Lab 01: Search Strategies

Write a program to find an (optimal) path from the **source** node to the **destination node** on a graph, using either of the following strategies:

- Breadth-first search (**BFS**)
- Tree-search depth-first search (**DFS**): avoid infinite loops by checking new states against those on the path from the root to the current node
- Uniform-cost search (**UCS**)
- Iterative deepening search (**IDS**)
- Greedy best first search (**GBFS**)
- Graph-search A* (**AStar**)
- Steepest-ascent Hill-climbing (**HC**)

❖ Run the above strategies on **a set of 3 different graphs** and report their lists of expanded nodes and found paths to the document.

❖ In the document, draw a bar chart to show the (average) number of nodes expanded by each strategy and then give your own comment.

Input format: the adjacency matrix of the given graph is stored in a text file named **input.txt**, whose format is as follows.

- First line: a positive integer N, which is the number of nodes in the graph.
- Second line: three non-negative integers for the source index, destination index, and the search strategy, respectively. They are separated by white spaces. Indices start from 0.
    - Codes for strategies: 0: BFS, 1: DFS, 2: UCS, 3: IDS, 4: GBFS, 5: A*, and 6: HC
- N next lines present the adjacency matrix, each of which has N integers separated by white spaces. [i, j] = A>0 if there is a link of weight A from node i to node j, and [i, j] = 0 otherwise.
- The last line contains N non-negative integers separated by white spaces, which are heuristic values for N nodes of the graph. These values are designated for the specified goal.

Output format: the result is stored in a text file named **output.txt**, whose format is as follows.

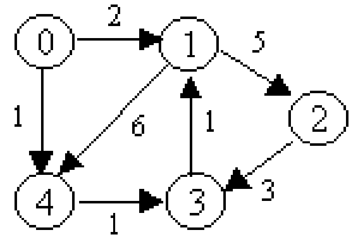
- First line: the list of expanded nodes
- Second line: a list of nodes representing the found path (if there is a path) or a notification of search failure (if there is no path). Nodes should appear following their exact order and they are separated by white spaces.

The **main function** must perform the following basic actions.

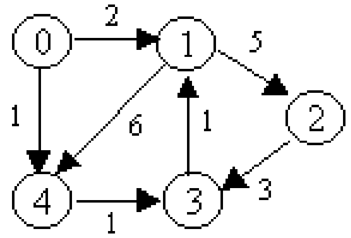
- Read the input data from the input file and store it in appropriate data structures,
- Call the function corresponding to the specified strategy to execute the path finding,
- Functions should be named exactly as what described above, and then
- Show the output.

When there are many candidate nodes of equal possibilities, **the algorithms must visit them following their ascending order of index values**.

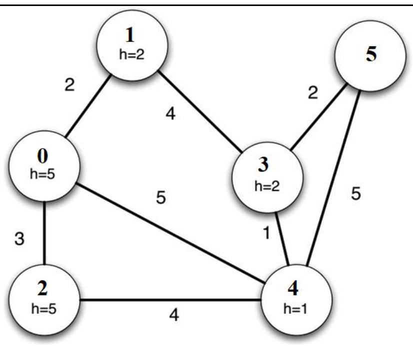
An example of the input graph and its corresponding files **input.txt** and **output.txt**

Graph	input.txt	output.txt	Note
	5 0 3 0 0 2 0 0 1 0 0 5 0 6 0 0 0 3 0 0 1 0 0 0 0 0 0 1 0 0 0 0 0 0	0 1 4 0 4 3	Find a path from node 0 to node 3 using BFS This example ignores the heuristic.

Another example with the same input graph yet a different pair of source – destination and a different search strategy.

Graph	input.txt	output.txt	Note
	5 **3 0 1** 0 2 0 0 1 0 0 5 0 6 0 0 0 3 0 0 1 0 0 0 0 0 0 1 0 0 0 0 0 0	3 1 2 4 No path.	Find a path from node 3 to node 0 using DFS This example ignores the heuristic.

Another example with a different input path

Graph	input.txt	output.txt	Note
	6 0 5 3 0 2 3 0 5 0 2 0 0 4 0 0 3 0 0 0 4 0 0 4 0 0 1 2 5 0 4 1 0 5 0 0 0 2 5 0 5 2 5 2 1 0	0 4 0 4 5	Find a path from node 0 to node 5 using GBFS

Grading

No.	Specifications	Scores (%)
1	Implement search strategies (5% each)	**35**
2	Correct results (both lists of expanded nodes and paths) (5% each)	**35**
3	A comprehensive documentation	**30**
Total		**100**
Note that submissions that do not follow the lab specifications will be rejected (0%).		

Notice

- This is an **INDIVIDUAL** assignment.
- Your program should be programmed in **Python.** Write down your report on a **PDF File.**
- You can use data structure functions/libraries (e.g., queue, stack), yet **you must implement the search algorithms by yourself**.