Lab 7
Isa Dzhumabaev

Q1: Were your predictions correct?

Yes and no. I used some of my work from Intro to AI class that I passed last year and when I were designing that problem I kind of created some optimal paths that I wanted algorithm to detect. So, I predicted some parts of solutions as I just remembered them, but in some places algorithm made different decisions.
I attached the design report that I made last year for Intro to AI class, you can find it in the end of this document.

Q2: Based on your experiments, is one of these search methods better in terms of taking fewer steps to find the goal?

I think this depends on a problem. Some problems are better solved by DFS if optimal solution lays in first branch so it just makes one dive and finds the solution. In case solution lays on shallowest depth on rightmost child of root BFS would be a lot better choice as it will not spend time trying to fully explore all the left nodes.

Q3: How is time/space relevant in this search and graph context?

I already answered time question in Q2.
BFS is less efficient in terms of memory.
DFS requires less memory because it only keeps track of previous node, so it just stores a chain from root to last node to be able to return back and explore next branch. BFS requires to store all the nodes of current depth in a queue. Suppose B is a branching factor and D is current depth, then BFS needs to store B^D nodes on D-th depth, while DFS would only need D nodes.

Q4: List at least 3 sample problems you've explored and how can you stress test them in terms of time/space constraints?

First problem is of course Karakol-Osh problem that I designed last year.

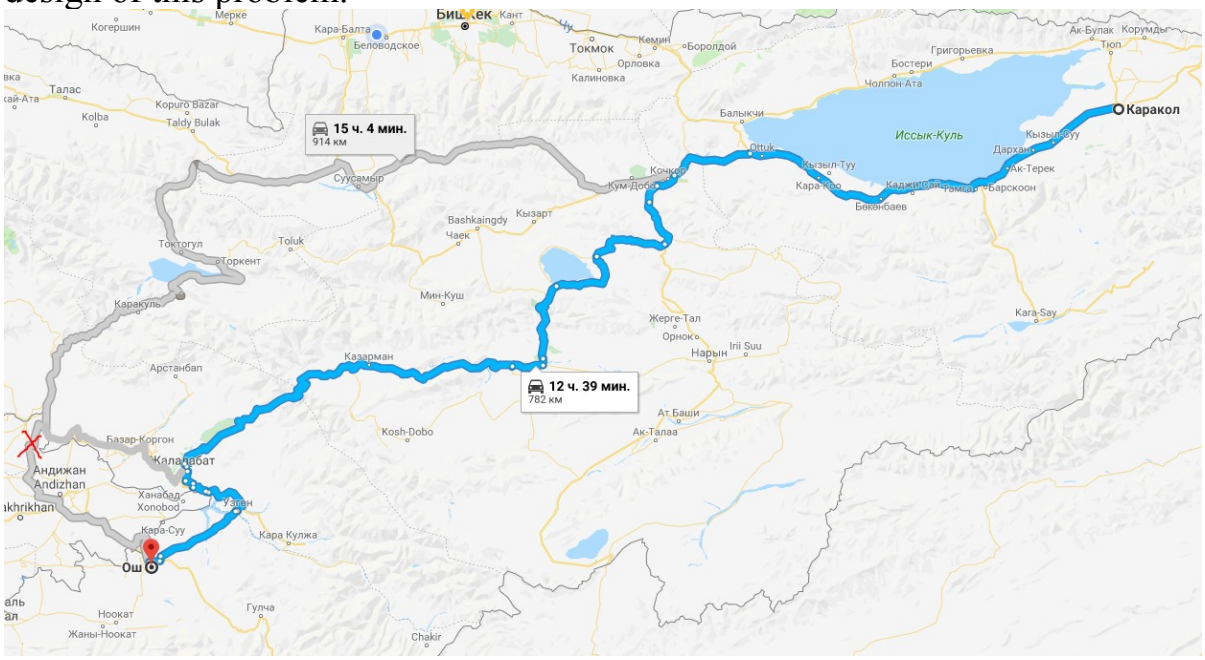
Second problem is Eulerian Path. I solved it during my freshman year, you can check my solution here: https://github.com/Rossolinyisanautist/Isa-Dzhumabaev/tree/master/Logic/Project.%20Task%203/src

Third problem is graph connectivity test, I also tried to solve it. I used DFS for that, you can check my code here: https://github.com/Rossolinyisanautist/Isa-Dzhumabaev/blob/master/Logic/Project.%20Task%202/src/Main.java

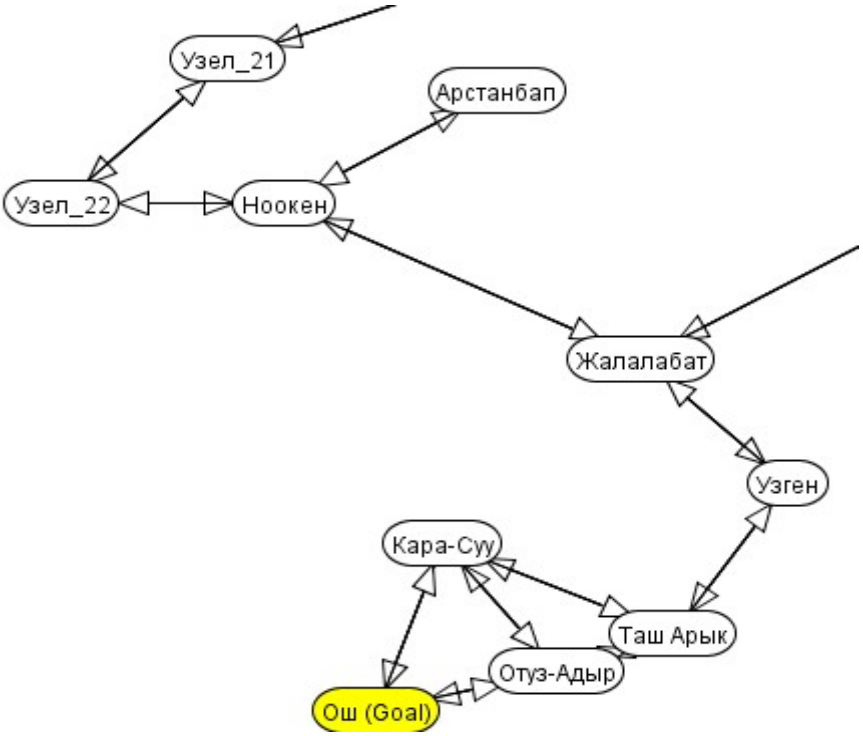
(These are not really good solutions as it was my first year and I was very bad at programming)

Problem design report from Intro to AI class I took last year.

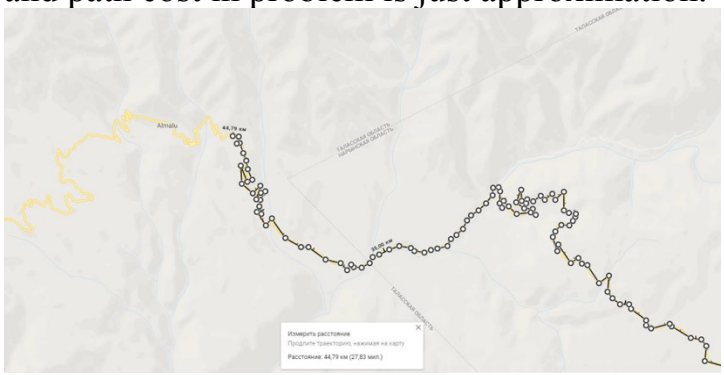
Two main ways to Osh, that were suggested by Google maps are used in the design of this problem.



In these map I used only those roads that are on the territory of Kyrgyzstan. That is why second road is modified in my design. You can see that there is a red cross on the map and way to Osh goes through to «Ноокен» and «Жалалабат» node reaches «Ош».

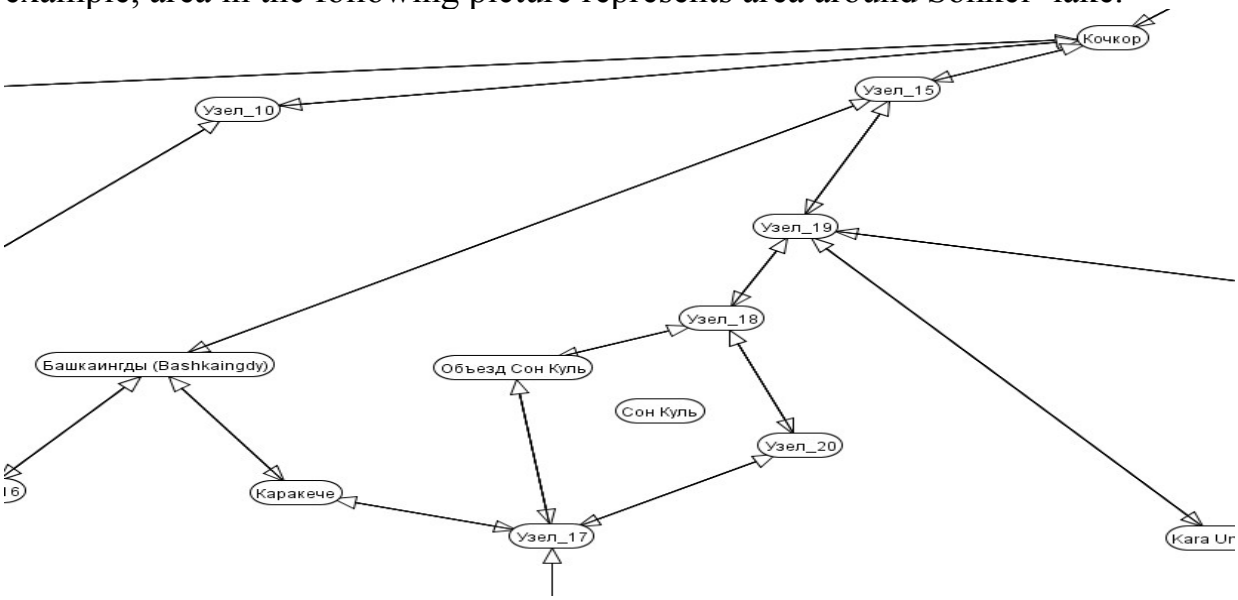


Path cost is based on distance measurement function in Google maps. 1 path cost unit is equals to 1 kilometer. Of course these measurements are very rough and path cost in problem is just approximation.

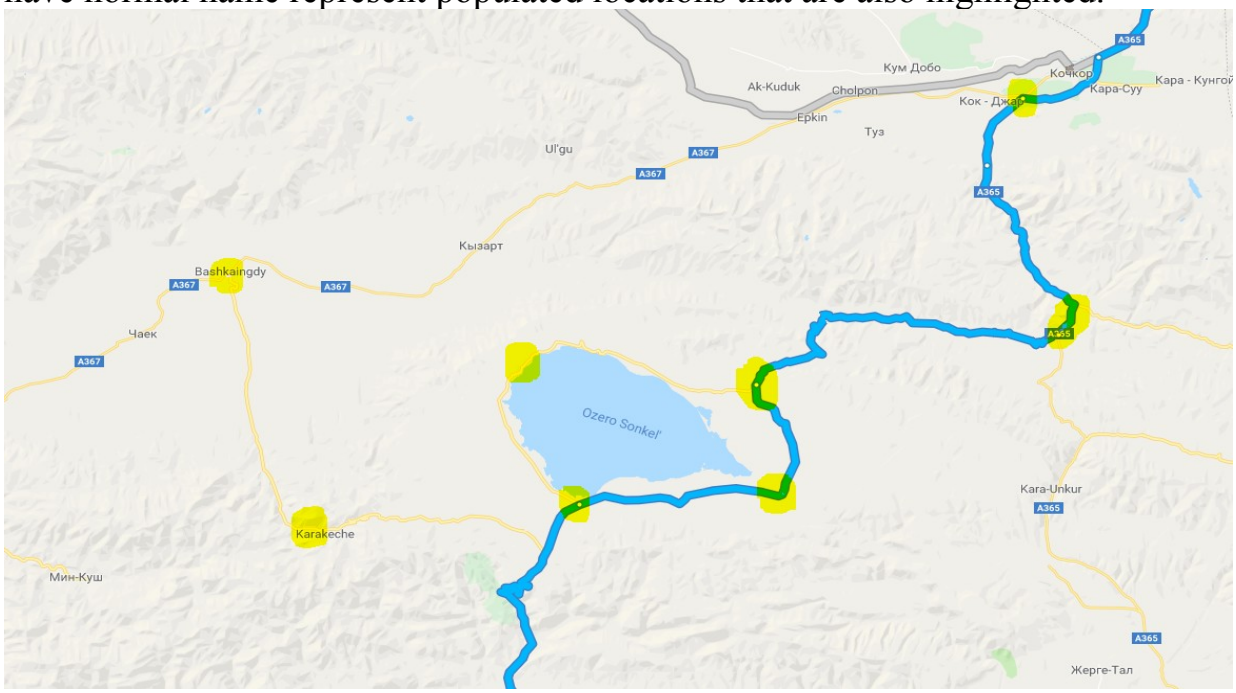


In the actual problem you can see that there is a lot of nodes called «Узел». These are just to show a form of some roads, these don't affect path cost. For

example, area in the following picture represents area around Sonkel' lake.

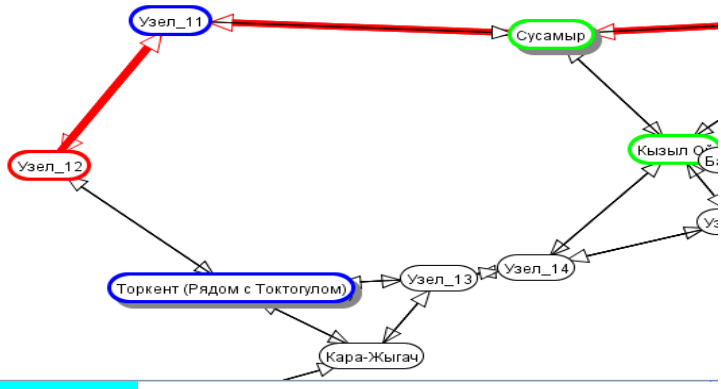


Nodes with name «Узел» match highlighted spots on the map and those that have normal name represent populated locations that are also highlighted.



Search performance

I used five strategies of search: Depth first, Breadth First, Lowest Cost, Best first, Heuristic Depth First, A*. Each of these strategies are tested with None Pruning, Loop Detection and Multiple Path Pruning with 2000 as maximum step limit. Some search strategies ended up with infinite loop in this area.



Best records

Minimum path cost: 714.81 km in A* with Multiple Path Pruning.
Minimum node expanded: 21 Nodes in Heuristic Depth First with Multiple Path Pruning.
Minimum steps: less than 60 steps in Heuristic Depth First with Multiple Path Pruning.

Statistics

Depth First

Pruning mode	Steps	Cost	Nodes Expanded	Result
None	> 2000	-	-	fail
Multiple Path	< 100	1019.17	63	success
Loop Detection	< 100	1019.17	63	success

Breadth First

Pruning mode	Steps	Cost	Nodes Expanded	Result
None	> 2000	-	-	fail
Multiple Path	< 500	753.93	64	success
Loop Detection	< 2000	-	-	fail

Lowest Cost Search

Pruning mode	Steps	Cost	Nodes Expanded	Result
None	> 2000	-	-	fail

Multiple Path	< 200	714.81	63	success
Loop Detection	< 2000	-	-	fail

Best First Search

Pruning mode	Steps	Cost	Nodes Expanded	Result
None	> 2000	-	-	fail
Multiple Path	< 200	753.93	64	success
Loop Detection	< 2000	-	-	fail

Heuristic Depth First Search

Pruning mode	Steps	Cost	Nodes Expanded	Result
None	> 2000	-	-	fail
Multiple Path	< 60	1019.27	21	success
Loop Detection	< 60	1019.27	21	fail

A*

Pruning mode	Steps	Cost	Nodes Expanded	Result
None	> 2000	-	-	fail
Multiple Path	< 200	714.27	63	success
Loop Detection	< 2000	-	-	fail