

Tester: Junyi Ma

Algorithm: Hash

Input: 13

Bottom: Insert

Expect: position 10

Output:

ALGORITHM

Hash

13

Enter a number (0-99)

Insert

Search

Delete

Operation	Input	Output	Runtime (ms)
Insert new number	13	Number Position 10	0.0000

Algorithm: Hash

Input: 44

Bottom: Insert

Expect: position 10, conflict to position 11

Output:

ALGORITHM

Hash

13

44

Enter a number (0-99)

Insert

Search

Delete

Insert new number	44	Number Position 11	0.0000
Insert new number	13	Number Position 10	0.0000

Algorithm: Hash

Input: 13

Bottom: Search

Expect: position 10, value found

Output:

localhost:5173 显示

Value found!

确定

ALGORITHM

Hash

13

13

44

Insert

Search

Delete

Operation	Input	Output	Runtime (ms)
Search number	13	Number Position 10	0.0000
Insert new number	44	Number Position 11	0.0000
Insert new number	13	Number Position 10	0.0000

Algorithm: Hash

Input: 44

Bottom: Delete

Expect: position 11, value deleted!

Output:

localhost:5173 显示

Value deleted!

确定

ALGORITHM

Hash

13

Enter a number (0-99)

Insert

Search

Delete

Operation	Input	Output	Runtime (ms)
Delete number	44	Number Position 11	0.0000

Algorithm: BST

Input: 1,2,5,6

Bottom: CREATE BST

Expect: create new tree with 1,2,5,6

Output:

ALGORITHM

Enter comma separated numbers

1,2,5,6

example: 1,2,3,4

CREATE BST

Insert a node

example: 12

INSERT NODE

Delete a node

example: 2

DELETE NODE

Search a node

example: 9

SEARCH NODE

RETRIEVE PAST INPUTS

Use saved input...

1

2

5

6

Traverse the Tree:

Step 1: Pick how do you want to traverse the BST tree

INORDER

PREORDER

POSTORDER

Step 2: Run it step by step

NEXT STEP

PREVIOUS STEP

Operation	Input	Output	Runtime (ms)
Create new tree	1, 2, 5, 6	1, 2, 5, 6	0.0000

Algorithm: BST

Input: 3

Bottom: INSERT NODE

Expect: 3 will in the left of 5

Output:

ALGORITHM

Enter comma separated numbers

1,2,5,6

example: 1,2,3,4

CREATE BST

Insert a node

3

example: 12

INSERT NODE

Delete a node

example: 2

DELETE NODE

Search a node

example: 9

SEARCH NODE

RETRIEVE PAST INPUTS

Use saved input...

1

2

5

3

6

Traverse the Tree:

Step 1: Pick how do you want to traverse the BST tree

INORDER

PREORDER

POSTORDER

Step 2: Run it step by step

NEXT STEP

PREVIOUS STEP

Operation	Input	Output	Runtime (ms)
Insert new node	3	Node Position 13	0.0000

Algorithm: BST

Input: 5

Bottom: DELETE NODE

Expect: 5 will be removed from the tree

Output:

ALGORITHM

Enter comma separated numbers

1,2,5,6

example: 1,2,3,4

CREATE BST

Insert a node

3

example: 12

INSERT NODE

Delete a node

5

example: 2

DELETE NODE

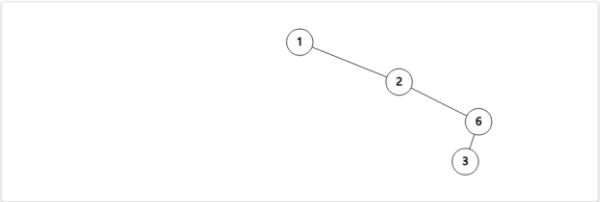
Search a node

example: 9

SEARCH NODE

RETRIEVE PAST INPUTS

Use saved input...



Traverse the Tree:

Step 1: Pick how do you want to traverse the BST tree

INORDER

PRE-ORDER

POSTORDER

Step 2: Run it step by step

NEXT STEP

PREVIOUS STEP

Operation	Input	Output	Runtime (ms)
Create new tree	1, 2, 6, 3	1, 2, 3, 6	0.0100
Delete node	5	Delete Node initial position 7	0.1000

Algorithm: Heap

Input: 1,3,6,8

Bottom: CREATE HEAP

Expect: create new tree with 1,3,6,8

Output:

ALGORITHM

Enter comma separated numbers

1,3,6,8

example: 1,2,3,4

CREATE HEAP

Insert a node

example: 12

INSERT NODE

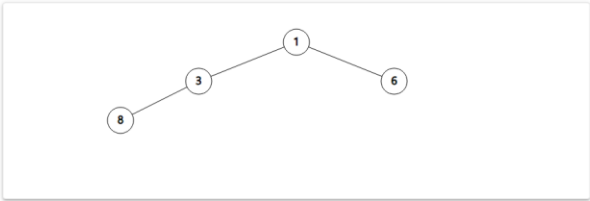
Delete a node

example: 2

DELETE NODE

RETRIEVE PAST INPUTS

Use saved input...



Step 1: Run it step by step

NEXT STEP

PREVIOUS STEP

Step 2: Run Final Heap

FINAL HEAP

Step 3: Extracting Maximum Number

EXTRA HEAP

Operation	Input	Output	Runtime (ms)
Create new tree	1, 3, 6, 8	8, 3, 6, 1	0.1000

Algorithm: Heap

Input: 4

Bottom: INSERT NODE

Expect: 4 will be added into right of 3

Output:

ALGORITHM

Enter comma separated numbers

1,3,6,8

example: 1,2,3,4

CREATE HEAP

Insert a node

4

example: 12

INSERT NODE

Delete a node

example: 2

DELETE NODE

RETRIEVE PAST INPUTS

Use saved input...

```
graph TD; 8((8)) --- 3((3)); 8 --- 6((6)); 3 --- 1((1)); 3 --- 4((4));
```

Step 1: Run it step by step

NEXT STEP

PREVIOUS STEP

Step 2: Run Final Heap

FINAL HEAP

Step 3: Extracting Maximum Number

EXTRA HEAP

Operation	Input	Output	Runtime (ms)
Insert new node	4	Node Position 2	0.0000

Algorithm: Heap

Input: NA

Bottom: NEXT STEP

Expect: 3 and 4 switch places

Output:

ALGORITHM

Enter comma separated numbers

1,3,6,8

example: 1,2,3,4

CREATE HEAP

Insert a node

4

example: 12

INSERT NODE

Delete a node

example: 2

DELETE NODE

RETRIEVE PAST INPUTS

Use saved input...

```
graph TD; 8((8)) --- 4((4)); 8 --- 6((6)); 4 --- 1((1)); 4 --- 3((3));
```

Step 1: Run it step by step

NEXT STEP

PREVIOUS STEP

Step 2: Run Final Heap

FINAL HEAP

Step 3: Extracting Maximum Number

EXTRA HEAP

Operation	Input	Output	Runtime (ms)
Insert new node	4	Node Position 2	0.0000

Algorithm: Dijkstra

Input: 1 3 5 8

1 0 3 0

0 3 4 6

1 3 1 0

Bottom: CREATE

Expect: Graph created by this path

Output:

ALGORITHM

Step 1: Select what kind of graph you want to create
Select Graph Kind
Directed

Step 2: Select how do you want to create the graph
Select Create Kind
Adjacency Matrix

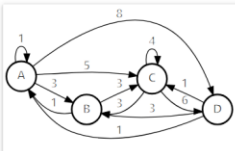
Step 3: Enter Adjacency Matrix
1 3 5 8
1 0 3 0
0 3 4 6
1 3 1 0

CREATE

Shortest Path:

Current Adjacency Matrix:
A B C D
A 1 3 5 8
B 1 0 3 0
C 0 3 4 6
D 1 3 1 0

RETRIEVE PAST INPUTS



Run Dijkstra Step by Step
NEXT STEP PREVIOUS STEP
Rest Dijkstra to the Beginning
RESET
Get Final Result in One Step
FIND SHORTEST PATH

Operation	Input	Output	Runtime (ms)
-----------	-------	--------	--------------

Algorithm: Dijkstra

Input: NA

Bottom: FIND SHORTEST PATH & NEXT STEP

Expect: Animation will lead to the shortest path for each node

Output:

ALGORITHM

Step 1: Select what kind of graph you want to create
Select Graph Kind
Directed

Step 2: Select how do you want to create the graph
Select Create Kind
Adjacency Matrix

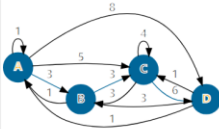
Step 3: Enter Adjacency Matrix
1 3 5 8
1 0 3 0
0 3 4 6
1 3 1 0

CREATE

Shortest Path:

Current Adjacency Matrix:
A B C D
A 1 3 5 8
B 1 0 3 0
C 0 3 4 6
D 1 3 1 0

RETRIEVE PAST INPUTS



Run Dijkstra Step by Step
NEXT STEP PREVIOUS STEP
Rest Dijkstra to the Beginning
RESET
Get Final Result in One Step
FIND SHORTEST PATH

Operation	Input	Output	Runtime (ms)
-----------	-------	--------	--------------

Algorithm: Dijkstra

Input: NA

Bottom: RESET

Expect: Color will be removed

Output:

ALGORITHM

Step 1: Select what kind of graph you want to create

Select Graph Kind

Directed

Step 2: Select how do you want to create the graph

Select Create Kind

Adjacency Matrix

Step 3: Enter Adjacency Matrix

1	3	5	8
1	0	3	0
0	3	4	6
1	3	1	0

CREATE

Shortest Path:

A B C D

0 3 5 8

Current Adjacency Matrix:

	A	B	C	D
A	1	3	5	8
B	1	0	3	0
C	0	3	4	6
D	1	3	1	0

Run Dijkstra Step by Step

NEXT STEP PREVIOUS STEP

Reset Dijkstra to the Beginning

RESET

Get Final Result in One Step

FIND SHORTEST PATH

Operation	Input	Output	Runtime (ms)
-----------	-------	--------	--------------

Input: NA

Bottom: About

Expect: Show all the teammates' information

Output:

Algo-Shadow

Home Algorithm About Contact Login

About Us

My Nguyen
Product Manager
I created the product roadmap, implemented developer workflow and ensure product quality.
✉

Fengyun
UI/UX
I created UI design, and implement our Web UI to make sure our customers have an seamless and comprehensive experience.
✉

Xinyue
Full Stack Engineer
I selected the web framework for our app, and implemented our Web UI. Also, I implemented Dijkstra algorithm.
✉ xinyue57@gmail.com

Junyi
Full Stack Engineer
I contributed to the development of the Hash and Bellman algorithms, ensuring their effective implementation and optimal performance.
✉

Yutong
Full Stack Engineer
I contributed to the development of the Heap and BST algorithms, ensuring their effective implementation and optimal performance.
✉