International school in informatics (ISIJ)

Innopolis, July 2, 2019.

Task "Sorting by ranking"

- Task idea — Michael Kinder
- Task solution — Michael Kinder

Task formulation

- A graph of n vertices is given. Each vertex has a rating positive integer.
- It is allowed to swap the rankings of two adjacent vertices of the graph.
- You have to sort the original set of ratings in ascending order, rearranging the ratings of only two vertices at each step. If this is not possible, put `-1`.

Main themes of the task:

- path in the graph;
- recursion;
- sorting with limits.

Solution. Subtask 1 ($n \leqslant 4$)

- First group of 14 tests.
- A simple enumerations of possible options;
- Accurate programming of the list of presented pairs (no more than eight steps).

Solution. Subtask 3 (*boss*)

- We construct a graph of subordination of employees (adjacency matrix);
- There is a vertex of graph B (boss) adjacent to all other vertices of the graph in subtask 3;
- Each sorting step is a rearrangement of two employees with x and y ratings in *one* or *three* steps, the position of the others does not change.

$x \ldots y \ldots B$	$x \quad B$
$B \ldots y \ldots x$	$y \quad B$
$b \ldots B \ldots x$	$x \quad B$
$y \ldots x \ldots B$	

Solution. Subtask 3 (*boss*)

Realization of algorithm:

- by the adjacency matrix of the graph we determine the vertex B adjacent to all other vertices;
- find the vertex with the *lowest* rating d_1 and use $\leqslant 3$ permutations to move it to the first place in the array of ratings d;
- then find the vertex with the next highest rating d_2 and put it in second place in the way, mentioned above.
- And so on...
- The complexity of the algorithm — $O(n \log n)$. You can also use slower sorting algorithms.

Solution. Subtask 3 (*boss*)

Realization of algorithm:

- by the adjacency matrix of the graph we determine the vertex B adjacent to all other vertices;
- find the vertex with the *lowest* rating d_1 and use $\leqslant 3$ permutations to move it to the first place in the array of ratings d;
- then find the vertex with the next highest rating d_2 and put it in second place in the way, mentioned above.
- And so on...
- The complexity of the algorithm — $O(n \log n)$. You can also use slower sorting algorithms.

Solution. Subtask 3 (*boss*)

Realization of algorithm:

- by the adjacency matrix of the graph we determine the vertex B adjacent to all other vertices;
- find the vertex with the *lowest* rating d_1 and use $\leqslant 3$ permutations to move it to the first place in the array of ratings d;
- then find the vertex with the next highest rating d_2 and put it in second place in the way, mentioned above.
- And so on...
- The complexity of the algorithm — $O(n \log n)$. You can also use slower sorting algorithms.

Solution. Subtask 3 (*boss*)

Realization of algorithm:

- by the adjacency matrix of the graph we determine the vertex B adjacent to all other vertices;
- find the vertex with the *lowest* rating d_1 and use $\leqslant 3$ permutations to move it to the first place in the array of ratings d;
- then find the vertex with the next highest rating d_2 and put it in second place in the way, mentioned above.
- And so on...
- The complexity of the algorithm — $O(n \log n)$. You can also use slower sorting algorithms.

Solution. Subtask 3 (*boss*)

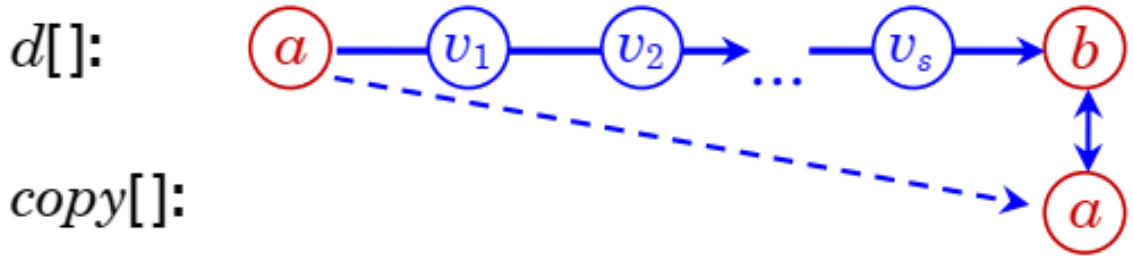
Realization of algorithm:

- by the adjacency matrix of the graph we determine the vertex B adjacent to all other vertices;
- find the vertex with the *lowest* rating d_1 and use $\leqslant 3$ permutations to move it to the first place in the array of ratings d;
- then find the vertex with the next highest rating d_2 and put it in second place in the way, mentioned above.
- And so on...
- The complexity of the algorithm — $O(n \log n)$. You can also use slower sorting algorithms.

Solution. Subtask 2 ($n \leqslant 20$)

- Consider a graph, the vertices are employee ratings, and the edges are the subordination relationships between them.
- Create a copy vector copy[] of the original array of ratings and sort the numbers in ascending order.
- Now we know where in the graph we need to move each number of the original set.

Let a be a arbitrary vertex of the graph and b be the vertex of the graph to which you want to move the vertex a.



Subtask 2. Recursive solution $O(n^2 \cdot 2^s)$

- Using the bfs-algoritm we find the shortest path from a to b. If there is no such path, output -1.
- Let this path consist of $s+1$ edges

$$a - v_1 - v_2 - \ldots - v_s - b.$$

- For example, if the path length is 1, employees a and a can be transplanted in one step; if it is 2 — in 3 steps.
- Let k be the number of steps for permutation a and b between which there are s edges. Consider a path from $s+1$ edges:

$$(a - v_1 - v_2 - \ldots - v_s) - b.$$

- Select the part from a to v_s and rearrange the numbers at vertices a and v_s (to k steps):

Subtask 2. Recursive solution $O(n^2 \cdot 2^s)$

- Using the bfs-algoritm we find the shortest path from a to b. If there is no such path, output -1.
- Let this path consist of $s+1$ edges

$$a - v_1 - v_2 - \ldots - v_s - b.$$

- For example, if the path length is 1, employees a and a can be transplanted in one step; if it is 2 — in 3 steps.
- Let k be the number of steps for permutation a and b between which there are s edges. Consider a path from $s+1$ edges:

$$(a - v_1 - v_2 - \ldots - v_s) - b.$$

- Select the part from a to v_s and rearrange the numbers at vertices a and v_s (to k steps):

Subtask 2. Recursive solution $O(n^2 \cdot 2^s)$

- Using the bfs-algoritm we find the shortest path from a to b. If there is no such path, output -1.
- Let this path consist of $s+1$ edges

$$a - v_1 - v_2 - \ldots - v_s - b.$$

- For example, if the path length is 1, employees a and a can be transplanted in one step; if it is 2 — in 3 steps.
- Let k be the number of steps for permutation a and b between which there are s edges. Consider a path from $s+1$ edges:

$$(a - v_1 - v_2 - \ldots - v_s) - b.$$

- Select the part from a to v_s and rearrange the numbers at vertices a and v_s (to k steps):

Subtask 2. Recursive solution $O(n^2 \cdot 2^s)$

- Using the bfs-algoritm we find the shortest path from a to b. If there is no such path, output -1.
- Let this path consist of $s+1$ edges

$$a - v_1 - v_2 - \ldots - v_s - b.$$

- For example, if the path length is 1, employees a and a can be transplanted in one step; if it is 2 — in 3 steps.
- Let k be the number of steps for permutation a and b between which there are s edges. Consider a path from $s+1$ edges:

$$(a - v_1 - v_2 - \ldots - v_s) - b.$$

- Select the part from a to v_s and rearrange the numbers at vertices a and v_s (to k steps):

Subtask 2. Recursive solution $O(n^2 \cdot 2^s)$

- Using the bfs-algoritm we find the shortest path from a to b. If there is no such path, output -1.
- Let this path consist of $s+1$ edges

$$a - v_1 - v_2 - \ldots - v_s - b.$$

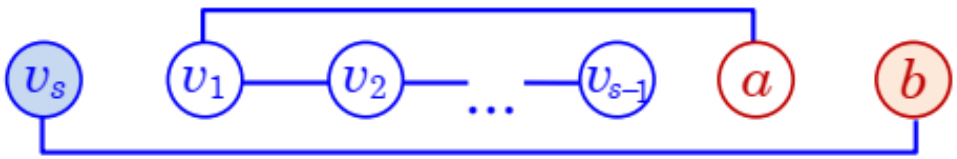
- For example, if the path length is 1, employees a and a can be transplanted in one step; if it is 2 — in 3 steps.
- Let k be the number of steps for permutation a and b between which there are s edges. Consider a path from $s+1$ edges:

$$(a - v_1 - v_2 - \ldots - v_s) - b.$$

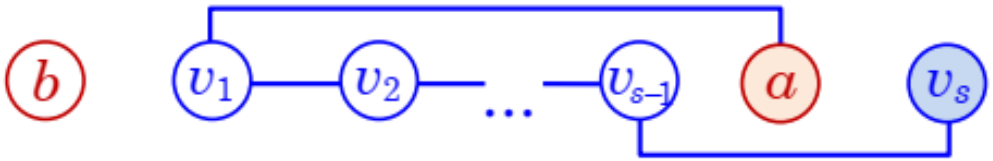
- Select the part from a to v_s and rearrange the numbers at vertices a and v_s (to k steps):

Subtask 2. Recursive solution $O(n^2 \cdot 2^s)$

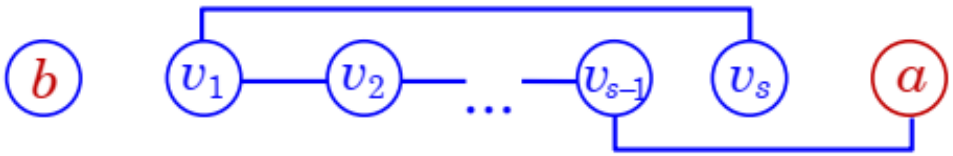
- In one step, rearrange the numbers b and v_s:



- Let's use recursion and change numbers in vertices a and v_s of the path $(a - v_1 - v_2 - \ldots - v_s) - b$:

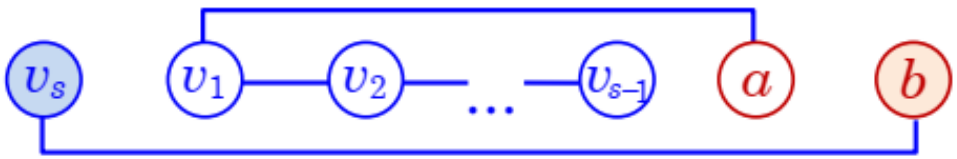


- Select the part from a to v_s and rearrange the numbers at vertices a and v_s (to k steps):

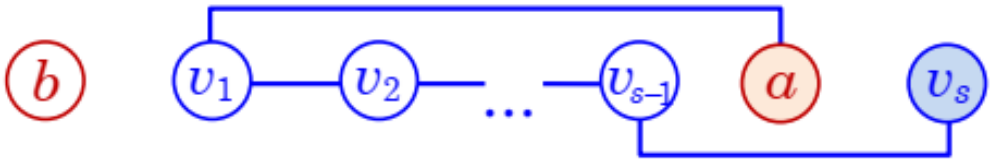


Subtask 2. Recursive solution $O(n^2 \cdot 2^s)$

- In one step, rearrange the numbers b and v_s:



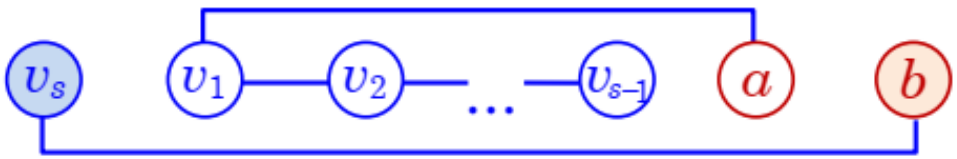
- Let's use recursion and change numbers in vertices a and v_s of the path $(a - v_1 - v_2 - \ldots - v_s) - b$:



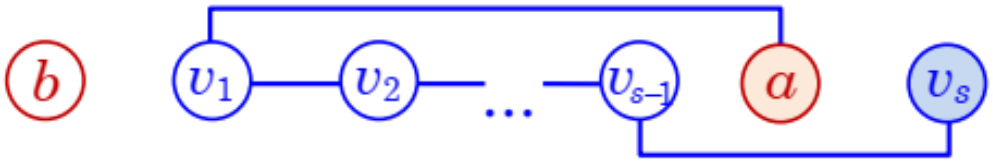
- Select the part from a to v_s and rearrange the numbers at vertices a and v_s (to k steps):

Subtask 2. Recursive solution $O(n^2 \cdot 2^s)$

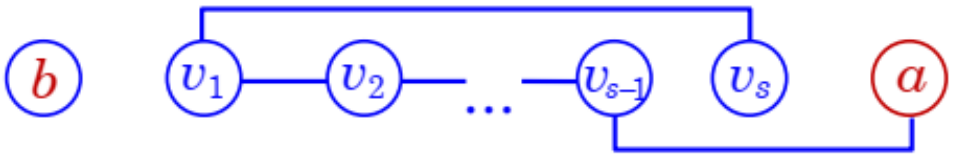
- In one step, rearrange the numbers b and v_s:



- Let's use recursion and change numbers in vertices a and v_s of the path $(a - v_1 - v_2 - \ldots - v_s) - b$:



- Select the part from a to v_s and rearrange the numbers at vertices a and v_s (to k steps):



Subtask 2. Recursive solution $O(n^2 \cdot 2^s)$

- This solution requires $k + 1 + k = 2k + 1$ steps to permute two numbers-ratings a and b.
- For example, for $k = 1$, the number of steps is three; the next number of steps is $2 \cdot 3 + 1 = 7$, for $k = 4$ will be $2 \cdot 7 + 1 = 15$ steps and so on.
- Total number of steps is $2^s - 1$.
- The complexity of the algorithm is $O(n^2 \cdot 2^s)$, where s is the maximum length of the path between vertexes a and b ($s < n$).

Comment. Connection with the task "Hanoi towers"

- The task "Hanoi towers" is solved using similar recursion!
- Solution comes to solving the task with the smallest arguments.

Subtask 4. Full solution $O(n^3)$

Cyclic shift of the array

- The main idea is a cyclic shift of numbers v_i along the path between vertices a and b:
- Illustration on the example of path $a - v_1 - v_2 - v_3 - b$. First, in $s = 3$ steps, perform a cyclic shift to the right of the first $s + 1 = 4$ elements of this path:

a	$-$	v_1	$-$	v_2	$-$	v_3	$-$	b	a	v_1
v_1	$-$	a		v_2	$-$	v_3	$-$	b	v_1	v_2
v_2		a	$-$	v_1		v_3	$-$	b	v_2	v_3
v_3		a	$-$	v_1	$-$	v_2		b		

Subtask 4. Full solution $O(n^3)$

- Change numbers in vertexes v_3 and b:

$$\begin{array}{ccc|cc} v_3 & a - v_1 - v_2 & b & v_3 & b \\ \hline b & a - v_1 - v_2 - v_3 & & & \end{array}$$

- Consider the path $a - v_1 - v_2 - v_3$ in the "reverse" direction. In $s = 3$ steps we perform a cyclic shift to the left $s + 1 = 4$ elements of this path: $v_1 - v_2 - v_3 \quad a$.
- As a result, we get the desired location of vertices a and b:

$$b \quad v_1 - v_2 - v_3 \quad a.$$

- The total number of steps is $2s - 1$, where s is the length of the path between vertices a and b.
- The final complexity is $O(n^3)$.

Solution

Questions?