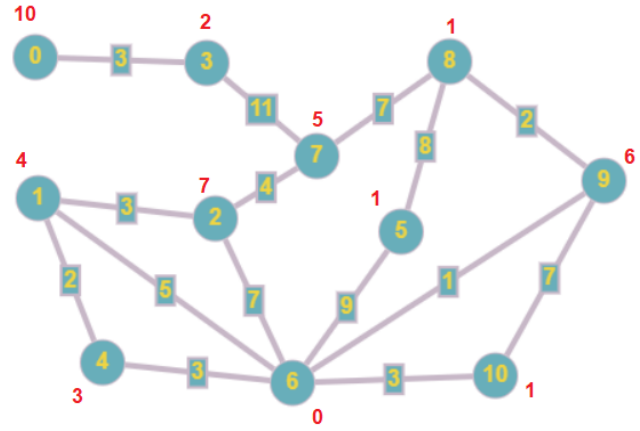
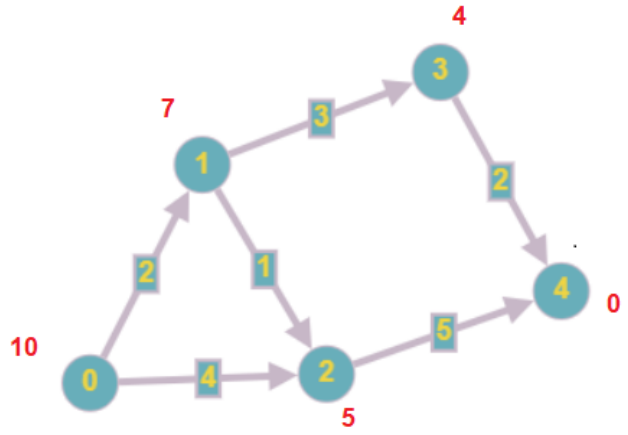
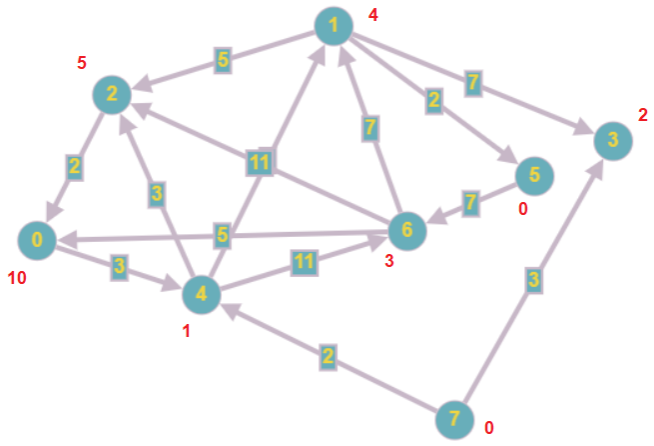
REPORT - LAB 01 - SEARCH STRATEGIES

Prepared by: 19127216 - Đặng Hoàn Mỹ
Instructors: Dr. Nguyen Ngoc Thao, Ph.D

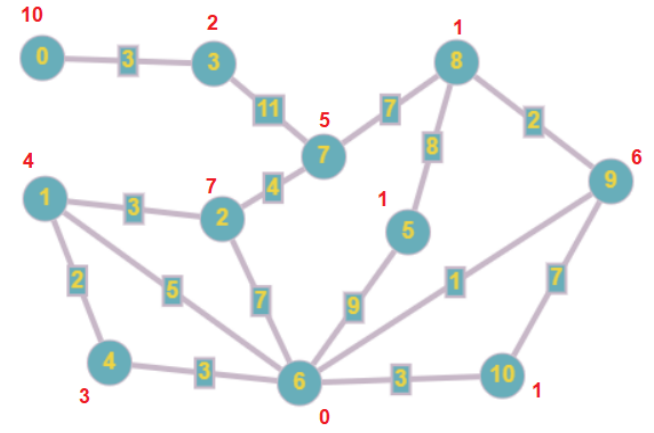
I. Run algorithms in 3 different graphs

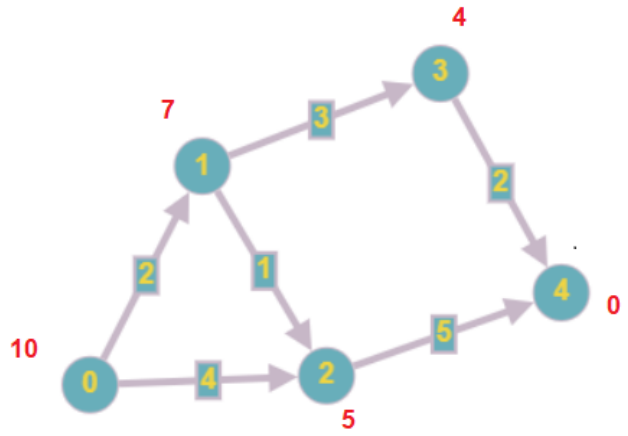
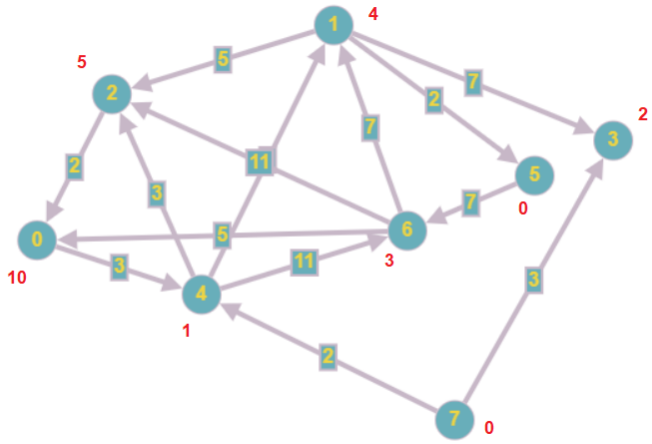
1. Breadth-first search (BFS)

Graph	Input file *input.txt*	Output file *output.txt*
	11 10 0 0 0 0 0 3 0 0 0 0 0 0 0 0 0 3 0 2 0 5 0 0 0 0 0 3 0 0 0 0 7 4 0 0 0 3 0 0 0 0 0 0 11 0 0 0 0 2 0 0 0 0 3 0 0 0 0 0 0 0 0 0 0 9 0 8 0 0 0 5 7 0 3 9 0 0 0 1 3 0 0 4 11 0 0 0 0 7 0 0 0 0 0 0 0 8 0 7 0 2 0 0 0 0 0 0 0 1 0 2 0 7 0 0 0 0 0 0 3 0 0 7 0 10 4 7 2 3 1 0 5 1 6 1	10 6 9 1 2 4 5 8 7 3 10 6 2 7 3 0
	5 0 4 0 0 2 4 0 0 0 0 1 3 0 0 0 0 0 5 0 0 0 0 2 0 0 0 0 0 10 7 5 4 0	0 1 2 0 2 4

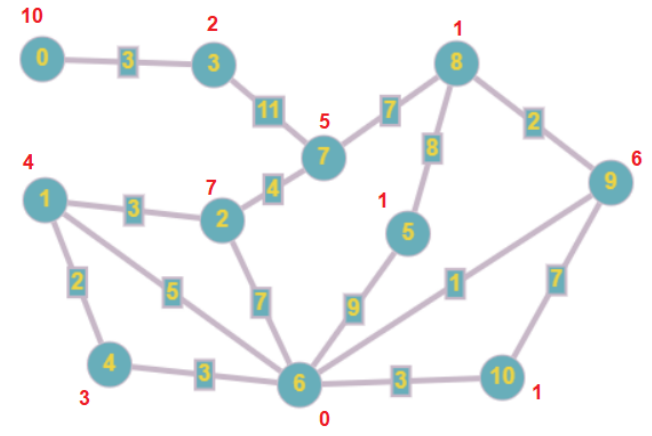
	8 0 3 0 0 0 0 0 3 0 0 0 0 0 5 7 0 2 0 0 2 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 7 3 0 0 0 11 0 0 0 0 0 0 0 7 0 5 7 11 0 0 0 0 0 0 0 0 3 2 0 0 0 10 4 5 2 1 0 3 0	0 4 1 0 4 1 3

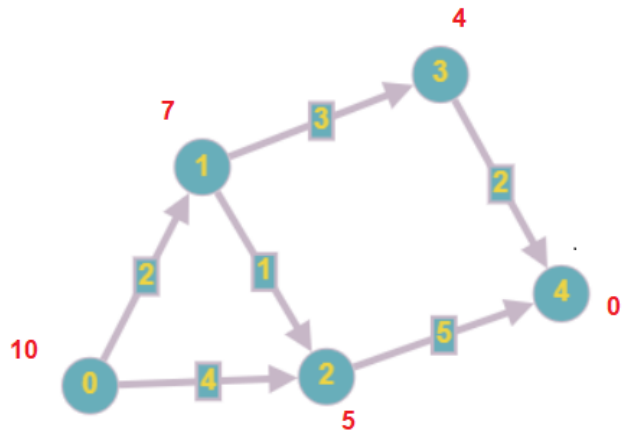
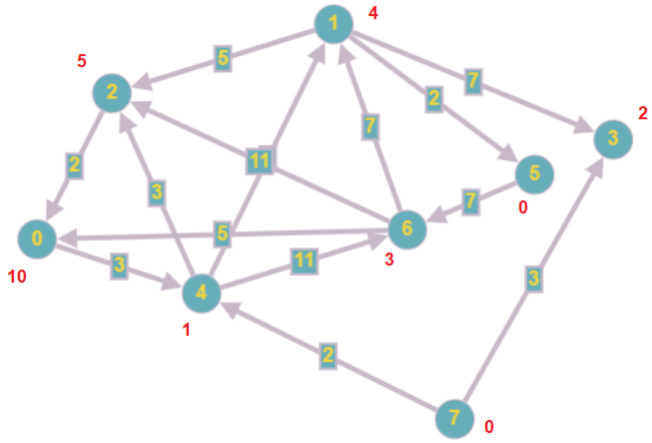
2. Tree-search depth-first search (DFS)

Graph	Input file *input.txt*	Output file *output.txt*
	11 10 0 1 0 0 0 3 0 0 0 0 0 0 0 0 0 3 0 2 0 5 0 0 0 0 0 3 0 0 0 0 7 4 0 0 0 3 0 0 0 0 0 0 11 0 0 0 0 2 0 0 0 0 3 0 0 0 0 0 0 0 0 0 0 0 9 0 8 0 0 0 5 7 0 3 9 0 0 0 1 3 0 0 4 11 0 0 0 0 7 0 0 0 0 0 0 0 0 8 0 7 0 2 0 0 0 0 0 0 0 0 1 0 2 0 7 0 0 0 0 0 0 0 3 0 0 7 0 10 4 7 2 3 1 0 5 1 6 1	10 6 1 2 7 3 0 10 6 1 2 7 3 0

	5 0 4 1 0 2 4 0 0 0 0 1 3 0 0 0 0 0 5 0 0 0 0 2 0 0 0 0 0 10 7 5 4 0	0 1 2 4 0 1 2 4
	8 0 3 1 0 0 0 0 3 0 0 0 0 0 5 7 0 2 0 0 2 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 7 3 0 0 0 11 0 0 0 0 0 0 0 7 0 5 7 11 0 0 0 0 0 0 0 0 3 2 0 0 0 10 4 5 2 1 0 3 0	0 4 1 2 3 0 4 1 3

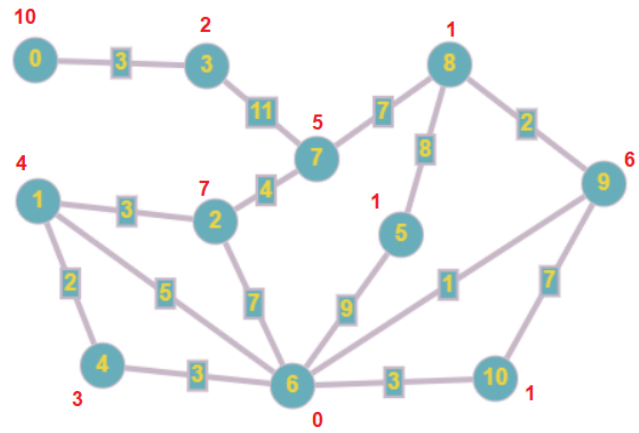
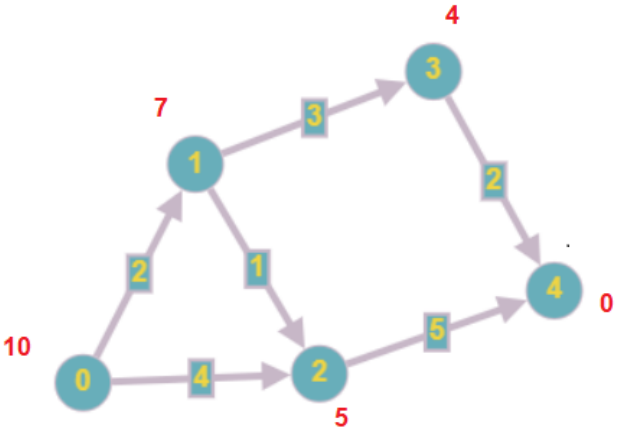
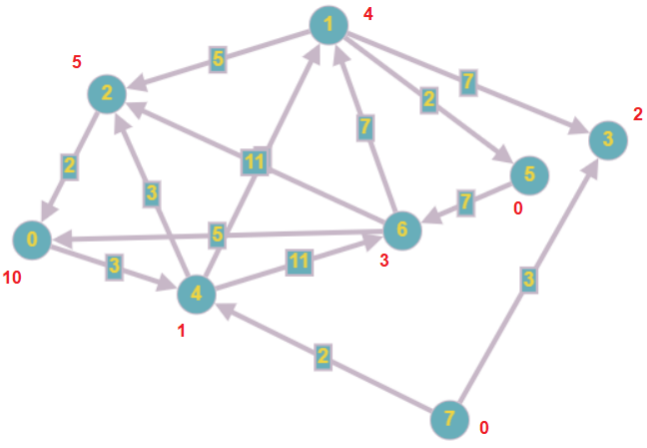
3. Uniform-cost search (UCS)

Graph	Input file *input.txt*	Output file *output.txt*
	11 10 0 2 0 0 0 3 0 0 0 0 0 0 0 0 0 3 0 2 0 5 0 0 0 0 0 3 0 0 0 0 7 4 0 0 0 3 0 0 0 0 0 0 11 0 0 0 0 2 0 0 0 0 3 0 0 0 0 0 0 0 0 0 0 9 0 8 0 0 0 5 7 0 3 9 0 0 0 1 3 0 0 4 11 0 0 0 0 7 0 0 0 0 0 0 0 8 0 7 0 2 0 0 0 0 0 0 0 1 0 2 0 7	10 6 9 4 8 1 2 5 7 3 0 10 6 9 8 7 3 0

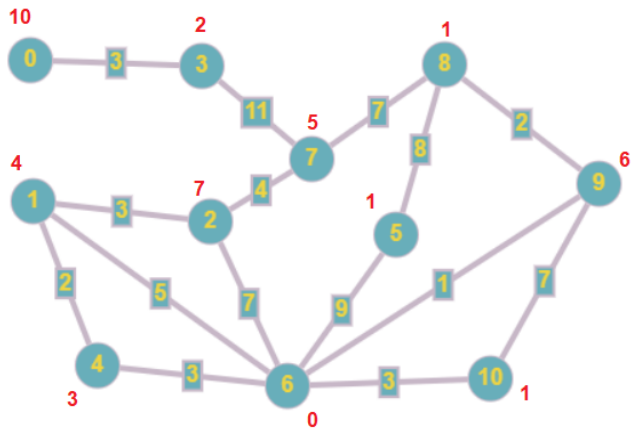
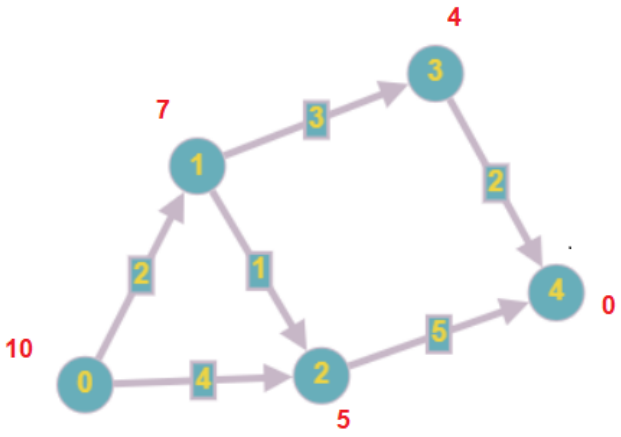
	0 0 0 0 0 0 3 0 0 7 0 10 4 7 2 3 1 0 5 1 6 1	
	5 0 4 2 0 2 4 0 0 0 0 1 3 0 0 0 0 0 5 0 0 0 0 2 0 0 0 0 0 10 7 5 4 0	0 1 2 3 4 0 1 3 4
	8 0 3 2 0 0 0 0 3 0 0 0 0 0 5 7 0 2 0 0 2 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 7 3 0 0 0 11 0 0 0 0 0 0 0 7 0 5 7 11 0 0 0 0 0 0 0 0 3 2 0 0 0 10 4 5 2 1 0 3 0	0 4 2 1 5 6 3 0 4 1 3

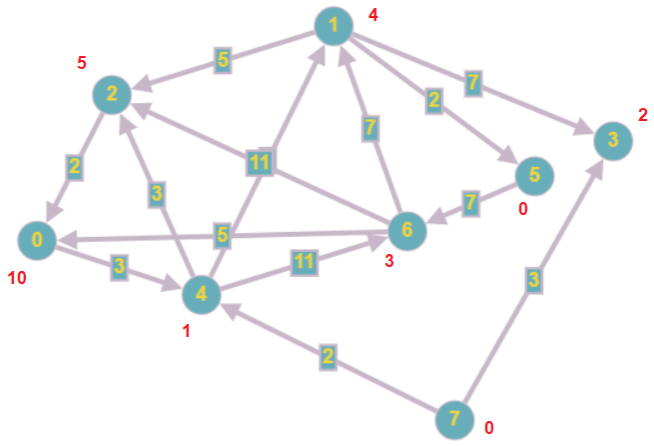
4. Iterative deepening search (IDS)

Graph	Input file *input.txt*	Output file *output.txt*

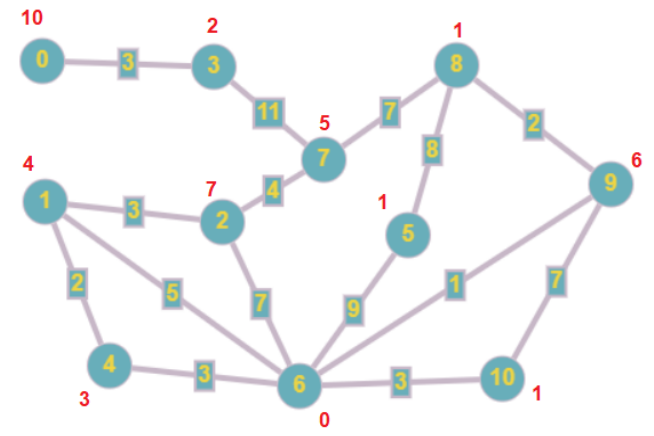
	11 10 0 3 0 0 0 3 0 0 0 0 0 0 0 0 0 3 0 2 0 5 0 0 0 0 0 3 0 0 0 0 7 4 0 0 0 3 0 0 0 0 0 0 11 0 0 0 0 2 0 0 0 0 3 0 0 0 0 0 0 0 0 0 0 9 0 8 0 0 0 5 7 0 3 9 0 0 0 1 3 0 0 4 11 0 0 0 0 7 0 0 0 0 0 0 0 8 0 7 0 2 0 0 0 0 0 0 0 1 0 2 0 7 0 0 0 0 0 0 3 0 0 7 0 10 4 7 2 3 1 0 5 1 6 1	10 10 6 9 10 6 1 2 4 5 9 9 6 8 10 6 1 2 4 2 1 7 4 1 5 8 9 8 9 6 1 2 4 5 8 5 7 10 6 1 2 7 4 2 1 4 7 3 8 4 1 2 5 8 7 9 9 8 5 7 9 6 1 2 4 2 1 7 4 1 5 8 8 5 6 7 2 3 10 6 1 2 7 3 8 4 2 1 4 7 3 0 10 6 2 7 3 0
	5 0 4 3 0 2 4 0 0 0 0 1 3 0 0 0 0 0 5 0 0 0 0 2 0 0 0 0 0 10 7 5 4 0	0 0 1 2 0 1 2 3 2 4 0 2 4
	8 0 3 3 0 0 0 0 3 0 0 0 0 0 5 7 0 2 0 0 2 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 7 3 0 0 0 11 0 0 0 0 0 0 0 7 0 5 7 11 0 0 0 0 0 0 0 0 3 2 0 0 0 10 4 5 2 1 0 3 0	0 0 4 0 4 1 2 6 0 4 1 2 3 0 4 1 3

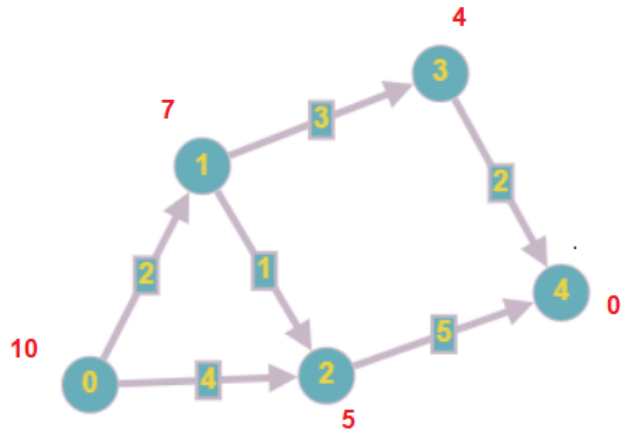
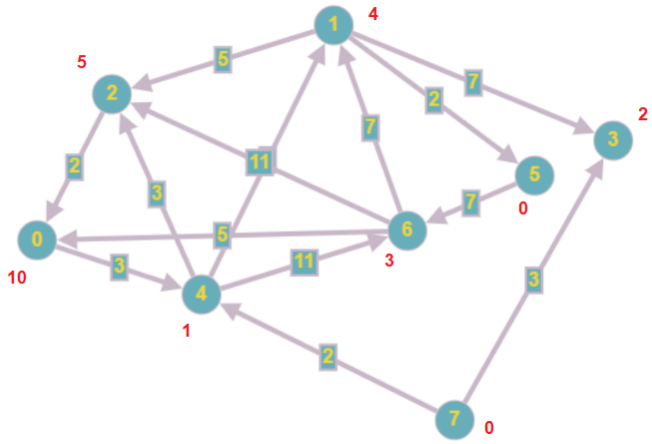
5. Greedy best-first search (GBFS)

Graph	Input file *input.txt*	Output file *output.txt*
	11 10 0 4 0 0 0 3 0 0 0 0 0 0 0 0 0 3 0 2 0 5 0 0 0 0 0 3 0 0 0 0 7 4 0 0 0 3 0 0 0 0 0 0 11 0 0 0 0 2 0 0 0 0 3 0 0 0 0 0 0 0 0 0 0 9 0 8 0 0 0 5 7 0 3 9 0 0 0 1 3 0 0 4 11 0 0 0 0 7 0 0 0 0 0 0 0 8 0 7 0 2 0 0 0 0 0 0 0 1 0 2 0 7 0 0 0 0 0 0 3 0 0 7 0 10 4 7 2 3 1 0 5 1 6 1	10 6 5 8 4 1 7 3 9 2 10 6 5 8 7 3 0
	5 0 4 4 0 2 4 0 0 0 0 1 3 0 0 0 0 0 5 0 0 0 0 2 0 0 0 0 0 10 7 5 4 0	0 2 0 2 4

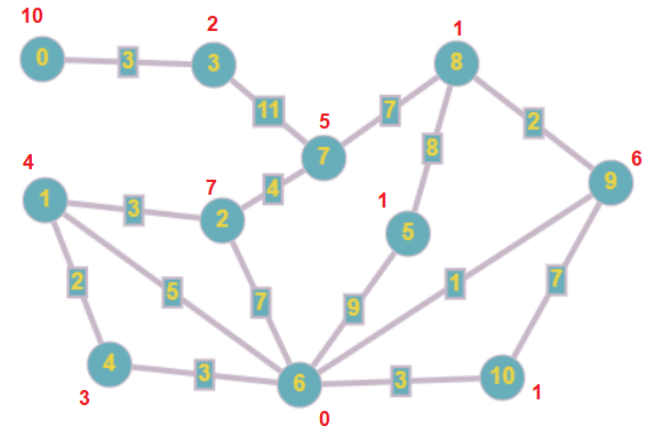
	8 0 3 4 0 0 0 0 3 0 0 0 0 0 5 7 0 2 0 0 2 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 7 3 0 0 0 11 0 0 0 0 0 0 0 7 0 5 7 11 0 0 0 0 0 0 0 0 3 2 0 0 0 10 4 5 2 1 0 3 0	0 4 6 1 5 0 4 1 3

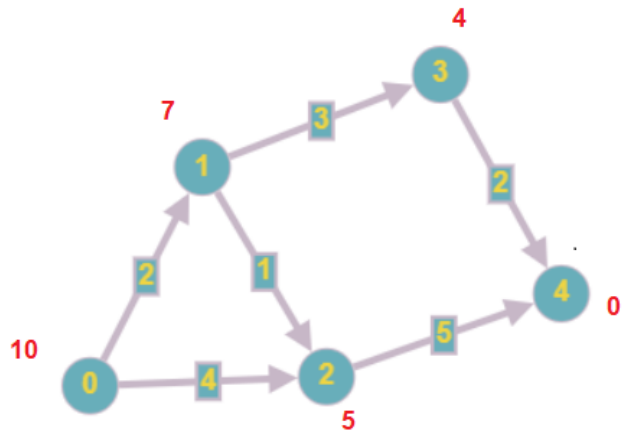
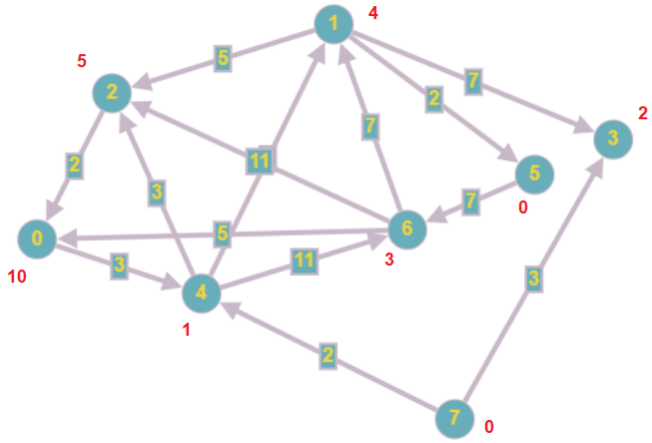
6. Graph-search A* (AStar)

Graph	Input file *input.txt*	Output file *output.txt*
	11 10 0 5 0 0 0 3 0 0 0 0 0 0 0 0 0 3 0 2 0 5 0 0 0 0 0 3 0 0 0 0 7 4 0 0 0 3 0 0 0 0 0 0 11 0 0 0 0 2 0 0 0 0 3 0 0 0 0 0 0 0 0 0 0 0 9 0 8 0 0 0 5 7 0 3 9 0 0 0 1 3 0 0 4 11 0 0 0 0 7 0 0 0 0 0 0 0 0 8 0 7 0 2 0 0 0 0 0 0 0 0 1 0 2 0 7 0 0 0 0 0 0 0 3 0 0 7 0 10 4 7 2 3 1 0 5 1 6 1	10 6 4 9 8 1 5 2 7 3 0 10 6 9 8 7 3 0

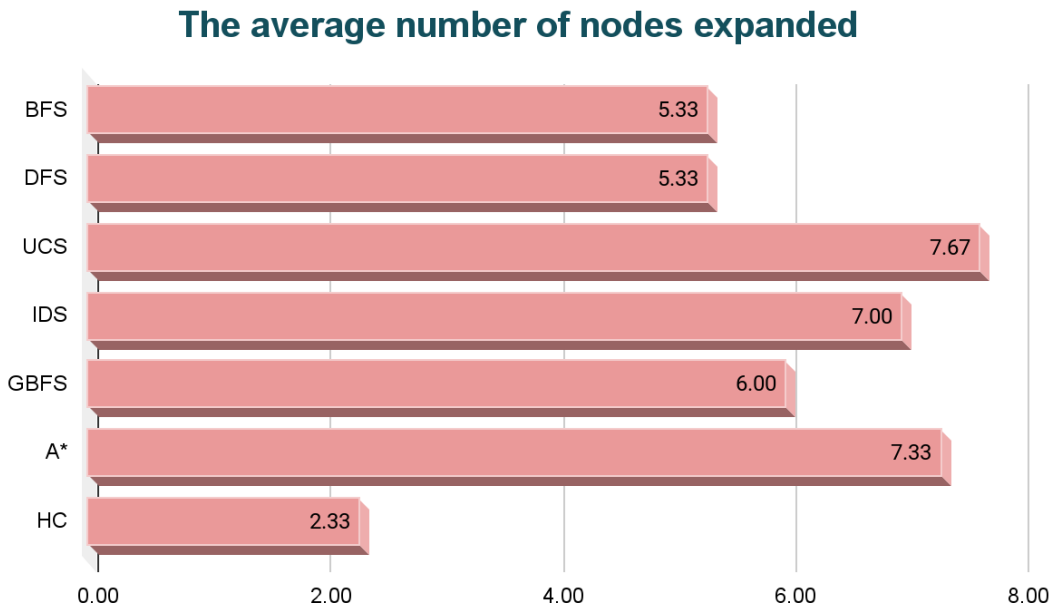
	5 0 4 5 0 2 4 0 0 0 0 1 3 0 0 0 0 0 5 0 0 0 0 2 0 0 0 0 0 10 7 5 4 0	0 1 2 4 0 1 2 4
	8 0 3 5 0 0 0 0 3 0 0 0 0 0 5 7 0 2 0 0 2 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 7 3 0 0 0 11 0 0 0 0 0 0 0 7 0 5 7 11 0 0 0 0 0 0 0 0 3 2 0 0 0 10 4 5 2 1 0 3 0	0 4 2 1 5 6 3 0 4 1 3

7. First-choice Hill-climbing (HC)

Graph	Input file *input.txt*	Output file *output.txt*
	11 10 0 6 0 0 0 3 0 0 0 0 0 0 0 0 0 3 0 2 0 5 0 0 0 0 0 3 0 0 0 0 7 4 0 0 0 3 0 0 0 0 0 0 11 0 0 0 0 2 0 0 0 0 3 0 0 0 0 0 0 0 0 0 0 9 0 8 0 0 0 5 7 0 3 9 0 0 0 1 3 0 0 4 11 0 0 0 0 7 0 0 0 0 0 0 0 8 0 7 0 2 0 0 0 0 0 0 0 1 0 2 0 7	10 6 No path.

	0 0 0 0 0 0 3 0 0 7 0 10 4 7 2 3 1 0 5 1 6 1	
	5 0 4 6 0 2 4 0 0 0 0 1 3 0 0 0 0 0 5 0 0 0 0 2 0 0 0 0 0 10 7 5 4 0	0 2 4 0 2 4
	8 0 3 0 0 0 0 3 0 0 0 0 0 5 7 0 2 0 0 2 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 7 3 0 0 0 11 0 0 0 0 0 0 0 7 0 5 7 11 0 0 0 0 0 0 0 0 3 2 0 0 0 10 4 5 2 1 0 3 0	0 4 No path.

II. General statements



- In most algorithms, more than half of the nodes in the graph are reached, except the Hill-Climbing algorithm because the finding for the next node done at random.
- The First Choice Hill-Climbing algorithm that I choose to implement, is about randomizing a vertex that has a better state, so the number of nodes depends on the number that had been randomized, it seems that search the least nodes - better than others. But it is easy to fall into the case of not finding the path.
- With algorithmic goals such as the shortest path with the lowest cost or minimum heuristic value as UCS, GBFS and A*, nodes must ensure that are all traversed to find a path that satisfies that goal; so these two algorithms are at the top of the chart (>= 6).
- IDS algorithm with the requirement of traversing the vertices of the graph by depth, the higher level the goal is at, the more nodes are approved; so it is clearly at the top 3 of the average number of nodes expanded.
- BFS is more suitable for searching vertices that are closer to the given source although DFS is more suitable when there are solutions away from the source. Therefore, in my given test-case graphs which have the far-away goal and also the close goal, it seems similar in the average of the number of nodes expanded.

III. References

- *How to Implement Breadth-First Search*. (2019). Scrapbook. https://stephanosterburg.gitbook.io/scrapbook/coding/python/how-to-implement-breadth-first-search-in-python
- *Depth First Search - Find path, Python 3 - rextester*. (2018). Rextester.Com. https://rextester.com/discussion/CIW47460/Depth-First-Search-Find-path
- K. (2019, December 25). *kartiikthakur/ArtificialIntelligence_Searches*. GitHub. https://github.com/kartiikthakur/ArtificialIntelligence_Searches
- GeeksforGeeks. (2016, December 22). *Iterative Deepening Search(IDS) or Iterative Deepening Depth First Search(IDDFS)*. https://www.geeksforgeeks.org/iterative-deepening-searchids-iterative-deepening-depth-first-searchiddfs/
- Iterative deepening depth-first search. (2021, January 20). In *Wikipedia*. https://en.wikipedia.org/wiki/Iterative_deepening_depth-first_search
- Nguyen Ngoc Thao. (2021, May 20). *Shared to students - Google Drive* [Slides]. Google Drive. https://drive.google.com/drive/folders/1eZ4DXow6q07xW6sr1pochIQzpUjmwhC2
- RayshineRen, R. (2021, March 2). *RayshineRen/Classcial-Search*. GitHub. https://github.com/RayshineRen/Classcial-Search/tree/1450bcd96eb25600833f4126b4e08031df37532b