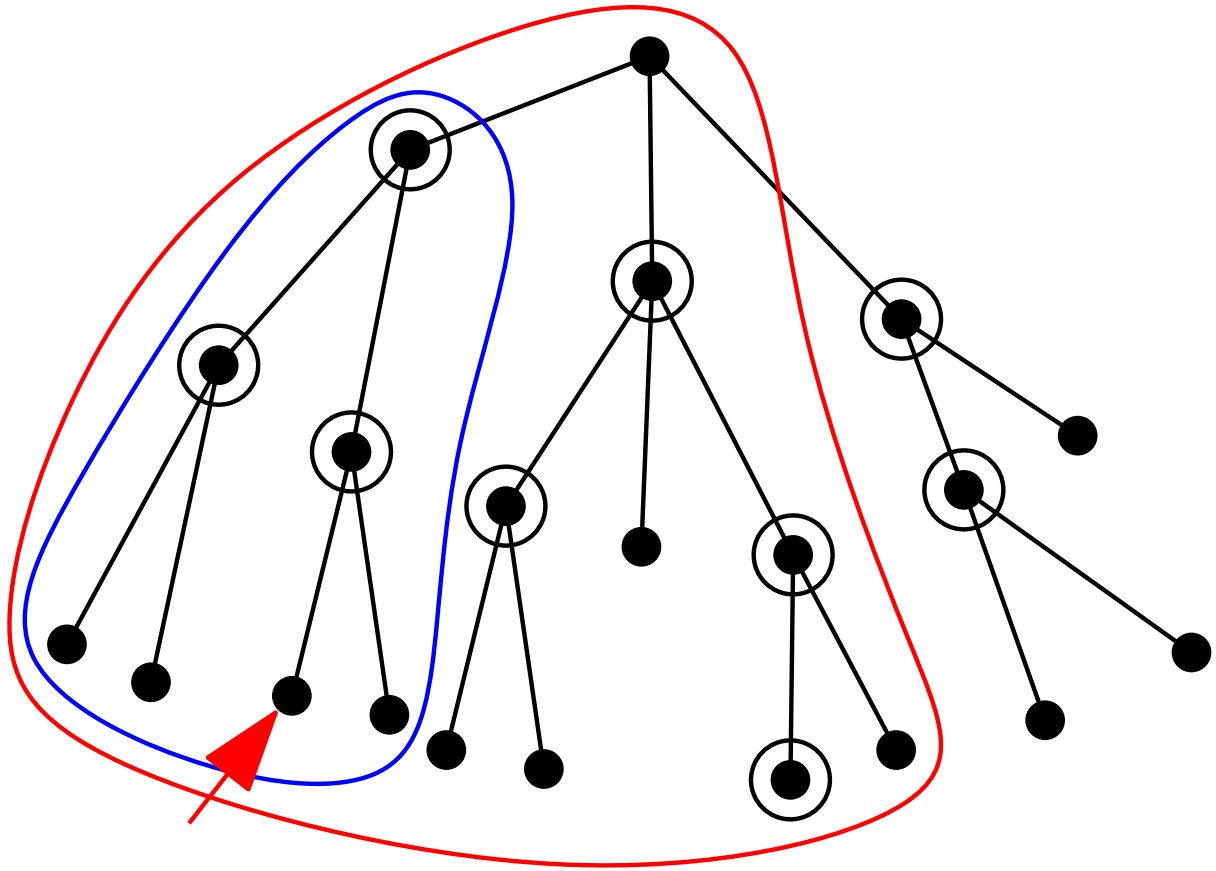
Bank

ATTACK can propagate non-occupied inner vertex closer to the leaves



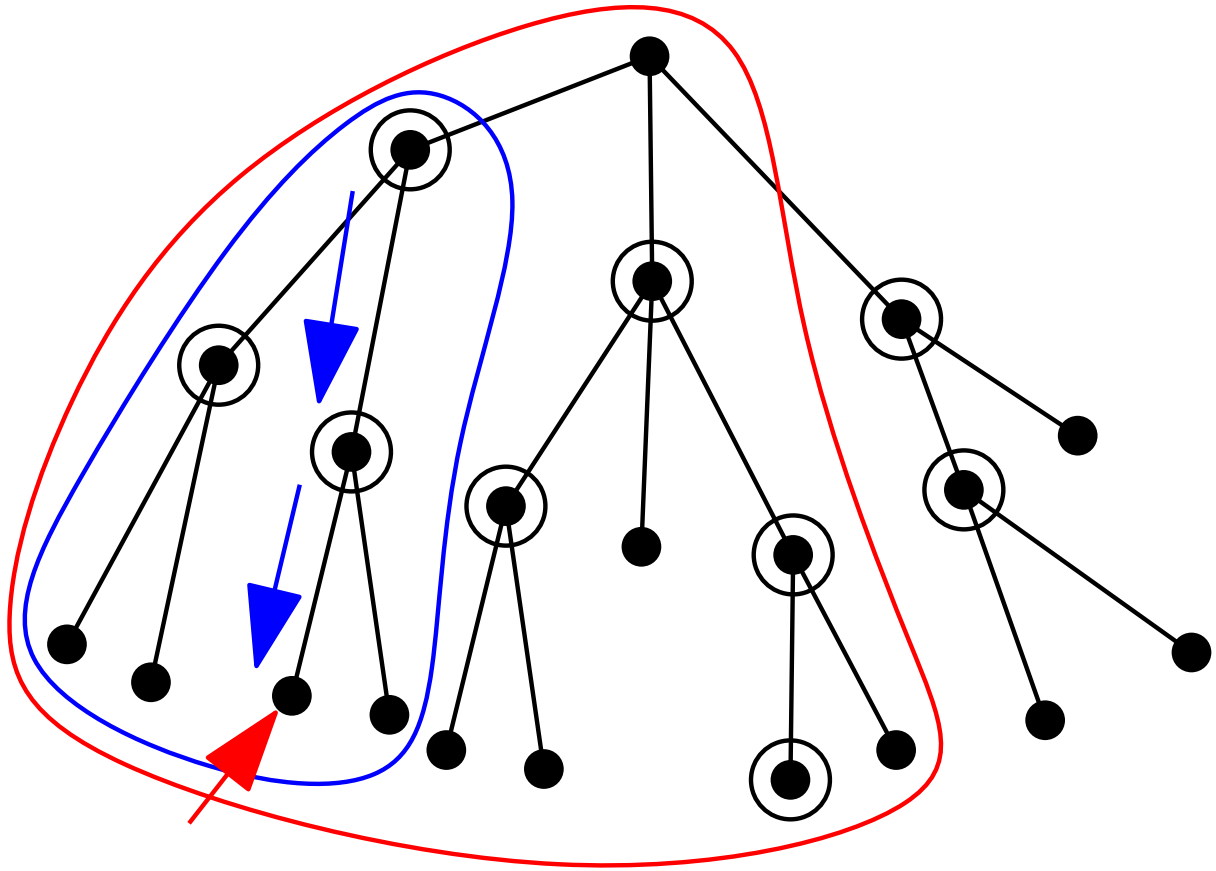
Bank

ATTACK can propagate non-occupied inner vertex closer to the leaves



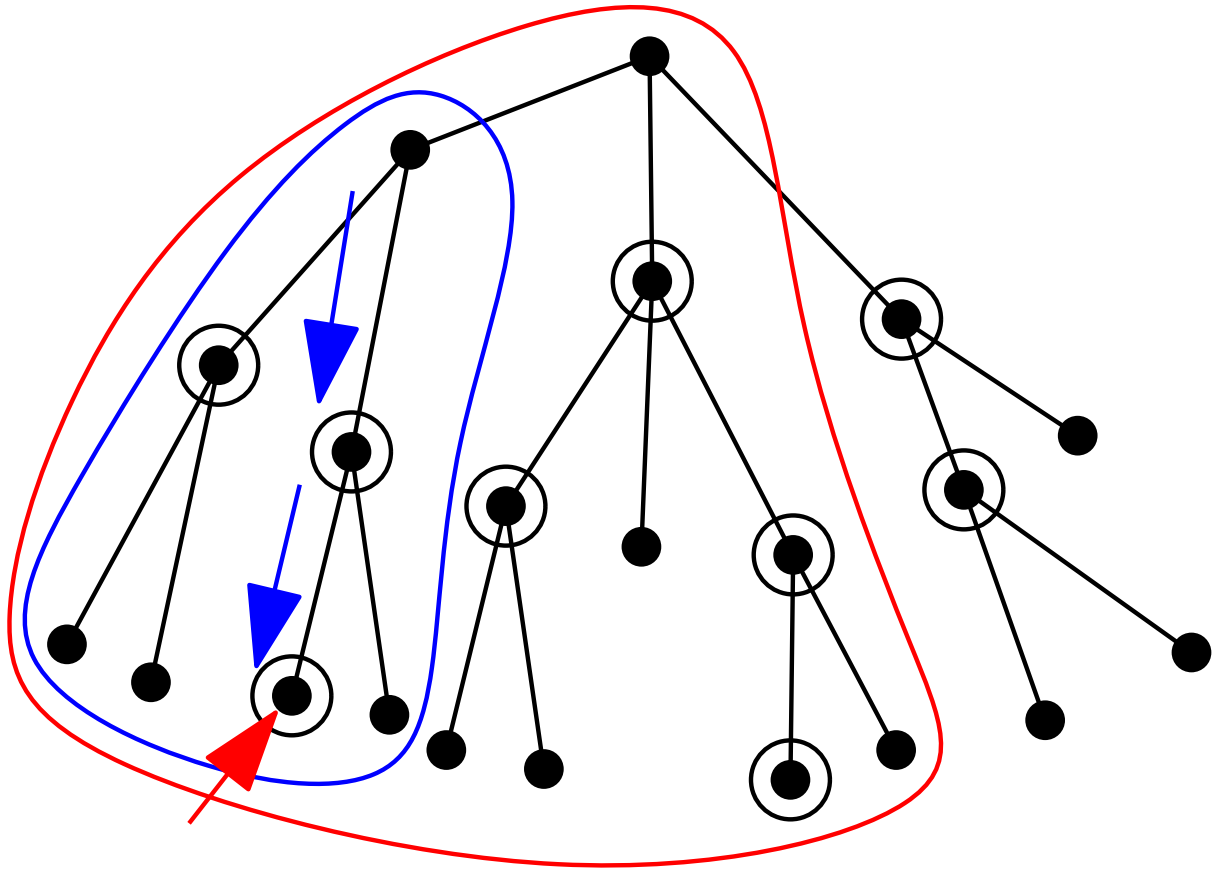
Bank

ATTACK can propagate non-occupied inner vertex closer to the leaves



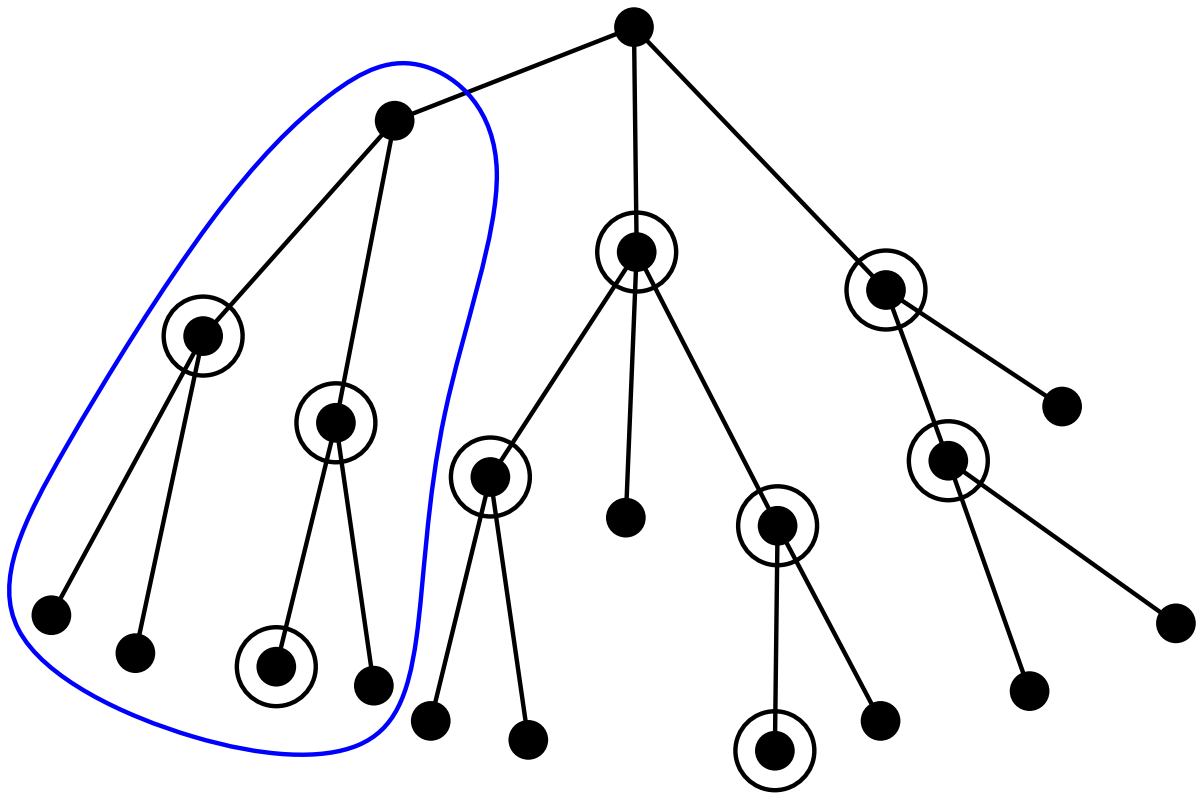
Bank

ATTACK can propagate non-occupied inner vertex closer to the leaves



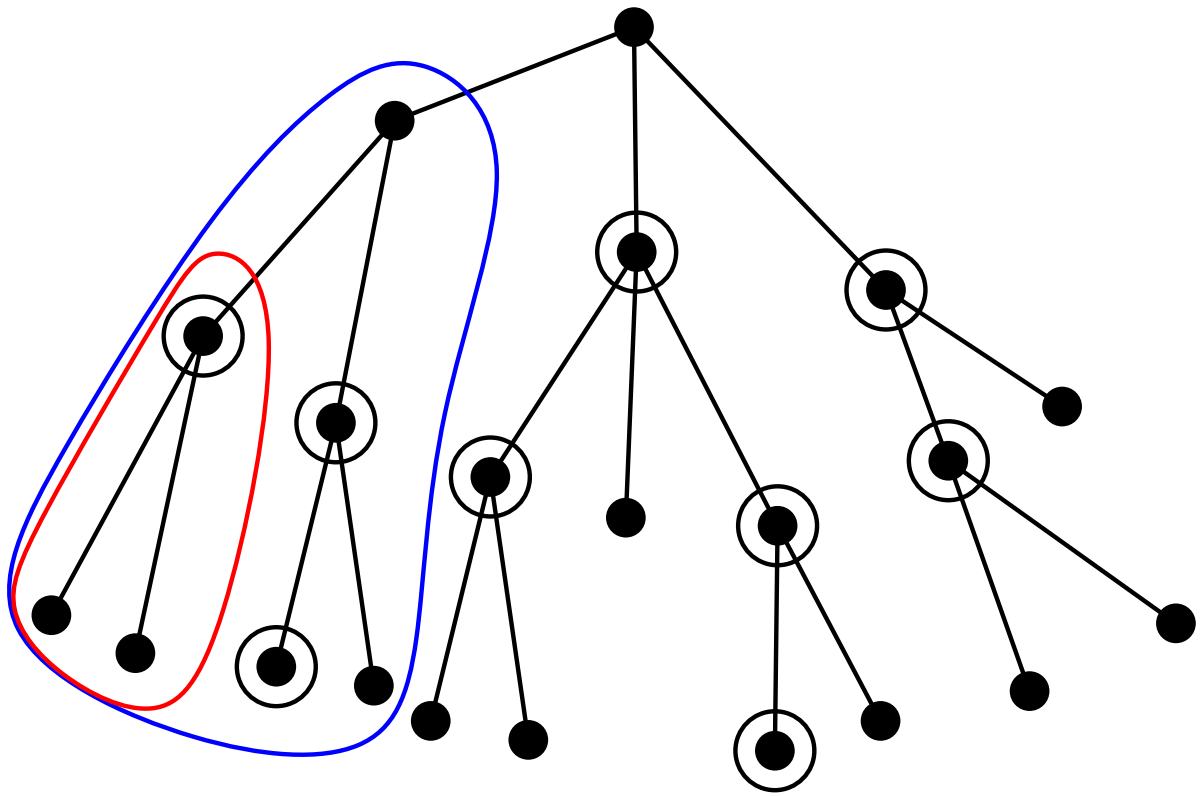
Bank

ATTACK can propagate non-occupied inner vertex closer to the leaves



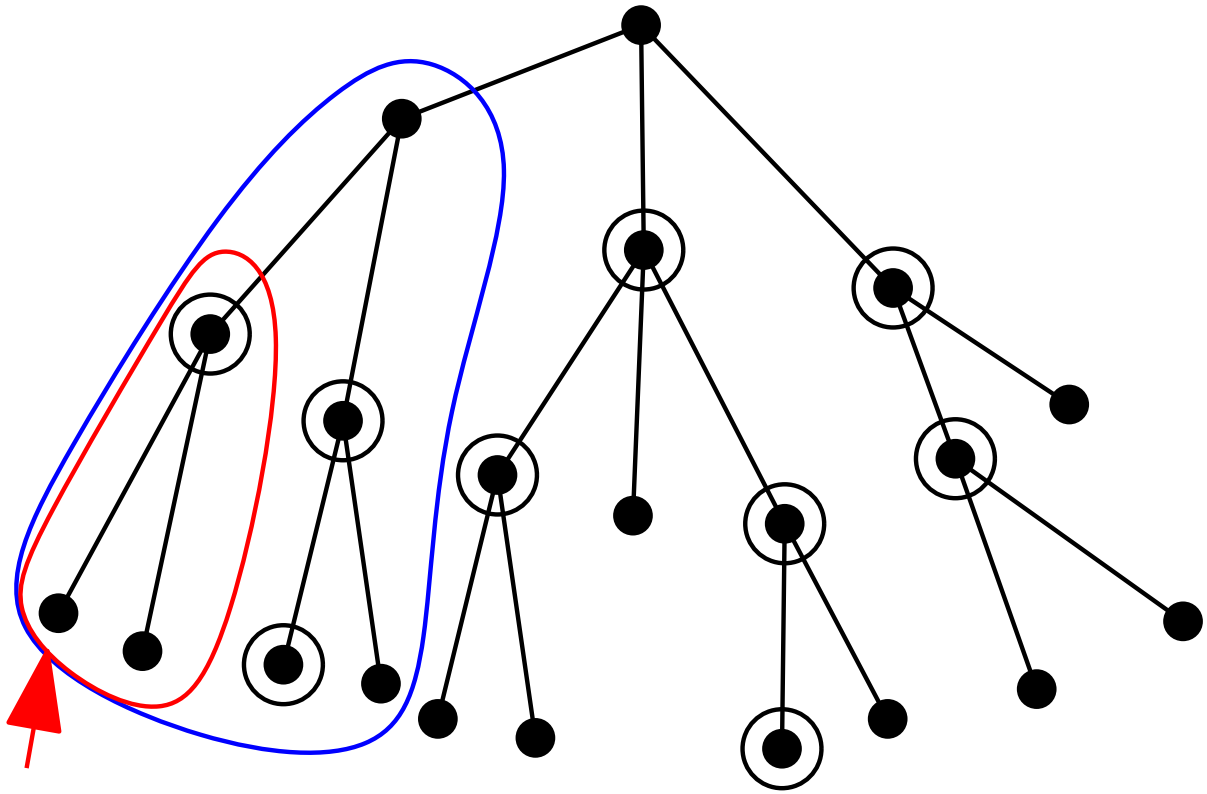
Bank

ATTACK can propagate non-occupied inner vertex closer to the leaves



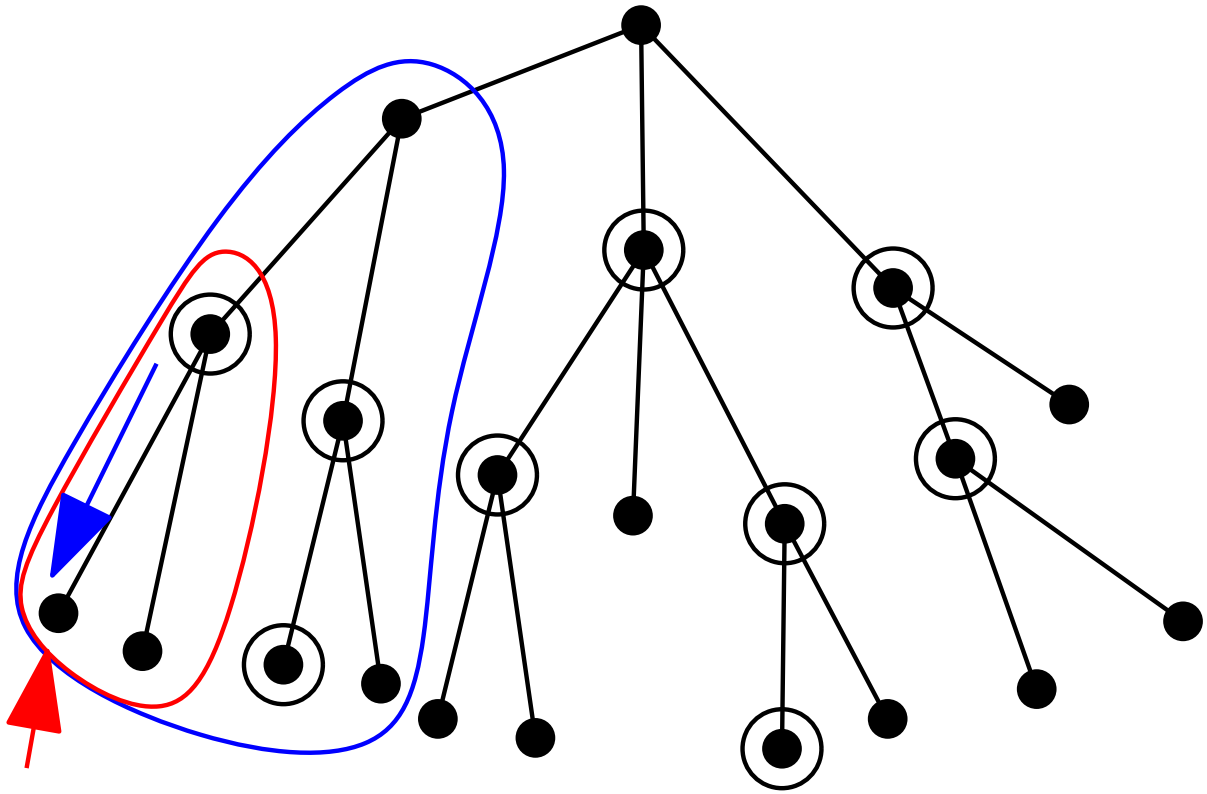
Bank

ATTACK can propagate non-occupied inner vertex closer to the leaves



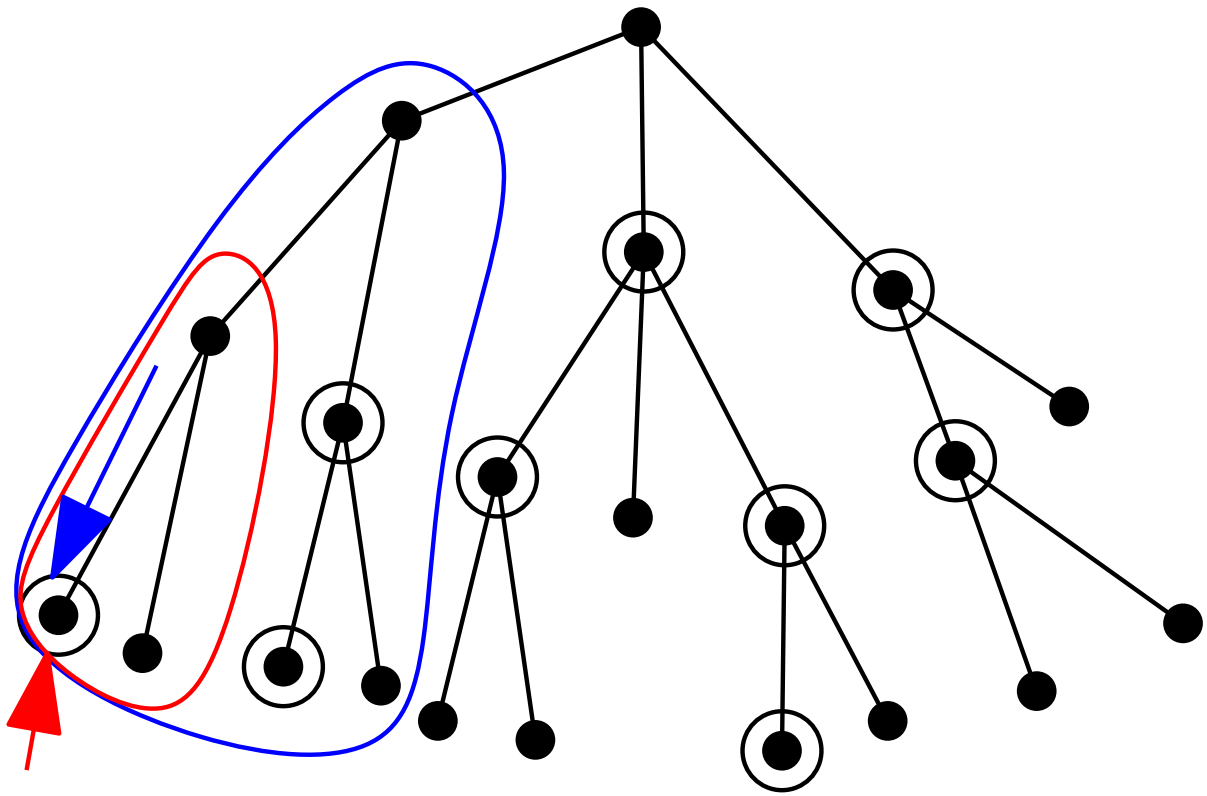
Bank

ATTACK can propagate non-occupied inner vertex closer to the leaves



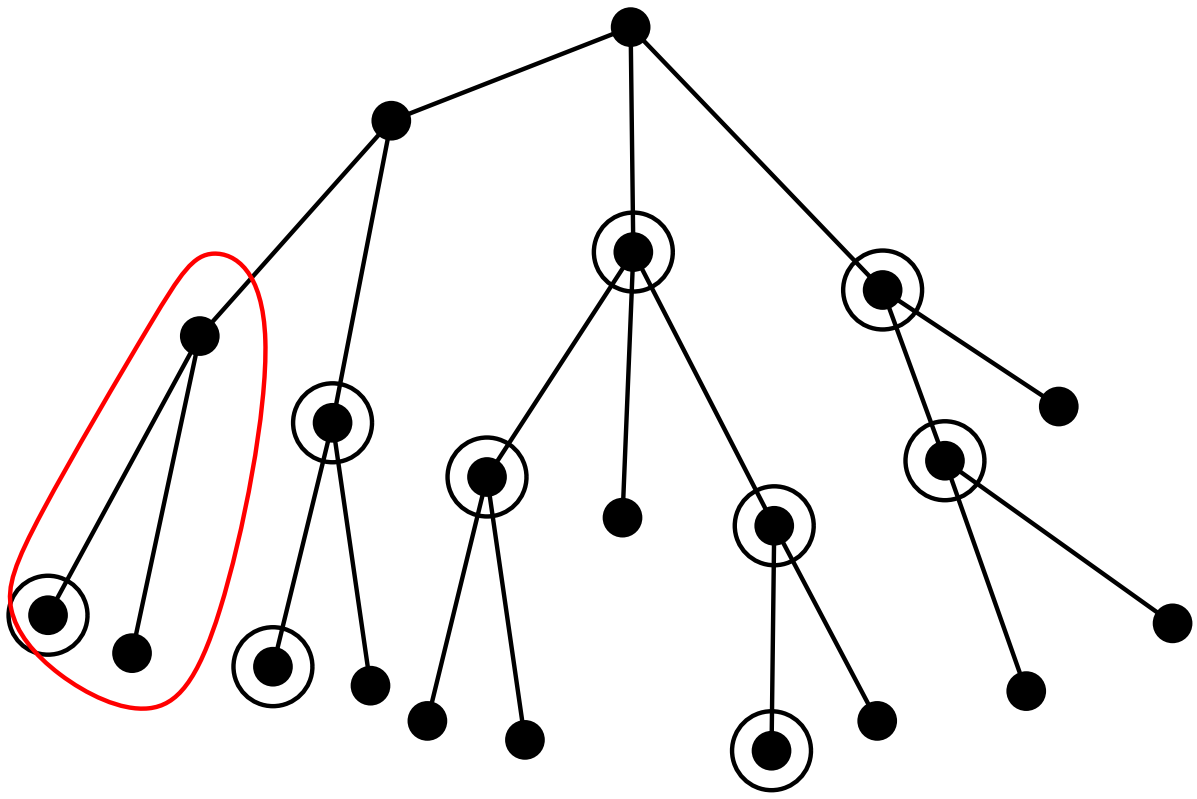
Bank

ATTACK can propagate non-occupied inner vertex closer to the leaves



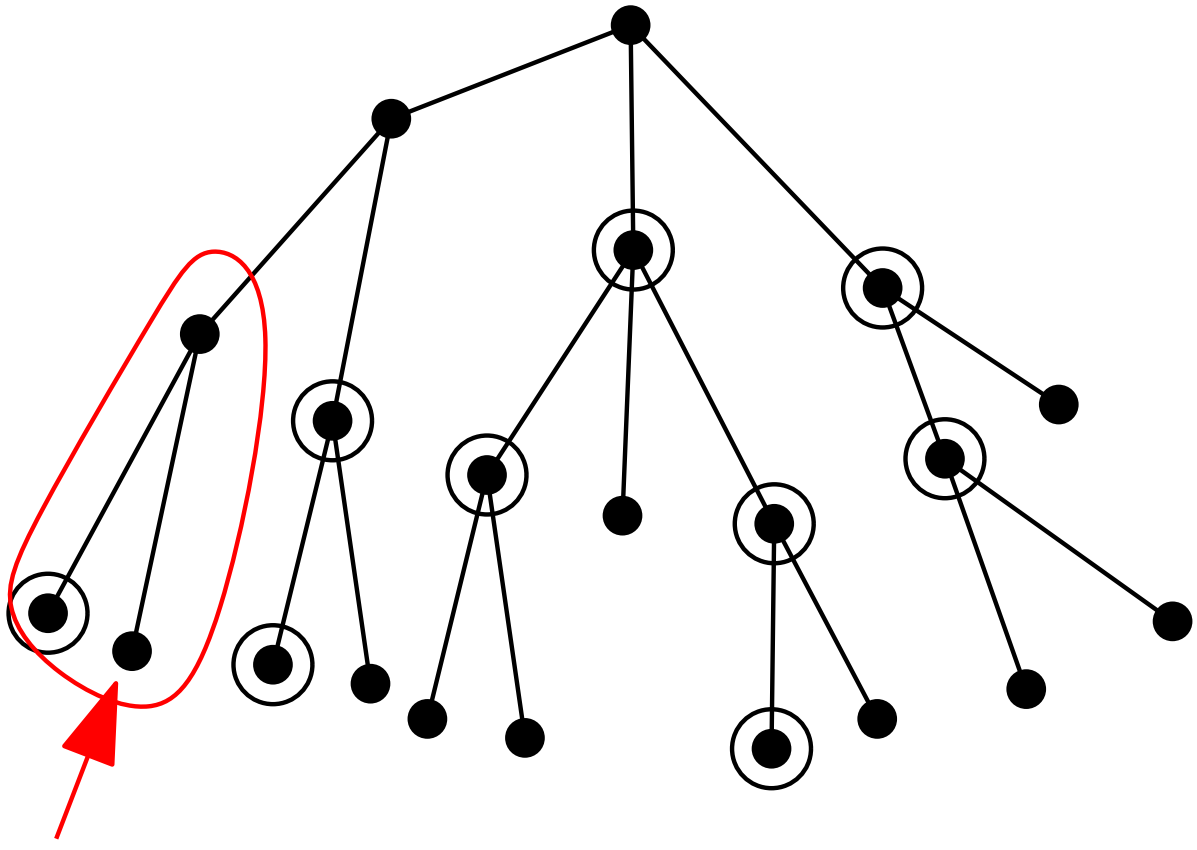
Bank

ATTACK can propagate non-occupied inner vertex closer to the leaves



Bank

ATTACK can propagate non-occupied inner vertex closer to the leaves



Screamers

Screamers

- Task: You are given an array of edges in a graph, and a batch of queries. Queries consist of intervals and for each interval we need to find out the number of subarrays (within bounds of the interval) of edges such that all the edges in the subarray don't form a cycle.
- Sadly there are far too many options to try out one by one.

Screamers

Screamers

- Firstly lets precompute for each possible start of the subarray the farthest point where it might end so that there is no cycle induced by the edges in the subarray (it is a valid subarray). This is equal to the number of valid subarrays starting at the given start.
- One of the ways how to do that is to use *linked-cut-tree*.
- Use two pointer technique, add edges one by one at the end of the current interval and remove all edges at the beginning while the edges induce a cycle.

Screamers

Screamers

- Secondly lets do a prefix sum on previous step's result (the number of valid intervals starting at each position).
- Finally for each query lets find such a position i such that all the longest valid intervals starting after i end after the query's endpoint. All the subarrays that start up to i can be counted using the previously mentioned prefix sum.
- The rest of the subarrays for the query is the number of subarrays inside the interval starting at $i + 1$ and ending at the query's end.

Art Transaction

Art Transaction

- Chupakabra is considered an animal.

Art Transaction

Art Transaction

- Task: Follow "a few simple rules".

Art Transaction

- Task: Follow "a few simple rules".
- Watch out for tiny nuances.
- Rules "House view up" and "House view down" are almost the same, except for used constant.
- On the other hand rules "Drake/grill" and "Grill/drake" are almost the same, but not quite.

Art Transaction

- Task: Follow "a few simple rules".
- Watch out for tiny nuances.
- Rules "House view up" and "House view down" are almost the same, except for used constant.
- On the other hand rules "Drake/grill" and "Grill/drake" are almost the same, but not quite.
- The clarification request whether the "House view up / down" rules are meant to be the same were asked X times.

Art Transaction

- Task: Follow "a few simple rules".
- Watch out for tiny nuances.
- Rules "House view up" and "House view down" are almost the same, except for used constant.
- On the other hand rules "Drake/grill" and "Grill/drake" are almost the same, but not quite.
- The clarification request whether the "House view up / down" rules are meant to be the same were asked X times.

Thank you for participating!